UNIVERSIDAD TÉCNICA DEL NORTE

Facultad de Ingeniería en Ciencias Aplicadas

Carrera de Ingeniería en Sistemas Computacionales

TESIS PREVIA OBTENCIÓN DEL TITULO DE INGENIERO EN SISTEMAS COMPUTACIONALES

Tema:

"Análisis y Estudio de Tecnología Ruby on Rails con bases de datos Postgres para Aplicaciones Web 2.0"

Aplicativo:

Implementación del Portal Web 2.0 para la Mancomunidad de la Cuenca del Río Mira

AUTOR : CHRISTIAN FERNANDO REALPE ROSERO.

DIRECTOR : ING. MARCO PUSDÁ.

IBARRA – ECUADOR, 2012

Certifico:

Que la Tesis previa a la obtención del título de Ingeniería en Sistemas Computacionales con el tema "**Análisis y Estudio de Tecnología Ruby on Rails con bases de datos Postgres para Aplicaciones Web 2.0**"con el aplicativo **"Implementación del Portal Web 2.0 para la Mancomunidad de la Cuenca del Río Mira"** ha sido desarrollada y terminada en su totalidad por el Sr. Christian Fernando Realpe Rosero con C.C. 100251109-3 bajo mi supervisión para lo cual firmo en constancia.

Atentamente,

Ing. Marco Pusdá
DIRECTOR DE TESIS

UNIVERSIDAD TÉCNICA DEL NORTE
CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE INVESTIGACIÓN
A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

Yo, CHRISTIAN FERNANDO REALPE ROSERO, con cedula de identidad Nro. 100251109-3, manifiesto mi voluntad de ceder a la Universidad Técnica del Norte los derechos patrimoniales consagrados en la ley de propiedad intelectual del Ecuador, articulo 4, 5 y 6, en calidad de autor del trabajo de grado denominado: **"ANÁLISIS Y ESTUDIO DE TECNOLOGÍA RUBY ON RAILS CON BASES DE DATOS POSTGRES PARA APLICACIONES WEB 2.0"** con el aplicativo **"IMPLEMENTACIÓN DEL PORTAL WEB 2.0 PARA LA MANCOMUNIDAD DE LA CUENCA DEL RÍO MIRA"**, que ha sido desarrollada para optar por el título de Ingeniería en Sistemas Computacionales, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En mi condición de autor me reservo los derechos morales de la obra antes mencionada, aclarando que el trabajo aquí descrito es de mi autoría y que no ha sido previamente presentado para ningún grado o calificación profesional.

En concordancia suscribo este documento en el momento que hago entrega del trabajo final en formato impreso y digital a la biblioteca de la Universidad Técnica del Norte

Firma ……..
Nombre: CHRISTIAN FERNANDO REALPE ROSERO
Cédula: 100251109-3
Ibarra a los 28 días del mes de Noviembre del 2012

UNIVERSIDAD TÉCNICA DEL NORTE
BIBLIOTECA UNIVERSITARIA
AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

La UNIVERSIDAD TÉCNICA DEL NORTE dentro del proyecto Repositorio Digital institucional determina la necesidad de disponer los textos completos de forma digital con la finalidad de apoyar los procesos de investigación, docencia y extensión de la universidad.
Por medio del presente documento dejo sentada mi voluntad de participar en este proyecto, para lo cual ponemos a disposición la siguiente investigación:

DATOS DE LA OBRA	
TITULO	"ANÁLISIS Y ESTUDIO DE TECNOLOGÍA RUBY ON RAILS CON BASES DE DATOS POSTGRES PARA APLICACIONES WEB 2.0"
AUTOR	CHRISTIAN FERNANDO REALPE ROSERO
FECHA	28 DE NOVIEMBRE DEL 2012
PROGRAMA	PREGRADO
TITULO POR EL QUE	INGENIERÍA EN SISTEMAS COMPUTACIONALES
DIRECTOR	ING. MARCO PUSDÁ

DATOS DE CONTACTO	
CEDULA DE IDENTIDAD	100251109-3
APELLIDOS Y NOMBRES	CHRISTIAN FERNANDO REALPE ROSERO
DIRECCIÓN	Ibarra Los Ceibos Río Patate 2-90
EMAIL	chriss_realpe3@hotmail.com
TELÉFONO FIJO	062642496
TELÉFONO MOVIL	0991490349 ó 0983500416

2. AUTORIZACIÓN DE USO A FAVOR DE LA UNIVERSIDAD

Yo, CHRISTIAN FERNANDO REALPE ROSERO, con cedula de identidad Nro. 100251109-3, en calidad de autor y titular de los derechos patrimoniales de la obra o trabajo de grado descrito anteriormente, hago entrega del ejemplar respectivo en forma digital y autorizo a la Universidad Técnica del Norte, la publicación de la obra en el Repositorio Digital Institucional y el uso del archivo digital en la biblioteca de la universidad con fines académicos, para ampliar la disponibilidad del material y como apoyo a la educación, investigación y extensión, en concordancia con la Ley de Educación Superior Artículo 143.

Firma ……..
Nombre: CHRISTIAN FERNANDO REALPE ROSERO
Cédula: 100251109-3
Ibarra a los 28 días del mes de Noviembre del 2012

Dedicatoria

A mis Padre Luis G. Realpe V. y a mi Madre Carmen G. Rosero O. que con todo su sacrificio y apoyo incansable, me han ayudado hasta verme realizado profesional y a mis hermanos por su apoyo incondicional.

Agradecimiento

Agradezco a Dios por ser mi guía mi camino en la vida por haber cuidado de mí y darme salud para poder estar aquí, y permitir que pueda cumplir con mis metas y objetivos.

Agradezco a mi padre Luis G. Realpe V. y a mi madre Carmen G. Rosero O. que con su educación me hicieron una persona capaz de ser mejor ya profesionalmente como con sus valores morales sea una mejor persona, con el apoyo tanto moral como económico supieron sacarme adelante gracias y mil gracias, agradezco a mis hermanos Jorge y Javier Realpe Rosero el apoyo del estar ahí cada día.

Agradezco a mis profesores por haberme compartido todo su conocimiento, y así poder enfrentar con dedicación y esmero a los retos que nos ofrece la vida

Agradezco a mi director de Tesis por la guía brindada, para poder lograr este objetivo.

Agradezco a la Universidad Técnica del Norte que con su acogida para mi preparación académica puedo desempeñarme como profesional ante la vida diaria en la cual me encuentro.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

RESUMEN

La presente Tesis tiene como objetivo aplicar la tecnología ruby on rails con patrón de arquitectura MVC y aplicar al desarrollo de aplicaciones web.

Los capítulos que componen la presente Tesis están descritos a continuación.

El capítulo I corresponde a las deficiones en general de la herramienta a estudiar en este caso realizaremos una intruduccion a ruby, características de ruby y el estudio de ruby; veremos la herramientas de ruby on Rails su introducción, definicionsu fucion, su arquitectura de desarrollo y por ultimo la fusión de postgersql con ruby on Rails veremos su introducción, en si la base de datos de postgres y fusión de ruby on Rails con postgresql.

El Capítulo II hablaremos de las herramientas compatibles con rubi on Rails tenemos herramientas afines hay muchas alternativas para trabajar con Ruby on Rails, tanto libres y gratuitas como de pago. A continuación se listan las principales: Aptana: Multiplataforma nació como plugins de eclipse para la edición y desarrollo web. Netbeans: Uno de los más usados, libre y totalmente gratuito. Viene muy bien integrado con JRuby (lo cual es algo lógico pues es un programa de Sun) y lo que se refiere a compatibilidad y soporte de las herramientas antes mencionada

El Capítulo III corresponde al desarrollo de los aplicativos, este incluye las características principales que contienen las aplicaciones: Modulo de Administración de todo el sitio Web y Módulo de los Usuarios; análisis de requerimientos flujos de trabajos, casos de usodiagramas de clases, infraestructura tecnológica; diseño; desarrollo e implementación especificaciones tecnológicas, arquitectura de aplicación; producto instalación, descripciones, administración, evaluación; pruebas finales

El Capítulo IV corresponde a las concluciones y recomendaciones de la presente tesis

SUMMARY

This thesis aims to apply the technology Ruby on Rails MVC architectural pattern and apply to web application development.

The chapters of this thesis are described below.

Chapter I corresponds to the definitions in overall tool to study here will make a intruduccion to ruby, ruby characteristics and studying ruby; see the Ruby on Rails tools introduction, definicionsu Fucion, architecture and development finally merging with Ruby on Rails postgersql see its introduction, if the postgres database and fusion ruby on Rails with postgresql.

Chapter II will discuss the tools compatible with ruby on Rails have many alternatives related tools for working with Ruby on Rails, and free both free and paid. Listed below are the main ones: Aptana: Multiplatform born as eclipse plugins for editing, and web development. Netbeans: One of the most used, free and free. It comes very well integrated with JRuby (which is logical since it is a program of Sun) and regard to compatibility and support tools above

Chapter III is the development of applications, this includes the main features that contain applications: Management Module entire website and Module Users, requirements analysis workflows, case usodiagramas class technology infrastructure; design, development and implementation technology specifications, application architecture, product installation, descriptions, administration, evaluation, final tests

Chapter IV corresponds to concluciones and recommendations of this thesis

INTRODUCCIÓN

El objetivo principal del proyecto es realizar un análisis y estudio de la tecnología Ruby on Rails y la base de datos Postgres. El segundo objetivo principal es construir un Portal Web 2.0 aplicando tecnología Ruby on Rails y base de datos Postgres para la Mancomunidad de la Cuenca del Río Mira que brinde la información necesaria para la divulgación en la zona Norte del Ecuador.

Para realizar esta aplicación se utilizó el lenguaje de programación Ruby on Rails con la base de datos postqresql para la programación se utilizó la herramienta de desarrollo Aptana

Para el desarrollo de la metodología se usó RUP (Rational Unified Process) con los artefactos que se presentan a continuación:

- Flujo de Trabajo
- Casos de Uso
- Diagramas de Clases
- Infraestructura Tecnológica.

CAPITULO I

Ruby on Rails y bases de datos Postgresql

1.1. Qué es Ruby?

1.2. Herramientas de Desarrollo de Ruby on Rails

1.3. Postgresql y la fusion con ruby on rails

1. Ruby on Rails y bases de datos Postgresql.

1.1. Qué es Ruby?

1.1.1. Introducción

Ruby, el lenguaje fue creado por Yukihiro "Matz" Matsumoto, quien empezó a trabajar en Ruby el 24 de febrero de 1993, y lo presentó al público en el año 1995. En el círculo de amigos de Matsumoto se le puso el nombre de "Ruby" (en español rubí) como broma aludiendo al lenguaje de programación "Perl" (perla).

La última versión estable de la rama 1.8 es la 1.8.7_p248, de la rama 1.9 es la 1.9.1_p378 . La versión en 1.9 que incorpora mejoras sustanciales en el rendimiento del lenguaje, que se espera queden reflejadas en la próxima versión estable de producción del lenguaje, Ruby 1.9.0.1 Diferencias en rendimiento entre la actual implementación de Ruby (1.8.6) y otros lenguajes de programación más arraigados han llevado al desarrollo de varias máquinas virtuales para Ruby. Entre ésas se encuentra JRuby, un intento de llevar Ruby a la plataforma Java, y Rubinius, un intérprete modelado basado en las máquinas virtuales de Smalltalk. Los principales desarrolladores han apoyado la máquina virtual proporcionada por el proyecto YARV, que se fusionó en el árbol de código fuente de Ruby el 31 de diciembre de 2006, y se dio a conocer como Ruby 1.9.

El creador del lenguaje, Yukihiro "Matz" Matsumoto, ha dicho que Ruby está diseñado para la productividad y la diversión del desarrollador, siguiendo los principios de una buena interfaz de usuario. Sostiene que el diseño de sistemas necesita enfatizar las necesidades humanas más que las de la máquina.

A menudo la gente, especialmente los ingenieros en computación, se centran en las máquinas. Ellos piensan, "Haciendo esto, la máquina funcionará más rápido. Haciendo esto, la máquina funcionará de manera más eficiente. Haciendo esto..." Están centrados en las máquinas, pero en realidad necesitamos centrarnos en las personas, en cómo hacen programas o cómo manejan las aplicaciones en los ordenadores. Nosotros somos los jefes. Ellos son los esclavos.

Ruby sigue el "principio de la menor sorpresa", lo que significa que el lenguaje debe comportarse de tal manera que minimice la confusión de los usuarios experimentados. Matz ha dicho que su principal objetivo era hacer un lenguaje que le divirtiera él mismo, minimizando el trabajo de programación y la posible confusión. Él ha dicho que no ha aplicado el principio de menor sorpresa al diseño de Ruby, pero sin embargo la frase se ha asociado al lenguaje de programación Ruby. La frase en sí misma ha sido fuente de controversia, ya que los no iniciados pueden tomarla como que la características de Ruby intentar ser similares a las características de otros lenguajes conocidos. En mayo de 2005 en una discusión en el grupo de noticias comp.lang.ruby, Matz trató de distanciar Ruby de la mencionada filosofía, explicando que cualquier elección de diseño será sorprendente para alguien, y que él usa un estándar personal de evaluación de la sorpresa. Si ese estándar personal se mantiene consistente habrá pocas sorpresas para aquellos familiarizados con el estándar.

Matz lo definió de esta manera en una entrevista.

Todo el mundo tiene un pasado personal. Alguien puede venir de Python, otro de Perl, y ellos pueden verse sorprendidos por distintos aspectos del lenguaje. Entonces ellos podrían decir 'Estoy sorprendido por esta característica del lenguaje, así que Ruby viola el principio de la menor sorpresa.' Esperad,

esperad. El principio de la menor sorpresa no es solo para ti. El principio de la menor sorpresa significa el principio de 'mi' menor sorpresa. Y significa el principio de la menor sorpresa después de que aprendes bien Ruby. Por ejemplo, yo fui un programador de C++ antes de empezar a diseñar Ruby. Yo programé solamente en C++ durante dos o tres años. Y después de dos años de programar en C++, todavía me sorprendía.[1].

1.1.2. Características de Ruby

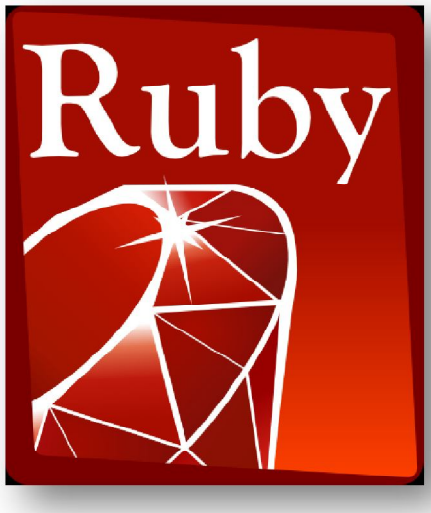


Figura 1.1 Logo de la Versión de Ruby 1.9.1(http://es.wikipedia.org/wiki/Ruby)

Lenguaje de guiones interpretado:

- Posibilidad de realizar directamente llamadas al sistema operativo
- Potentes operaciones sobre cadenas de caracteres y expresiones regulares
- Retroalimentación inmediata durante el proceso de desarrollo

[1] Referencia tomada de la pagina http://es.wikipedia.org/wiki/Ruby

Rápido y sencillo:

- Son innecesarias las declaraciones de variables
- Las variables no tienen tipo
- La sintaxis es simple y consistente
- La gestión de la memoria es automática

Programación orientada a objetos:

- Todo es un objeto
- Clases, herencia, métodos, ...
- Métodos singleton
- Mixins por módulos
- Iteradores y cierres

También:

- Enteros de precisión múltiple
- Modelo de procesamiento de excepciones
- Carga dinámica
- Hilos

Ventajas de Ruby

- Simple: fácil de aprender y mantener
- Poderoso
- "Language stays out of your way"
- Equipado con excelentes librerías
- Desarrollo rápido
- Código abierto
- "Divertido"

Desventajas de Ruby

- Rendimiento comparable a Perl o Python, pero lejos de C o C++

- Podemos extender Ruby con estos lenguajes
- No existen muchas frameworks desarrolladas en Ruby
- Ruby on Rails (http://www.rubyonrails.com/) es la excepción
- No existe una framework de GUI multi-plataforma ampliamente
- aceptada
- RAA – Ruby Application Archive (http://raa.ruby-lang.org/)
- No tan grande como CPAN – Comprehensive Perl Archive Network
- (http://www.cpan.org/)

1.1.3. Estudio de Ruby

Ruby on Rails, también conocido como RoR o Rails es un framework de aplicaciones web de código abierto escrito en el lenguaje de programación Ruby, siguiendo el paradigma de la arquitectura Modelo Vista Controlador (MVC). Trata de combinar la simplicidad con la posibilidad de desarrollar aplicaciones del mundo real escribiendo menos código que con otros frameworks y con un mínimo de configuración. El lenguaje de programación Ruby permite la metaprogramación, de la cual Rails hace uso, lo que resulta en una sintaxis que muchos de sus usuarios encuentran muy legible. Rails se distribuye a través de RubyGems, que es el formato oficial de paquete y canal de distribución de bibliotecas y aplicaciones Ruby.

Filosofía

Los principios fundamentales de Ruby on Rails incluyen No te repitas (del inglés Don't repeat yourself, DRY) y Convención sobre configuración.

No te repitas significa que las definiciones deberían hacerse una sola vez. Dado que Ruby on Rails es un framework de pila completa, los componentes están integrados de manera que no hace falta establecer puentes entre ellos. Por

ejemplo, en ActiveRecord, las definiciones de las clases no necesitan especificar los nombres de las columnas; Ruby puede averiguarlos a partir de la propia base de datos, de forma que definirlos tanto en el código como en el programa sería redundante.

Así, cuando se diseña una aplicación partiendo de cero sin una base de datos preexistente, el seguir las convenciones de Rails significa usar menos código (aunque el comportamiento puede ser configurado si el sistema debe ser compatible con un sistema heredado anterior)

Arquitectura MVC de Rails

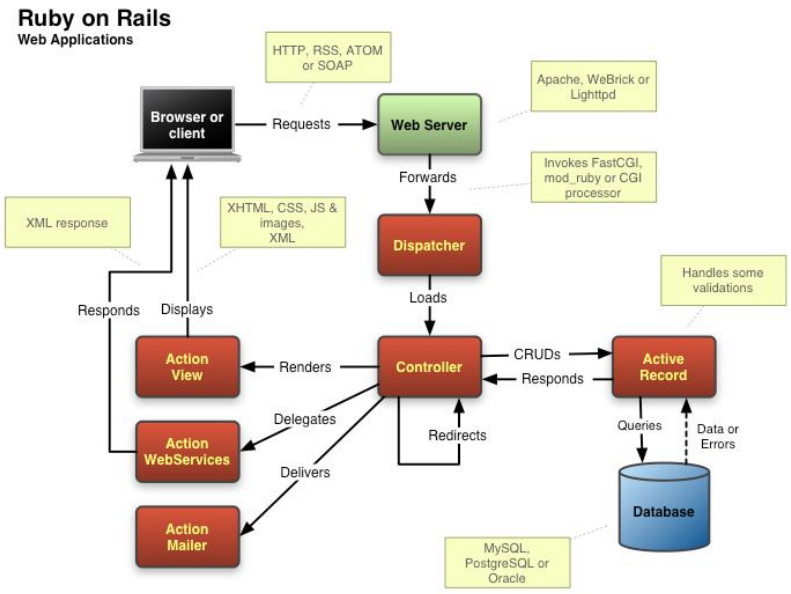


Figura 1.2 Mapa de Arquitectura de Ruby on Rails.(http://ftp.gul.uc3m.es/pub/gul/cursos/2005/noviembre/ruby_on_rails_ajax_web2.0_gul_uc3m/)

Las piezas de la arquitectura Modelo Vista Controlador en Ruby on Rails son las siguientes:

Modelo

En las aplicaciones web orientadas a objetos sobre bases de datos, el Modelo consiste en las clases que representan a las tablas de la base de datos.

En Ruby on Rails, las clases del Modelo son gestionadas por ActiveRecord. Por lo general, lo único que tiene que hacer el programador es heredar de la clase ActiveRecord::Base, y el programa averiguará automáticamente qué tabla usar y qué columnas tiene.

Las definiciones de las clases también detallan las relaciones entre clases con sentencias de mapeo objeto relacional. Por ejemplo, si la clase Imagen tiene una definición has_many: comentarios, y existe una instancia de Imagen llamada a, entonces a.comentarios devolverá un array con todos los objetos Comentario cuya columna imagen_id (en la tabla comentarios) sea igual a a.id.

Las rutinas de validación de datos (p.e. validates_uniqueness_of:checksum) y las rutinas relacionadas con la actualización (p.e. after_destroy:borrar_archivo, before_update:actualizar_detalles) también se especifican e implementan en la clase del modelo.

El modelo representa:

Las Tablas de la Base de Datos.
Migraciones (Expresan Cambios en las BD)
Observadores

Vista

En MVC, Vista es la lógica de visualización, o cómo se muestran los datos de las clases del Controlador. Con frecuencia en las aplicaciones web la vista consiste en una cantidad mínima de código incluido en HTML.

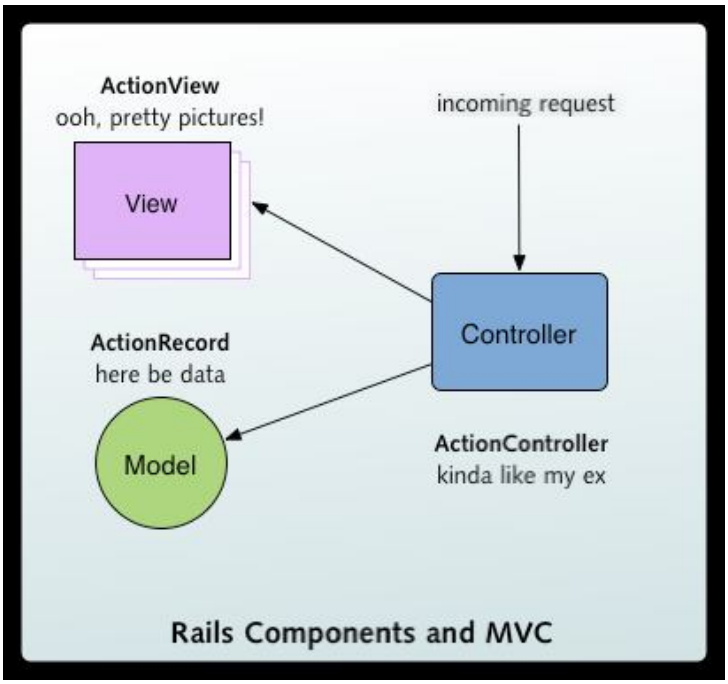


Figura 1.3 Componente de Rails y MVC
(http://ftp.gul.uc3m.es/pub/gul/cursos/2005/noviembre/ruby_on_rails_ajax_web2.0_gul_uc3m/)

Existen en la actualidad muchas maneras de gestionar las vistas. El método que se emplea en Rails por defecto es usar Ruby Embebido (archivos.rhtml, desde la versión 2.x en adelante de RoR archivos.html.erb), que son básicamente fragmentos de código HTML con algo de código en Ruby, siguiendo una sintaxis similar a JSP. También pueden construirse vistas en HTML y XML con Builder o usando el sistema de plantillas Liquid.

Es necesario escribir un pequeño fragmento de código en HTML para cada método del controlador que necesita mostrar información al usuario. El "maquetado" o distribución de los elementos de la página se describe separadamente de la acción del controlador y los fragmentos pueden invocarse unos a otros.

Controlador

En MVC, las clases del Controlador responden a la interacción del usuario e invocan a la lógica de la aplicación, que a su vez manipula los datos de las clases del Modelo y muestra los resultados usando las Vistas. En las aplicaciones web basadas en MVC, los métodos del controlador son invocados por el usuario usando el navegador web.

La implementación del Controlador es manejada por el ActionPack de Rails, que contiene la clase ApplicationController. Una aplicación Rails simplemente hereda de esta clase y define las acciones necesarias como métodos, que pueden ser invocados desde la web, por lo general en la forma http://aplicacion/ejemplo/metodo, que invoca a EjemploController#método, y presenta los datos usando el archivo de plantilla /app/views/ejemplo/método.html.erb, a no ser que el método redirija a algún otro lugar.

Rails también proporciona andamiaje, que puede construir rápidamente la mayor parte de la lógica y vistas necesarias para realizar las operaciones más frecuentes.

Otros módulos

Además, Rails ofrece otros módulos, como Action Mailer (para enviar correo electrónico) o Active Resource que proporciona la infraestructura necesaria para crear de manera sencilla recursos REST, algo por lo que apuesta claramente Rails en sus últimas versiones desplazando así a otros modelos como SOAP y XML-RPC a los que se les daba soporte en versiones anteriores mediante Action Web Service.

Ajax on Rails

Ajax es una técnica que permite usar Javascript y XML para procesar peticiones de un navegador web a un servidor web como procesamiento en segundo plano sin cargar otras páginas web adicionales. Rails proporciona diferentes facilidades que hacen más fácil implementar aplicaciones Ajax.

Rails incluye el framework de Javascript Prototype (una serie de herramientas que proporcionan llamadas Ajax y otra funcionalidad habitual en las tareas cliente-servidor) y script.aculo.us, una biblioteca en Javascript con mejoras en la interfaz de usuario (controles avanzados en los formularios, efectos visuales, arrastrar y soltar, etc.).

Gemas

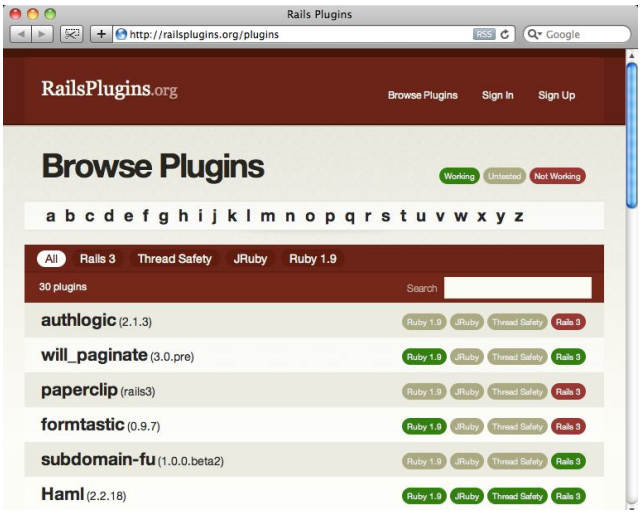


Figura 1.4 Ejemplos de gemas en Ruby on Rails (http://railsplugins.org/plugins)

Las gemas son plugins y/o códigos añadidos a nuestros proyectos Ruby on Rails, que nos permiten nuevas funcionalidades como nuevos create, nuevas funciones pre-escritas (como login de usuarios) o nuevas herramientas para el desarrollo como puedan ser Haml y SASS (la primera es una nueva forma de

template basada en html pero más sencilla y potente, y la segunda es igual pero para el caso de las CSS). Para encontrar el listado de gemas disponibles puedes ir a RubyForge.

Soporte de servidores Web

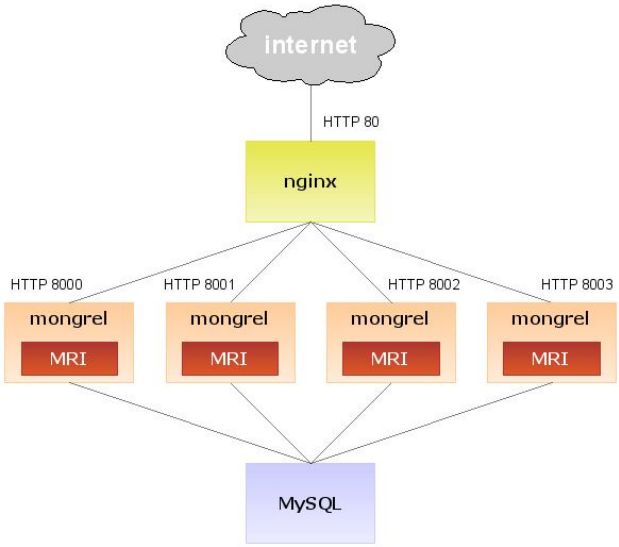


Figura 1.5 Mongrel es una de las arquitecturas más utilizadas para poner en producción una aplicación Rails.

(http://www.xdocs400.com/spip.php?article381&lang=fr**)**

Para desarrollo y pruebas, se utiliza Mongrel o WEBrick, incluido con Ruby. Para utilizar Rails en servidores en producción se está extendiendo el uso de Passenger, una suerte de mod_rails para Apache desarrollado en 2008 por la empresa holandesa Phusion. Otras opciones para producción son Nginx, Mongrel, Apache, Lighttpd con FastCGI o alguna combinación de ambos(por ejemplo utilizando Apache como proxy para los procesos Mongrel). Sobre Apache, mod ruby puede mejorar considerablemente el rendimiento, aunque su

uso no se recomienda porque no es seguro utilizar múltiples aplicaciones RoR sobre Apache. [2]

Soporte de Bases de Datos

Dada que la arquitectura Rails favorece el uso de bases de datos se recomienda usar un SGBDR para almacenamiento de datos.

Los servidores de bases de datos son Sistemas de Gestión de Bases de Datos o SGBD (en inglés DBMS = DataBase Management System), aplicaciones cuyo objetivo no es solo almacenar y recuperar datos, sino también facilitar la manipulación de éstos de la forma más eficiente y segura.

Un servidor de datos debe elaborar la información que va a devolver al cliente a partir de los datos recuperados del sistema de archivos usando para ellos diferentes estructuras físicas, así como asegurar su integridad verificando restricciones y utilizando transacciones.

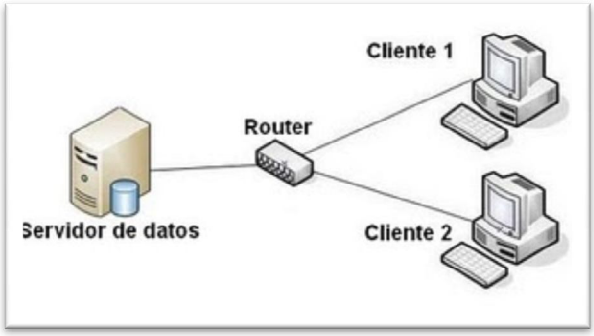


Figura 1.6 Servidor de datos Cliente / Servidor(http://1.bp.blogspot.com/_xOVSsUQX57M/StevpPU3HdI/AAAAAAAAACk/eEkkMzey0_Q/s1600-h/cliente+Servidor.JPG)

[2] Referencia tomada de la pagina http://1.bp.blogspot.com/_xOVSsUQX57M/StevpPU3HdI/AAAAAAAAACk/eEkkMzey0_Q/s1600-h/cliente+Servidor.JPG

Aunque en un principio surgieron SGBD de distintos tipos, según la estructura con la que se almacenaba la información, los más extendidos y conocidos son los Sistemas Administradores de Bases de Datos Relacionales SGBDR (en inglés RDBMS = Relational Data Base Management System) llamados así porque los datos se estructuran con ciertas relaciones entre ellos (Modelo Relacional), simplificando la recuperación y el tratamiento y evitando la repetición innecesaria de información. Algunos ejemplos de SGBDR son los mencionados SQL Server, Oracle, MySQL, Firebird, etc.

Durante años, las bases de datos se han utilizado en sistemas que se ajustaban a una arquitectura conocida como cliente/servidor. En ella los datos residen en un ordenador que actúa como servidor, ejecutando el software que denominábamos antes servidor de datos (SGBD). Los usuarios, desde ordenadores remotos, se sirven de un software cliente para comunicarse con el servidor de datos. Ese software cliente es especifico para cada servidor de datos existen. [3]

Rails soporta la biblioteca SQLite por defecto. El acceso a la base de datos es totalmente abstracto desde el punto de vista del programador, es decir que es agnóstico a la base de datos, y Rails gestiona los accesos a la base de datos automáticamente (aunque, si se necesita, se pueden hacer consultas directas en SQL) Rails intenta mantener la neutralidad con respecto a la base de datos, la portatibilidad de la aplicación a diferentes sistemas de base de datos y la reutilización de bases de datos preexistentes. Sin embargo, debido a la diferente naturaleza y prestaciones de los SGBDRs el framework no puede garantizar la compatibilidad completa. Se soportan diferentes SGBDRs, incluyendo MySQL, PostgreSQL, SQLite, IBM DB2 y Oracle.

[3] Referencia tomada de la pagina Web http://www.dokshor.com/las-mejores-herramientas-php-y-ruby-on-rails-desarrollo-web-profesional

Requisitos

- Servidor web como Apache 1.3.x or 2.x, lighttpd, algún servidor web compatible con FastCGI con un módulo similar a mod_rewrite, o Nginx. Para desarrollo, Rails permite utilizar Mongrel(un servidor HTTP ligero creado para soportar aplicaciones en Ruby y muy extendido entre aplicaciones en producción) o WEBrick(un pequeño servidor a medida de rendimiento limitado y no recomendado para su uso en producción). Rails soporta la extensión mod ruby de Apache (servidor web).
- Base de datos (por ejemplo, MySQL, PostgreSQL, o SQLite).

1.2. Herramientas de Desarrollo de Ruby on Rails

1.2.1 Introducción

Hay muchas alternativas para trabajar con Ruby on Rails, tanto libres y gratuitas como de pago. A continuación se listan las principales:

- Aptana: Multiplataforma. Nació como plugins de eclipse para la edición y desarrollo web. Actualmente puedes instalarlo como plugins o autónomo de forma independiente. Las últimas versiones están muy bien integradas con Ruby on Rails. En este momento Aptana 3 está en una versión BETA
- Netbeans: Uno de los más usados, libre y totalmente gratuito. Viene muy bien integrado con JRuby (lo cual es algo lógico pues es un programa de Sun).
- TextMate: Sólo para Mac. Es el entorno más usado entre la comunidad Rails. Es pago pero su potencia y forma de trabajo favorece la producción y desarrollo con Ruby on Rails.

Existen otros muchos, sólo es necesario usar algún buscador para encontrar más alternativas. [4].

1.2.2 Definición

- **Aptana**

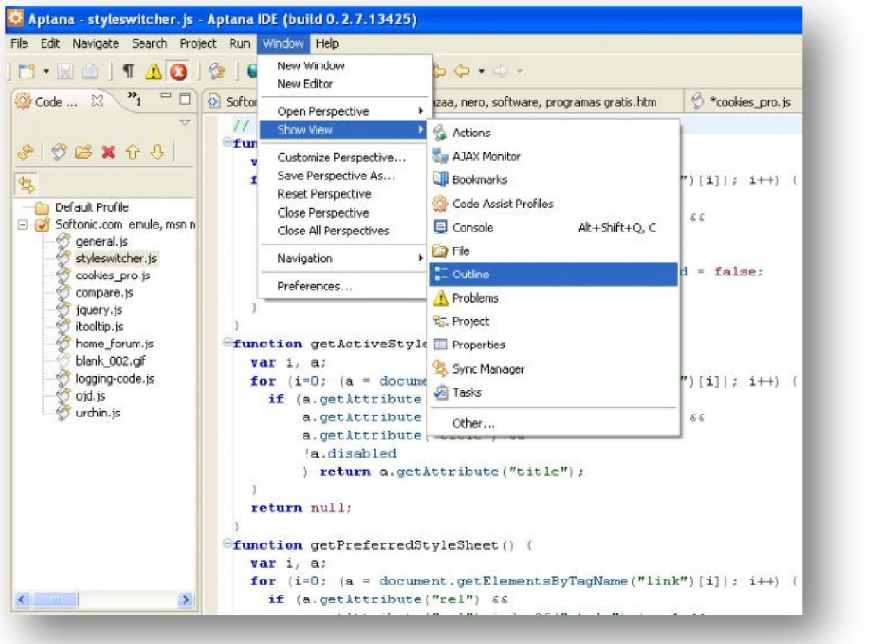


Figura 1.7 Imagen del software Aptana

(http://es.wikipedia.org/wiki/Ruby_on_Rails)

Aptana Studio es un entorno de desarrollo integrado gratuito basado en eclipse y desarrollado por Aptana, Inc., que puede funcionar bajo Windows, Mac y Linux y provee soporte para lenguajes como: Php, Python, Ruby, CSS, Ajax, HTML y Adobe AIR. Tiene la posibilidad de incluir complementos para nuevos lenguajes y funcionalidades.

[4] Referencia tomada de la pagina http://es.wikipedia.org/wiki/Ruby_on_Rails

Características

- Asistente de código para HTML y Javascript.
- Librerías ajax (jQuery, prototype, scriptaculous, Ext JS, dojo, YUI y Spry entre otras).
- Conexión vía FTP, SFTP, FTPS y Aptana Cloud.
- Herramientas para trabajo con base de datos.
- Marcado de sintáxis mediante colores.
- Compatible con extensiones para Eclipse (existen más de 1000). [5]

Eclipse

Eclipse es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado Java Development Toolkit (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse). Sin embargo, también se puede usar para otros tipos de aplicaciones cliente, como BitTorrent o Azureus.

Eclipse es también una comunidad de usuarios, extendiendo constantemente las áreas de aplicación cubiertas. Un ejemplo es el recientemente creado Eclipse Modeling Project, cubriendo casi todas las áreas de Model Driven Engineering.

Eclipse fue desarrollado originalmente por IBM como el sucesor de su familia de herramientas para VisualAge. Eclipse es ahora desarrollado por la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una

[5] Referencia tomada de la pagina http://www.aptana.com/

comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

Arquitectura

La base para Eclipse es la Plataforma de cliente enriquecido (del Inglés Rich Client Platform RCP). Los siguientes componentes constituyen la plataforma de cliente enriquecido:

- Plataforma principal - inicio de Eclipse, ejecución de plugins
- OSGi - una plataforma para bundling estándar.
- El Standard Widget Toolkit (SWT) - Un widget toolkit portable.
- JFace - manejo de archivos, manejo de texto, editores de texto
- El Workbench de Eclipse - vistas, editores, perspectivas, asistentes

Características

Eclipse dispone de un Editor de texto con resaltado de sintaxis. La compilación es en tiempo real. Tiene pruebas unitarias con JUnit, control de versiones con CVS, integración con Ant, asistentes (wizards) para creación de proyectos, clases, tests, etc., y refactorización.

Asimismo, a través de "plugins" libremente disponibles es posible añadir control de versiones con Subversion e integración con Hibernate. [6].

Netbeans

NetBeans es un entorno de desarrollo, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extender el NetBeans IDE. NetBeans IDE es un producto libre y gratuito sin restricciones de uso.

[6] Referencia tomada de la pagina http://es.wikipedia.org/wiki/Eclipse_(software)

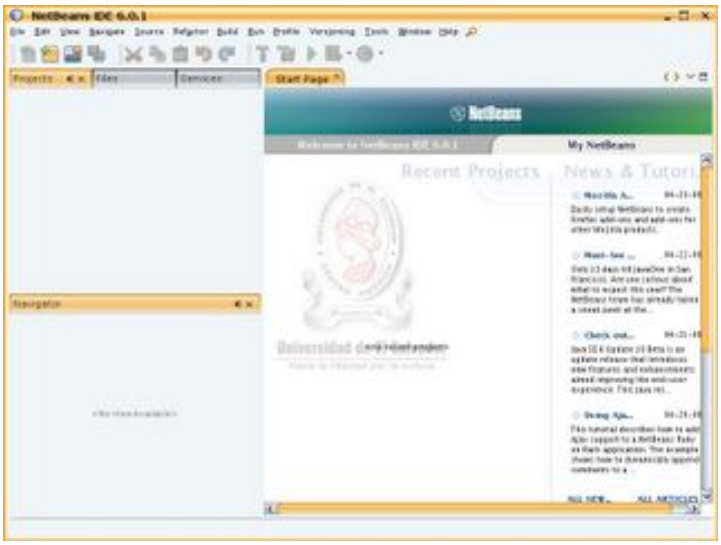


Figura 1.8 Pantalla de Netbeans 6.0 (http://es.wikipedia.org/wiki/NetBeans)

NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios en todo el mundo. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio de 2000 y continúa siendo el patrocinador principal de los proyectos.

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

NetBeans Hoy

Un proyecto de código abierto no es nada más ni nada menos que un proceso. Toma tiempo encontrar el equilibrio. El primer año, fue crucial como inicio. Los dos años siguientes, se orientó hacia código abierto. Como muestra de lo abierto que era, en los primeros dos años había más debate que implementación.

Con NetBeans 3.5 se mejoró enormemente en desempeño, y con la llegada de NetBeans 3.6, se reimplementó el sistema de ventanas y la hoja de propiedades, y se limpió enormemente la interfaz. NetBeans 4.0 fue un gran cambio en cuanto a la forma de funcionar del IDE, con nuevos sistemas de proyectos, con el cambio no solo de la experiencia de usuario, sino del reemplazo de muchas piezas de la infraestructura que había tenido NetBeans anteriormente. NetBeans IDE 5.0 introdujo un soporte mucho mejor para el desarrollo de nuevos módulos, el nuevo constructor intuitivo de interfaces Matisse, un nuevo y rediseñado soporte de CVS, soporte a Sun ApplicationServer 8.2, Weblogic9 y JBoss 4.

Con Netbeans 6.01 y 6.8 Se dio soporte a frameworks comerciales como son Struts, Hibernate.

La Plataforma NetBeans

Durante el desarrollo del NetBeans IDE ocurrió una cosa interesante. La gente empezó a construir aplicaciones usando el NetBeans core runtime con sus propios plug-ins, de hecho, esto se convirtió en un mercado bastante grande.

La Plataforma NetBeans es una base modular y extensible usada como una estructura de integración para crear aplicaciones de escritorio grandes. Empresas independientes asociadas, especializadas en desarrollo de software, proporcionan extensiones adicionales que se integran fácilmente en la

plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones.

La plataforma ofrece servicios comunes a las aplicaciones de escritorio, permitiéndole al desarrollador enfocarse en la lógica específica de su aplicación. Entre las características de la plataforma están:

- Administración de las interfaces de usuario (ej. menús y barras de herramientas)
- Administración de las configuraciones del usuario
- Administración del almacenamiento (guardando y cargando cualquier tipo de dato)
- Administración de ventanas
- Framework basado en asistentes (diálogos paso a paso)

NetBeans IDE

El IDE NetBeans es un IDE - una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java - pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el IDE NetBeans. El IDE NetBeans es un producto libre y gratuito sin restricciones de uso.

El NetBeans IDE es un IDE de código abierto escrito completamente en Java usando la plataforma NetBeans. El NetBeans IDE soporta el desarrollo de todos los tipos de aplicación Java (J2SE, web, EJB y aplicaciones móviles). Entre sus características se encuentra un sistema de proyectos basado en Ant, control de versiones y refactoring. [7].

- **Textmate.-**

[7] Referencia tomada de la pagina http://es.wikipedia.org/wiki/NetBeans

TextMate es un editor de textos con GUI para Mac OS X, creado por Allan Odgaard. Es fácilmente personalizable. Muchos de sus usuarios publican las personalizaciones que hacen. Al igual que vi o Emacs, sus usuarios en la mayoría son programadores.[1] También se puede usar para screenwriting, es decir, para hacer grabaciones de vídeo del proceso de escritura.[2]

Entre las características notables están fichas para abrir documentos, grabación de Macros, editor de folders y un sistema de paquetes extensible [8]

1.2.3 Fusión Aplicaciones

La fusión existente para el desarrollo de aplicaciones web La sintaxis de Ruby es similar a la de Perl o Python. La definición de clases y métodos está definida por palabras clave. Sin embargo, en Perl, las variables no llevan prefijos. Cuando se usa, un prefijo indica el ámbito de las variables. La mayor diferencia con C y Perl es que las palabras clave son usadas para definir bloques de código sin llaves. Los saltos de línea son significativos y son interpretados como el final de una sentencia; el punto y coma tiene el mismo uso. De forma diferente que Python, la indentación no es significativa.

Una de las diferencias entre Ruby y Python y Perl es que Ruby mantiene todas sus variables de instancia privadas dentro de las clases y solo la expone a través de métodos de acceso (attr_writer, attr_reader, etc). A diferencia de los métodos "getter" y "setter" de otros lenguajes como C++ o Java, los métodos de acceso en Ruby pueden ser escritos con una sola línea de código. Como la invocación de estos métodos no requiere el uso de paréntesis, es trivial cambiar una variable de instancia en una función sin tocar una sola línea de código o refactorizar dicho código. Los descriptores de propiedades de Python son similares pero tienen una desventaja en el proceso de desarrollo. Si uno

[8] Referencia tomada de la pagina http://es.wikipedia.org/wiki/NetBeans

comienza en Python usando una instancia de variable expuesta públicamente y después cambia la implementación para usar una instancia de variable privada expuesta a través de un descriptor de propiedades, el código interno de la clase necesitará ser ajustado para usar la variable privada en vez de la propiedad pública. Ruby elimina está decisión de diseño obligando a todas las variables de instancia a ser privadas, pero también proporciona una manera sencilla de declarar métodos set y get. Esto mantiene el principio de que en Ruby no se puede acceder a los miembros internos de una clase desde fuera de esta; en lugar de esto se pasa un mensaje (se invoca un método) a la clase y recibe una respuesta. [9]

En conclusión no hay compatibilidad entre lenguajes de programación ya que haciendo el análisis anterior vemos que Ruby es similar pero no se acopla a otros lenguajes de programación vemos aquí otras instancias con ortos lenguajes de programación web

Cuando aparece alguna novedad siempre hay tendencia a hacer comparaciones, cuando en muchos casos tampoco tiene mucho sentido hacerlas. En el caso de establecer una comparación entre PHP y RoR, se parte de un error de base, no son comparables, ¿podríamos hacer una comparación entre peras y manzanas? ¿Por qué se genera esta falsa comparación? PHP es un lenguaje de script, extremadamente liberal, no lo ata con ninguna tecnología extra y su forma de uso está más basado en la programación estructurada, influencia del lenguaje C. Sin embargo RoR es un entorno de programación basado en Ruby, siendo Ruby un lenguaje orientado a objetos 100%.

[9] Referencia tomada de la pagina http://www.ruby-forum.com/topic/144754

La mejora del modelo de OOP en PHP5 ha traído aparejado el surgimiento de un número importante de entornos de programación (Cake, PRADO, Symfony, etc.) que se

van agregando al lote del Zend Framework, aunque ninguno logra imponerse fuertemente sobre el resto. Amazon apuesta por Rails en su sitio unspun.amazon.org Por otro lado, el concepto de Rails como Framework MVC + ActiveRecord, ideal para el desarrollo ágil de sitios Web 2.0, seguramente no es la panacea universal ni la solución definitiva, ya que podemos encontrar otras posibles soluciones en otros lenguajes:

- Cake (PHP). Anteriormente citado,
- Django (Python),
- Trails (Java).

Estos funcionan tan bien o inclusive mejor que RoR, dependiendo del sistema con el que estamos lidiando. . [10]

1.2.4 Arquitectura de Desarrollo

La arquitectura MVC (Model – Control – View) en Rails

Rails es un framework de aplicaciones web Open Source (de Código Abierto) para trabajar con Ruby – lenguaje de programación orientado a objetos – ideado para programar de forma rápida y ágil.

La rapidez en el desarrollo de proyectos con Rails está fundamentada en la idea de construir la aplicación **separando** de forma clara las capas

[10] Referencia tomada de la pagina http://www.ruby-forum.com/topic/144754

de **Modelo** (Datos), **Vista** (Presentación) y **Controlador**(Funciones, métodos …) para reducir el acoplamiento entre la lógica de negocios y la de presentación. De modo que, antes de empezar a trabajar con Rails, no está de más tener claro los conceptos que engloba la **arquitectura Modelo Vista Controlador (MVC)**.

Los desarrolladores web utilizan una variante ligeramente distinta de MVC llamada "Model2". Model2 utiliza los mismos principios que MVC pero los ajusta a aplicaciones Web sin estado. En estas aplicaciones, un navegador llama a un controlador a través de estándares web. El controlador interactúa con el Modelo para obtener los datos y pone los objetos de dominio disponibles para su visualización en pantalla. A continuación el controlador invoca al generador de vista basándose en los resultados de validación de los datos extraídos generando una página web para el usuario.

1.3. Postgresql y la fusión con Ruby on Rails

1.3.1. Introducción

Para el estudio de la tecnología Ruby on Rails se analiza su implementación con base de datos siendo necesario explicar brevemente sobre su concepto, características y ventajas.

Por tanto una base de datos es un "almacén" que nos permite guardar grandes cantidades de información de forma organizada para que luego podamos encontrar y utilizar fácilmente. A continuación te presentamos una guía que te explicará el concepto y características de las bases de datos.

El término de bases de datos fue escuchado por primera vez en 1963, en un simposio celebrado en California, USA. Una base de datos se puede definir

como un conjunto de información relacionada que se encuentra agrupada ó estructurada.

Desde el punto de vista informático, la base de datos es un sistema formado por un conjunto de datos almacenados en discos que permiten el acceso directo a ellos y un conjunto de programas que manipulen ese conjunto de datos.

Cada base de datos se compone de una o más tablas que guarda un conjunto de datos. Cada tabla tiene una o más columnas y filas. Las columnas guardan una parte de la información sobre cada elemento que queramos guardar en la tabla, cada fila de la tabla conforma un registro.

Definición de base de datos

Se define una base de datos como una serie de datos organizados y relacionados entre sí, los cuales son recolectados y explotados por los sistemas de información de una empresa o negocio en particular.

Características

Entre las principales características de los sistemas de base de datos podemos mencionar:

- Independencia lógica y física de los datos.
- Redundancia mínima.
- Acceso concurrente por parte de múltiples usuarios.
- Integridad de los datos.
- Consultas complejas optimizadas.
- Seguridad de acceso y auditoría.
- Respaldo y recuperación.
- Acceso a través de lenguajes de programación estándar.

Sistema de Gestión de Base de Datos (SGBD)

Los Sistemas de Gestión de Base de Datos (en inglés DataBase Management System) son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta.

Ventajas de las bases de datos

- Control sobre la redundancia de datos
- Consistencia de datos
- Compartición de datos
- Mantenimiento de estándares
- Mejora en la integridad de datos
- Mejora en la seguridad
- Mejora en la accesibilidad a los datos
- Mejora en la productividad
- Mejora en el mantenimiento
- Aumento de la concurrencia
- Mejora en los servicios de copias de seguridad
- Desventajas de las bases de datos
- Complejidad
- Coste del equipamiento adicional
- Vulnerable a los fallos:

Tipos de Base de Datos

Entre los diferentes tipos de base de datos, podemos encontrar los siguientes:

- **MySql:** es una base de datos con licencia GPL basada en un servidor. Se caracteriza por su rapidez. No es recomendable usar para grandes volúmenes de datos.
- **PostgreSql y Oracle:** Son sistemas de base de datos poderosos. Administra muy bien grandes cantidades de datos, y suelen ser utilizadas en intranets y sistemas de gran calibre.
- Access: Es una base de datos desarrollada por Microsoft. Esta base de datos, debe ser creada bajo el programa access, el cual crea un archivo .mdb con la estructura ya explicada.
- **Microsoft SQL Server:** es una base de datos más potente que access desarrollada por Microsoft. Se utiliza para manejar grandes volúmenes de informaciones.[11].

1.3.2. La Base de Datos Postgresql

Figura 1.9 Logotipo de Postgresql(http://es.wikipedia.org/wiki/PostgreSQL)

PostgreSQL está ampliamente considerado como el sistema de bases de datos de código abierto más avanzado del mundo. Posee muchas características que tradicionalmente sólo se podían ver en productos comerciales de alto calibre. El curso de PostgreSQL, SQL y PHP se aborda a continuación del de MySQL y

[11] Referencia tomada de la pagina http://es.wikipedia.org/wiki/PostgreSQL

PHP y como en ese caso, también trabajaremos en tiempo real sobre una PostgreSQL desarrollando en una versión amplia y avanzada, el lenguaje SQL, comprendiendo el soporte de funciones de la base en estudio.

¿Qué es PostgreSQL?

PostgreSQL es un Sistema de Gestión de Bases de Datos Objeto-Relacionales (ORDBMS) que ha sido desarrollado de varias formas desde 1977. Comenzó como un proyecto denominado Ingres en la Universidad Berkeley de California. Ingres fue más tarde desarrollado comercialmente por la Relational Technologies/Ingres Corporation.

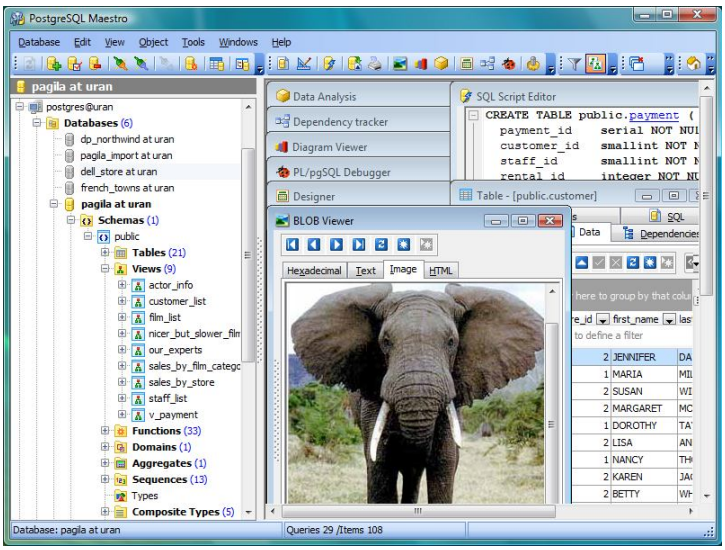


Figura 1.10 Interfaz de Postgress (http://www.postgresql.org.es/)

En 1986 otro equipo dirigido por Michael Stonebraker de Berkeley continuó el desarrollo del código de Ingres para crear un sistema de bases de datos objeto-relacionales llamado Postgres. En 1996, debido a un nuevo esfuerzo de código abierto y a la incrementada funcionalidad del software, Postgres fue

renombrado a PostgreSQL, tras un breve periplo como Postgres95. El proyecto PostgreSQL sigue actualmente un activo proceso de desarrollo a nivel mundial gracias a un equipo de desarrolladores y contribuidores de código abierto.

PostgreSQL está ampliamente considerado como el sistema de bases de datos de código abierto más avanzado del mundo. Posee muchas características que tradicionalmente sólo se podían ver en productos comerciales de alto calibre.

PostgreSQL y Java. PostgreSQL y PHP

PostgreSQL forma parte de la capacitación que incluye a SQL avanzado, trabajando en esa ocasión, junto a MySQL.

El objetivo básico, luego de la capacitación SQL y PostgreSQL, es la interacción de la base de datos con PHP y Java. Este entrenamiento implica el desarrollo de aplicaciones web, por un lado, las JSP (Java Server Pages) y Postgres proponen el desarrollo de aplicaciones seguras, para un ámbito cada vez menos seguro; la misma consideración es oportuna para la simbiosis Postgres y PHP.

Estar capacitado para desarrollar aplicaciones web con Postgres, Java, JSP, PHP genera posibilidades profesionales de excepción.

Características de PostgreSQL

PostgreSQL está considerado como la base de datos de código abierto más avanzada del mundo. PostgreSQL proporciona un gran número de características que normalmente sólo se encontraban en las bases de datos comerciales tales como DB2 u Oracle. La siguiente es una breve lista de algunas de esas características, a partir de PostgreSQL 7.1.x.

DBMS Objeto-Relacional

PostgreSQL aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas. Ejemplos de su avanzada funcionalidad son consultas SQL declarativas, control de concurrencia multi-versión, soporte multi-usuario, transactions, optimización de consultas, herencia, y arrays.

Altamente_Extensible

PostgreSQL soporta operadores, funcionaes métodos de acceso y tipos de datos definidos por el usuario.

Soporte_SQL_Comprensivo

PostgreSQL soporta la especificación SQL99 e incluye características avanzadas tales como las uniones (joins) SQL92.

Integridad Referencial

PostgreSQL soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos.

API Flexible

La flexibilidad del API de PostgreSQL ha permitido a los vendedores proporcionar soporte al desarrollo fácilmente para el RDBMS PostgreSQL. Estas interfacaes incluyen Object Pascal, Python, Perl, PHP, ODBC, Java/JDBC, Ruby, TCL, C/C++, y Pike.

Lenguajes Procedurales

PostgreSQL tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL. Este lenguaje es comparable al

lenguaje procedural de Oracle, PL/SQL. Otra ventaja de PostgreSQL es su habilidad para usar Perl, Python, o TCL como lenguaje procedural embebido.

Cliente/Servidor

PostgreSQL usa una arquitectura proceso-por-usuario cliente/servidor. Esta es similar al método del Apache 1.3.x para manejar procesos. Hay un proceso maestro que se ramifica para proporcionar conexiones adicionales para cada cliente que intente conectar a PostgreSQL.[12].

1.3.3. Fusion de ruby on rails con postgresql

En lo referente a la fusión de las base de datos por defecto para cualquier aplicación o desarrolladores web se aplican las bases de datos que vienen por defecto en este caso son las de Msql el propio SQL En este caso la fusión de Ruby on Rails con Postgresql relativamente fue exitosa ya que Postgresql presenta características acoplable y es considerada una base de datos de código abierto más avanzada del mundo que proporciona grandes características que normalmente solo se encuentran en bases de datos comerciales como DB2 y Oracle.

Análisis de la tecnología Ruby on Rails y la base de datos Postgresql

Realizado el estudio de Ruby on Rails y la base de datos Postgresql se a realizar un análisis del porque no se elige otra plataforma para desarrollo de la aplicación en primer lugar las otras plataformas ya se encuentran estudiadas y el objetivo como viene hacer el caso es el estudio de nuevas tecnologías para desarrollo web2.0, en este caso la las características que se presentan en el lenguaje que son las siguientes:

- La posibilidad de hacer llamada directamente al sistema operativo

[12] Referencia tomada de la pagina http://www.postgresql.org.es/

- Muy potente para el manejo de cadenas y expresiones regulares
- No Hay declaraciones de variables
- Gestión de memoria automáticamente
- Todo es un objeto

El Objetivo de utilizar una base de datos ya no común como es Msql nos lleva hacer un análisis de poder utilizar base de datos más robustas y en mi caso elegí Postgresql para hacer más interesante el trabajo por el cual me encuentro satisfecho el haber utilizado dicha base de datos.

CAPITULO II

Herramientas compatibles con Ruby on Rails

2.1 Descripción de herramientas afines

2.2 Compatibilidad y soporte

2. Herramientas compatibles con Ruby on Rails

2.1. Descripción de Herramientas afines

Introducción

Hay muchas alternativas para trabajar con Ruby on Rails, tanto libres y gratuitas como de pago. A continuación se listan las principales:

- Aptana: Multiplataforma. Nació como plugins de eclipse para la edición y desarrollo web. Actualmente puedes instalarlo como plugins o autónomo de forma independiente. Las últimas versiones están muy bien integradas con Ruby on Rails. En este momento Aptana 3 está en una versión BETA
- Netbeans: Uno de los más usados, libre y totalmente gratuito. Viene muy bien integrado con JRuby (lo cual es algo lógico pues es un programa de Sun).
- TextMate: Sólo para Mac. Es el entorno más usado entre la comunidad Rails. Es pago pero su potencia y forma de trabajo favorece la producción y desarrollo con Ruby on Rails.

Existen otros muchos, sólo es necesario usar algún buscador para encontrar más alternativas. [13].

[13] Referencia tomada de la pagina http://railsplugins.org/plugins, http://www.rubyist.net/~slagell/ruby/index.html

Netbeans

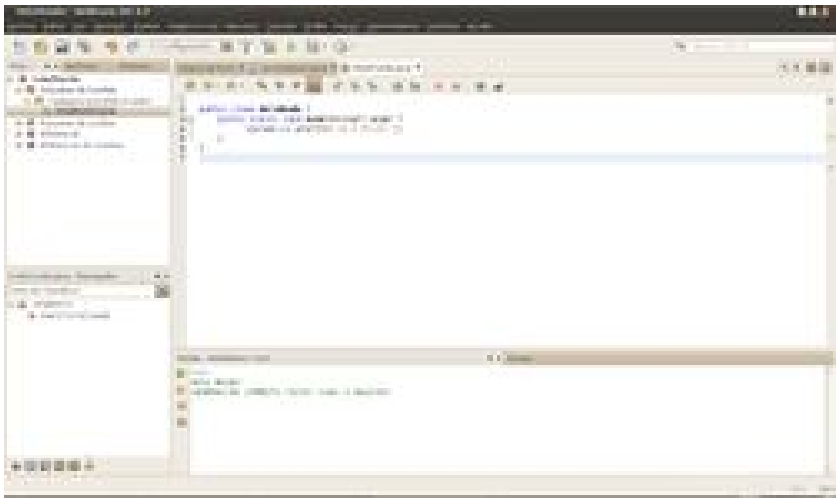


Figura 2.1 Imagen de Netbeans 6.9(http://es.wikipedia.org/wiki/NetBeans)

NetBeans es un entorno de desarrollo, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extender el NetBeans IDE. NetBeans IDE es un producto libre y gratuito sin restricciones de uso.

NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios en todo el mundo. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio de 2000 y continúa siendo el patrocinador principal de los proyectos.

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma

NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

Historia

NetBeans comenzó como un proyecto estudiantil en Republica Checa (originalmente llamado Xelfi), en 1996 bajo la tutoría de la Facultad de Matemáticas y Física en la Universidad Carolina en Praga. La meta era escribir un entorno de desarrollo integrado (IDE) para Java parecida a la de Delphi. Xelfi fue el primer entorno de desarrollo integrado escrito en Java, con su primer pre-release en 1997-.

Xelfi fue un proyecto divertido para trabajar, ya que las IDEs escritas en Java eran un territorio desconocido en esa época. El proyecto atrajo suficiente interés, por lo que los estudiantes, después de graduarse, decidieron que lo podían convertir en un proyecto comercial. Prestando espacios web de amigos y familiares, formaron una compañía alrededor de esto. Casi todos ellos siguen trabajando en NetBeans.

Tiempo después, fueron contactados por Roman Stanek, un empresario que ya había estado relacionado con varias iniciativas en la Republica Checa. Estaba buscando una buena idea en la que invertir, y encontró en Xelfi una buena oportunidad. Así, tras una reunión, el negocio surgió.

El plan original era desarrollar unos componentes JavaBeans para redes. Jarda Tulach, quien diseñó la arquitectura básica de la IDE, propuso la idea de llamarlo NetBeans, a fin de describir este propósito. Cuando las especificaciones de los Enterprise JavaBeans salieron, decidieron trabajar con este estándar, ya que no tenía sentido competir contra él, sin embargo permaneció el nombre de NetBeans.

En la primavera de 1999, Netbeans DeveloperX2 fue lanzado, soportando Swing. Las mejoras de rendimiento que llegaron con el JDK 1.3, lanzado en otoño de 1999, hicieron de NetBeans una alternativa realmente viable para el desarrollo de herramientas. En el verano de 1999, el equipo trabajó duro para rediseñar DeveloperX2 en un NetBeans más modular, lo que lo convirtió en la base de NetBeans hoy en día.

Algo más ocurrió en el verano de 1999. Sun Microsystems quería una herramienta mejor de desarrollo en Java, y comenzó a estar interesado en NetBeans. En otoño de 1999, con la nueva generación de NetBeans en Beta, se llegaría a un acuerdo.

Sun adquirió otra compañía de herramientas al mismo tiempo, Forté, y decidió renombrar NetBeans a Forté for Java. El nombre de NetBeans desapareció por un tiempo.

Seis meses después, se tomó la decisión de hacer a NetBeans open source. Mientras que Sun había contribuido considerablemente con líneas de código en varios proyectos de código abierto a través de los años, NetBeans se convirtió en el primer proyecto de código abierto patrocinado por ellos. En Junio del 2000 NetBeans.org fue lanzado.

NetBeans Hoy

Un proyecto de código abierto no es nada más ni nada menos que un proceso. Toma tiempo encontrar el equilibrio. El primer año, fue crucial como inicio. Los dos años siguientes, se orientó hacia código abierto. Como muestra de lo abierto que era, en los primeros dos años había más debate que implementación.

Con NetBeans 3.5 se mejoró enormemente en desempeño, y con la llegada de NetBeans 3.6, se reimplementó el sistema de ventanas y la hoja de propiedades,

y se limpió enormemente la interfaz. NetBeans 4.0 fue un gran cambio en cuanto a la forma de funcionar del IDE, con nuevos sistemas de proyectos, con el cambio no solo de la experiencia de usuario, sino del reemplazo de muchas piezas de la infraestructura que había tenido NetBeans anteriormente. NetBeans IDE 5.0 introdujo un soporte mucho mejor para el desarrollo de nuevos módulos, el nuevo constructor intuitivo de interfaces Matisse, un nuevo y rediseñado soporte de CVS, soporte a Sun ApplicationServer 8.2, Weblogic9 y JBoss 4.

Con Netbeans 6.01 y 6.8 Se dio soporte a frameworks comerciales como son Struts, Hibernate.

La Plataforma NetBeans

Durante el desarrollo del NetBeans IDE ocurrió una cosa interesante. La gente empezó a construir aplicaciones usando el NetBeans core runtime con sus propios plug-ins, de hecho, esto se convirtió en un mercado bastante grande.

La Plataforma NetBeans es una base modular y extensible usada como una estructura de integración para crear aplicaciones de escritorio grandes. Empresas independientes asociadas, especializadas en desarrollo de software, proporcionan extensiones adicionales que se integran fácilmente en la plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones.

La plataforma ofrece servicios comunes a las aplicaciones de escritorio, permitiéndole al desarrollador enfocarse en la lógica específica de su aplicación. Entre las características de la plataforma están:

- Administración de las interfaces de usuario (ej. menús y barras de herramientas)

- Administración de las configuraciones del usuario
- Administración del almacenamiento (guardando y cargando cualquier tipo de dato)
- Administración de ventanas
- Framework basado en asistentes (diálogos paso a paso)

NetBeans IDE

El IDE NetBeans es un IDE - una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java - pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el IDE NetBeans. El IDE NetBeans es un producto libre y gratuito sin restricciones de uso.

El NetBeans IDE es un IDE de código abierto escrito completamente en Java usando la plataforma NetBeans. El NetBeans IDE soporta el desarrollo de todos los tipos de aplicación Java (J2SE, web, EJB y aplicaciones móviles). Entre sus características se encuentra un sistema de proyectos basado en Ant, control de versiones y refactoring.

NetBeans IDE 6.5, la cual fue lanzada el 19 de noviembre de 2008, extiende las características existentes del Java EE (incluyendo Soporte a Persistencia, EJB 3 y JAX-WS). Adicionalmente, el NetBeans Enterprise Pack soporta el desarrollo de Aplicaciones empresariales con Java EE 5, incluyendo herramientas de desarrollo visuales de SOA, herramientas de esquemas XML, orientación a web servicies (for BPEL), y modelado UML. El NetBeans C/C++ Pack soporta proyectos de C/C++, mientras el PHP Pack, soporta PHP 5.

Modularidad. Todas las funciones del IDE son provistas por módulos. Cada módulo provee una función bien definida, tales como el soporte de Java, edición, o soporte para el sistema de control de versiones. NetBeans contiene

todos los módulos necesarios para el desarrollo de aplicaciones Java en una sola descarga, permitiéndole al usuario comenzar a trabajar inmediatamente.

Sun Studio, Sun Java Studio Enterprise, y Sun Java Studio Creator de Sun Microsystems han sido todos basados en el IDE NetBeans.

Desde julio de 2006, NetBeans IDE es licenciado bajo la Common Development and Distribution License (CDDL), una licencia basada en la Mozilla Public License (MPL). [14].

TextMate

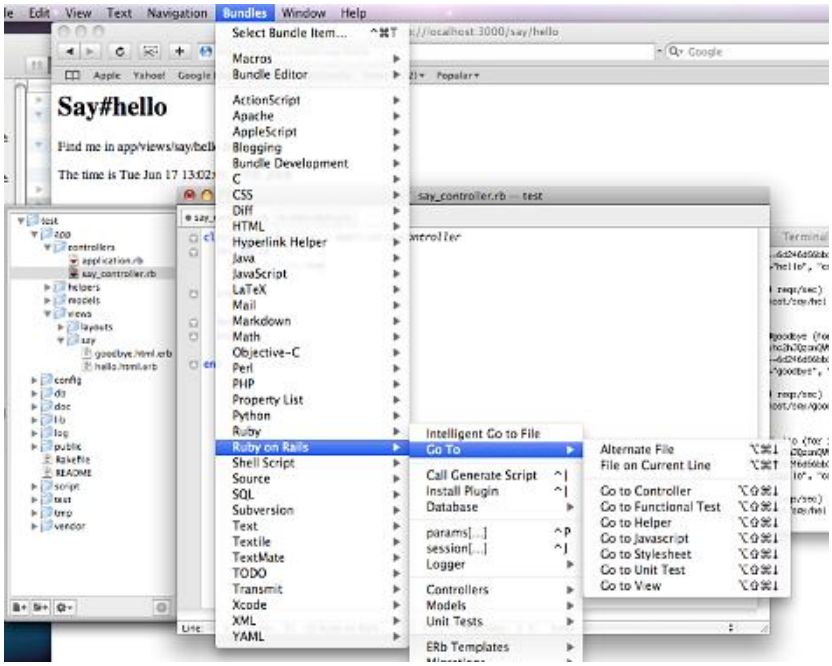


Figura 2.2 Imagen del Editor TextMate(http://es.wikipedia.org/wiki/TextMate)

[14] Referencia tomada de la pagina http://es.wikipedia.org/wiki/NetBeans

TextMate (TM) es todavía el mejor editor de textos para programadores disponible para **Mac OS X**. Celebrado por traer buena parte de la potencia encontrada en editores mucho más difícil de dominar, como Vim, e integrarlo con la muy accesible interface para el usuario del sistema operativo de Apple. Al mismo tiempo, TextMate introdujo una *extensibilidad* muy poco común en la forma de "Bundles" que fué rápidamente adoptada y explotada por su comunidad de usuarios, inspirando a otros proyectos similares, como el editor gEdit de GNOME.

Increíblemente, **TextMate** fué creado y aún sigue siendo desarrollado por una sola persona: Allan Odgaard, que además viene siendo víctima del éxito de su creación desde hace años, al ser incapaz de terminar su muy anticipada **versión 2.0** (TM2). Trágicamente, la esperada actualización de un producto tan exitoso, amado e imitado puede convertirse en un caso más de vaporware, lo que no en poca medida debido a su propia naturaleza cerrada y propietaria.

Afortunadamente, Odgaard está comenzando a ver la luz y por lo menos ya considera que hay una verdadera la posibilidad de salvar a TextMate de la irrelevancia abriendo su desarrollo. [15].

[15] Referencia tomada de la pagina http://textmate.softonic.com/mac

Aptana

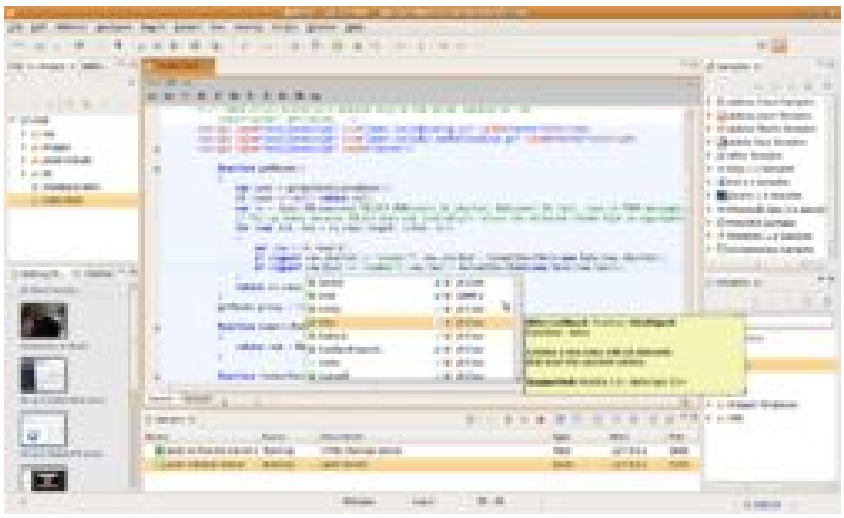


Figura 2.3 Imagen de Aptana (http://www.aptana.com/)

Aptana Studio es un entorno de desarrollo integrado gratuito basado en eclipse y desarrollado por Aptana, Inc., que puede funcionar bajo Windows, Mac y Linux y provee soporte para lenguajes como: Php, Python, Ruby, CSS, Ajax, HTML y Adobe AIR. Tiene la posibilidad de incluir complementos para nuevos lenguajes y funcionalidades.

Características

- Asistente de código para HTML y Javascript.
- Librerías ajax (jQuery, prototype, scriptaculous, Ext JS, dojo, YUI y Spry entre otras).
- Conexion vía FTP, SFTP, FTPS y Aptana Cloud.
- Herramientas para trabajo con base de datos.
- Marcado de sintáxis mediante colores.
- Compatible con extensiones para Eclipse (existen más de 1000).

Un IDE para el desarrollo de aplicaciones web, enfocado en Ajax y la Web 2.0.

Aptana Studio es un IDE de desarrollo para aplicaciones de la web 2.0, gratuito, código libre, con soporte Ajax, PHP, Ruby on Rails, Adobe Air, iPhone, etc. Con Aptana se facilita en desarrollo integrado de Ajax con las tecnologías emergentes. El líneas generales, así es como se presentan ellos y la verdad es que tiene buena pinta.

Aptana está basado en el conocido entorno de desarrollo Eclipse (IDE = Integrated Development Environment), también Open Source. Pero mientras que Eclipse está focalizado en el desarrollo para Java, Aptana Studio es una distribución focalizada en el desarrollo web, con soporte a HTML, CSS y Javascript, así como opcionalmente a otras tecnologías mencionadas como PHP, Adobe Air o Ruby on Rails. Aptana Studio está disponible como una aplicación independiente o como plug-in para Eclipse. Nosotros estamos comentando aquí la aplicación independiente, que se puede descargar desde http://www.aptana.com

El programa se distribuye para todos los sistemas operativos más comunes: Windows, Linux y Mac OS X. En dos versiones:

- **Aptana Studio Community Edition**: Es la versión gratuita, que contiene la mayoría de las funcionalidades del IDE, como edición, debugging, sincronización y administración de proyectos. Con soporte para todas las tecnologías que veníamos comentando.
- **Aptana Studio Profesional Edition**: Esta otra versión, de pago, tiene además algunas funcionalidades extra, útiles aunque no necesarias para empezar a trabajar con Aptana. Por ejemplo, la versión "Pro" incluye: Soporte para JSON, un motor de reportes, debug en Internet Explorer (la

versión community sólo tiene debug en Firefox), gestión remota de proyectos y soporte para FTPS y SFTP.

De las funcionalidades de la versión Profesional posiblemente las más destacables o interesantes para justificar su compra sea la gestión remota de proyectos, pero el propio Aptana Community dispone de una herramienta para conexión por FTP para hacer upload o download de archivos, así como la sincronización de nuestros contenidos en local con los del servidor remoto. También si queremos hacer una conexión segura por FTP con los protocolos FTPS y SFTP será necesaria la compra de la versión Pro. El debug en Internet Explorer también resulta interesante, como no, pero si tenemos la oportunidad de hacerlo en Firefox puede ser suficiente para evaluar si nos interesa la compra antes de hacerla.

La descarga del software ocupa unos 85 Mb y se puede hacer desde la propia página web de Aptana. Pero ojo que cuando instalamos Aptana Studio obtenemos el entorno con soporte para HTML, CSS y Javascript. Luego podremos instalar plug-ins para disponer también del editor PHP, Ruby on Rails y otros como soporte para Adobe AIR y el desarrollo para iPhone.

Para instalar plug-ins tenemos el Plug-in Manager, en una de las ventanitas del entorno, situada en la parte de abajo, accesible por una pestaña. También se pueden instalar plug-ins desde la página de inicio de Aptana y desde el menú Help - Software updates - Find and Install.

En resumen, estas son algunas de las características principales de Aptana Studio:

- Ayudas visuales para la escritura de scripts en diversos lenguajes, como coloreado y auto escritura del código, ayudas contextuales de referencia a medida que se escribe, etc.

- Visualización de errores de sintaxis a medida que se escribe.
- Soporte para hacer FTP a servidores remotos, con herramientas para sincronización.
- Debug en Firefox (Debug Internet Explorer también con la versión Profesional)
- Librerías de funciones en Javascript populares en Ajax/Javascript para utilizar en los proyectos.
- Ejemplos ya creados para empezar a conocer las posibilidades de desarrollo rápidamente.
- Pre visualización de estilos CSS con el editor CSS.
- Extensible a partir de plug-ins que puede crear Aptana u otras empresas y herramientas para estar al tanto de cualquier nuevo añadido.
- Extensible por Javascript. Los usuarios pueden escribir scripts para realizar acciones y macros.
- Los Snippets permiten insertar fragmentos de texto que se utilizan muy a menudo.
- Y un largo etc...

Todo esto hace de Aptana un programa muy interesante para los desarrolladores del web, gratuito y basado en otro gran sistema Eclipse, lo que garantiza aun más el buen trabajo demostrado hasta ahora por el equipo de Aptana.

2.2. Compatibilidad y Soporte

Compatibilidad

Ruby on Rails es un lenguaje de programación que tiene compatibilidad con aptana la herramienta para la elaboración de la pagina web a construir.

En el estudio realizado se comprobó que aptana es la mejor herramienta para el desarrollo de la página web es así que la programación se la realizó en esta herramienta.

Soporte

En virtud del alojamiento de sitio web se hizo difícil el alojamiento así que se realizo un servidor web el cual aloja a la página web contratamos un apuntador a nuestro servidor web para poder salir hacia el mundo.

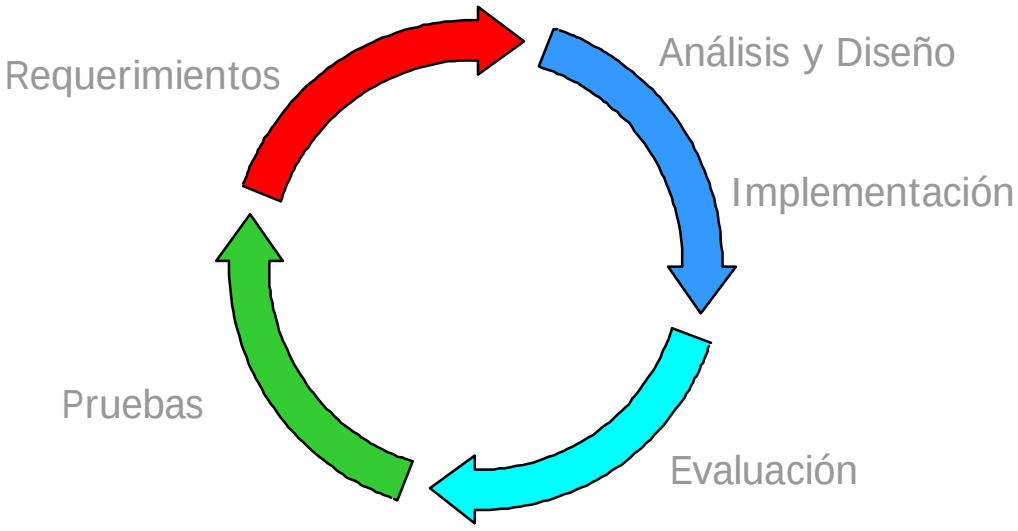
En el diario vivir de los programadores hace que para un soporte de estos lenguajes de programación no exista muchas personas que se interesen por Ruby on Rails mas se interesan por los lenguaje que se encuentran como comunes para todos los programadores como es el caso de PHP, Simphony, etc.

CAPITULO III

Análisis, Diseño e Implementación de "Portal Web 2.0 para la Mancomunidad de la Cuenca del Rio Mira"

3.1 Análisis de Requerimientos
3.2 Diseño
3.3 Desarrollo e Implementación
3.4 Producto
3.5 Pruebas Finales

DESARROLLO DE SOFTWARE
UTILIZANDO LA METODOLOGÍA RUP (RATIONAL UNIFIED PROCESS)



3. Análisis, Diseño e Implementación de "Portal Web 2.0 para la Mancomunidad de la Cuenca del Rio Mira"

3.1 ANALISIS DE REQUERIMIENTOS

INTRODUCCIÓN.

A continuación se describe una configuración de RUP para este proyecto. Por las características del proyecto, se han incluido artefactos, roles y actividades de la metodología, manteniendo los más esenciales. Dicha configuración está basada en la siguiente selección de artefactos:

DOCUMENTACIÓN TÉCNICA DEL PROYECTO CENTRO DE CAPACITACIÓN CONTINUA

ENTREGABLES

A continuación se presentan y describen cada uno de los entregables. Esta lista se basa en los documentos indicados por RUP, como proceso de desarrollo de software.

Cabe destacar que todos los entregables citados han sido objeto de modificaciones a lo largo del proceso de desarrollo, finalizando con una versión definitiva y completa de cada uno de ellos.

REQUISITOS

VISIÓN.-

Este documento define la visión del producto desde la perspectiva del cliente, especificando las necesidades y características del producto. Constituye una base de acuerdo a los requisitos del sistema.

3.1.1. Flujo de trabajo

El aplicativo está enfocado principalmente en la presentación de la información relevante de la Mancomunidad a través del portal web. La aplicación Web es aplicada a la automatización de varios procesos, tales como:

- Administración de los contenidos del portal.

El sitio Web cuenta con 2 módulos:

- Módulo de Administración de todo el sitio Web
- Módulo de cliente.

Descripción de las características del Sistema.-

Las características del sitio Web en el Módulo de Administración son:

- Ingreso, Modificación, Eliminación de los contenidos del Portal.

Las características del sitio Web en el Módulo del Usuario son:

- Mostrar al usuario de forma organizada la información contenida en el portal.

Descripción de las necesidades del MCRM.-

- Información de la mancomunidad.

Es un modelo de las funciones de negocio vistas desde la perspectiva de los actores externos (Agentes de registro, solicitantes finales, otros sistemas etc.). Permite situar al sistema en el contexto organizacional haciendo énfasis en los objetivos en este ámbito.

Para la representación de este modelo se utilizan diagramas de Actividad para modelar los Flujos de Trabajo (workflows) del área problema, y Hojas de Trabajo para analizar los requerimientos.

HOJA DE TRABAJO VALIDACIÓN DEL PROYECTO CON EL CLIENTE

Cliente: Mancomunidad Cuenca del Rio Mira
Proyecto: DESARROLLO DE UNA APLICACIÓN WEB PARA LA MCRM.
Tema a tratar: Análisis de requerimientos
Fecha: lunes 07 de mayo del 2010

Observaciones:
Se acordó que el proceso sería el siguiente en cuanto al MCRM: • Administración de contenidos • Ingreso, modificación y eliminación de contenidos.

Figura 3.1 Hoja de trabajo (Autoria)

3.1.2. Casos de Uso

Este diagrama permite definir los límites del sistema y las relaciones entre el sistema y el entorno.

En la figura 2 se puede observar el diagrama de casos de uso del Portal del MCRM:

En el siguiente diagrama de casos de uso, se describe claramente el funcionamiento de la aplicación Web, los actores son el Administrador y el Usuario.

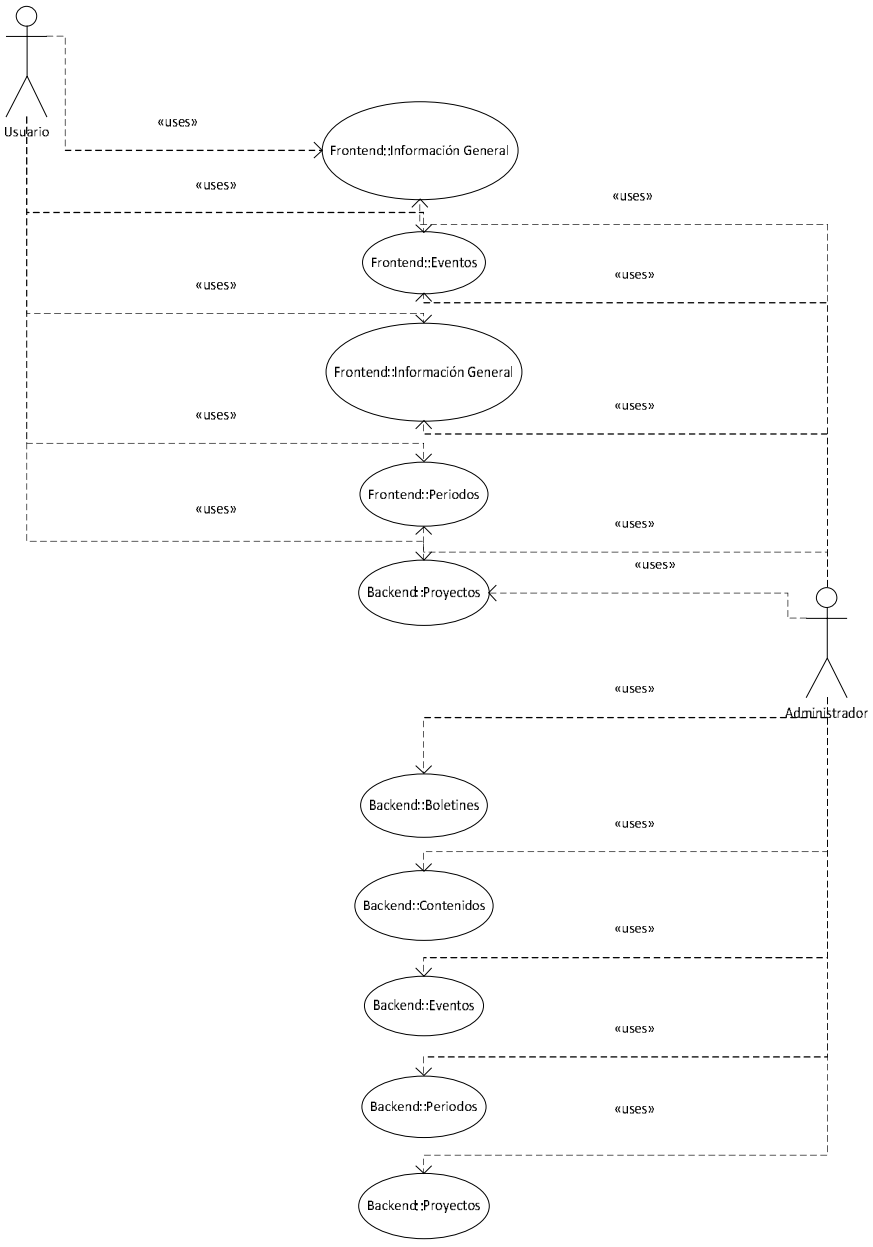


Figura 3.2 Diagrama de Casos de Uso (Autoria)

MODELO DE CASOS DE USO:

El modelo de Casos de Uso presenta la funcionalidad del sistema y los actores que hacen uso de ella. Se representa mediante Diagramas de Casos de Uso.

desde una representación en términos de análisis hacia una de diseño (incluyendo una orientación hacia el entorno de implementación), de acuerdo al avance del proyecto.

3.1.3. Diagrama de Clases

Un diagrama de clases presenta las clases del sistema con sus relaciones estructurales y de herencia. La definición de clase incluye definiciones para atributos y operaciones.

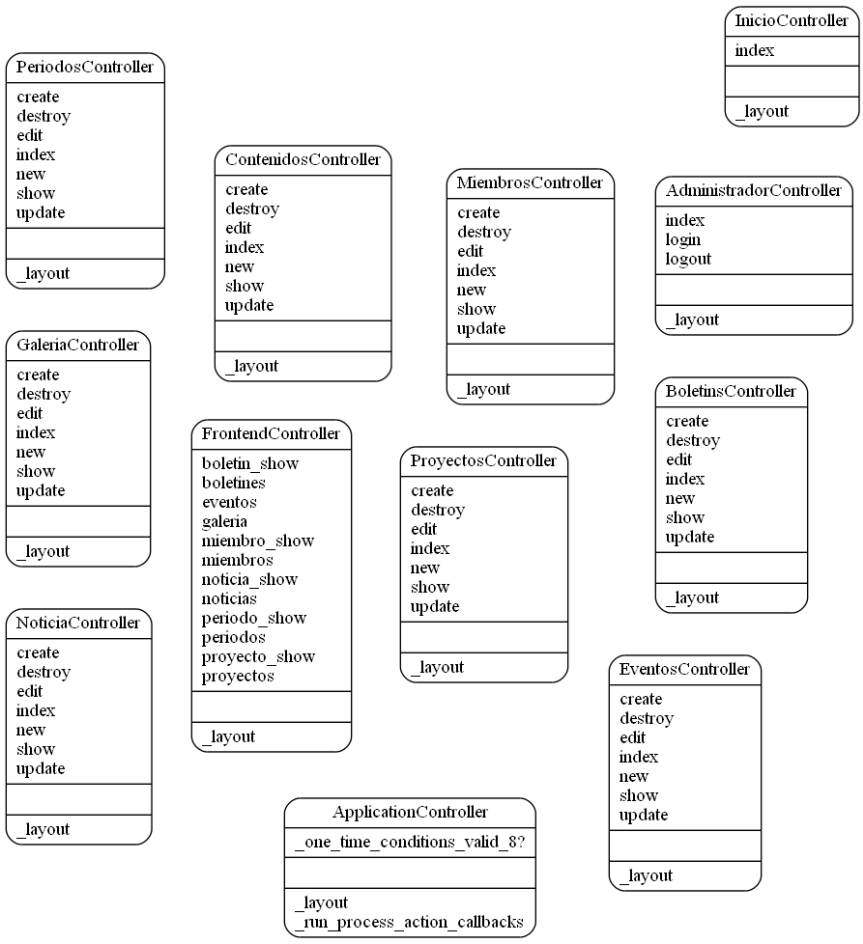


Figura 3.3 Base de datos (Autoria)

Diagrama de estados

El diagrama de estado captura pequeñas realidades.

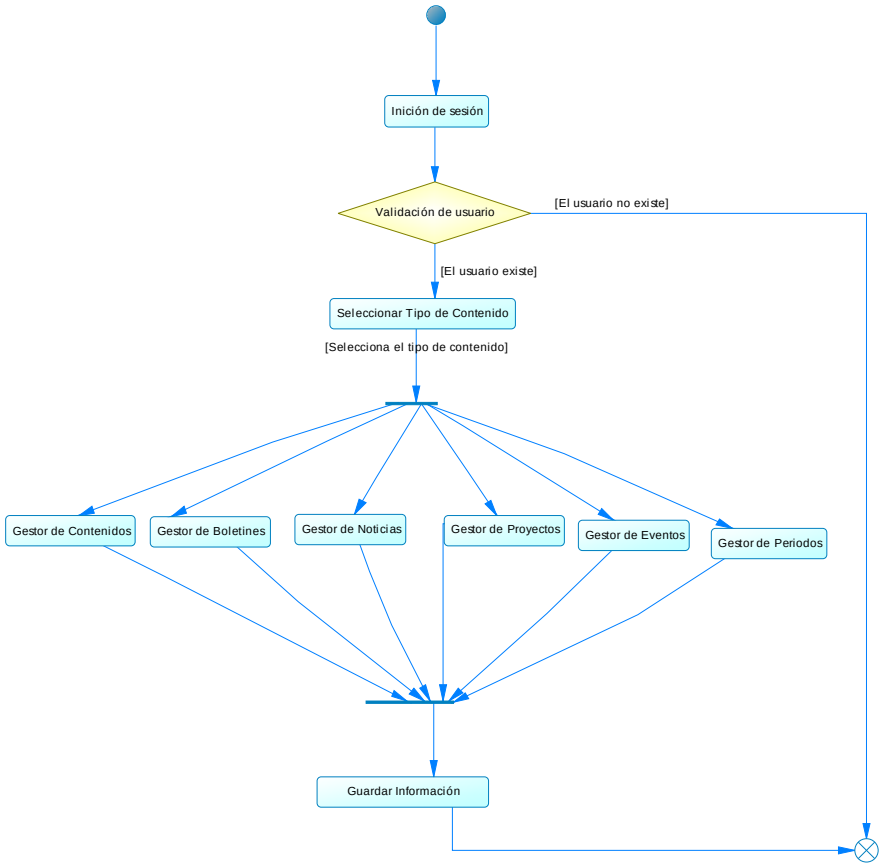


Figura 3.4 Diagrama de estados (Autoria)

Diagrama de Secuencias

Un diagrama de secuencia muestra las interacciones entre objetos ordenadas en secuencia temporal.

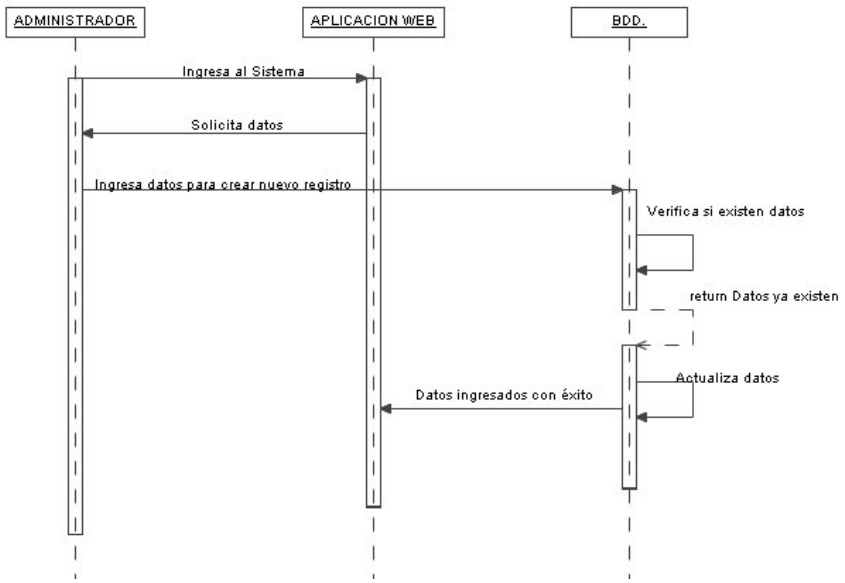


Figura 3.5 Diagrama de secuencia para ingresar datos (Autoria)

Diagrama de Actividades

Un diagrama de actividades ilustra la naturaleza dinámica de un sistema, mediante el modelado del flujo ocurrente de actividad en actividad

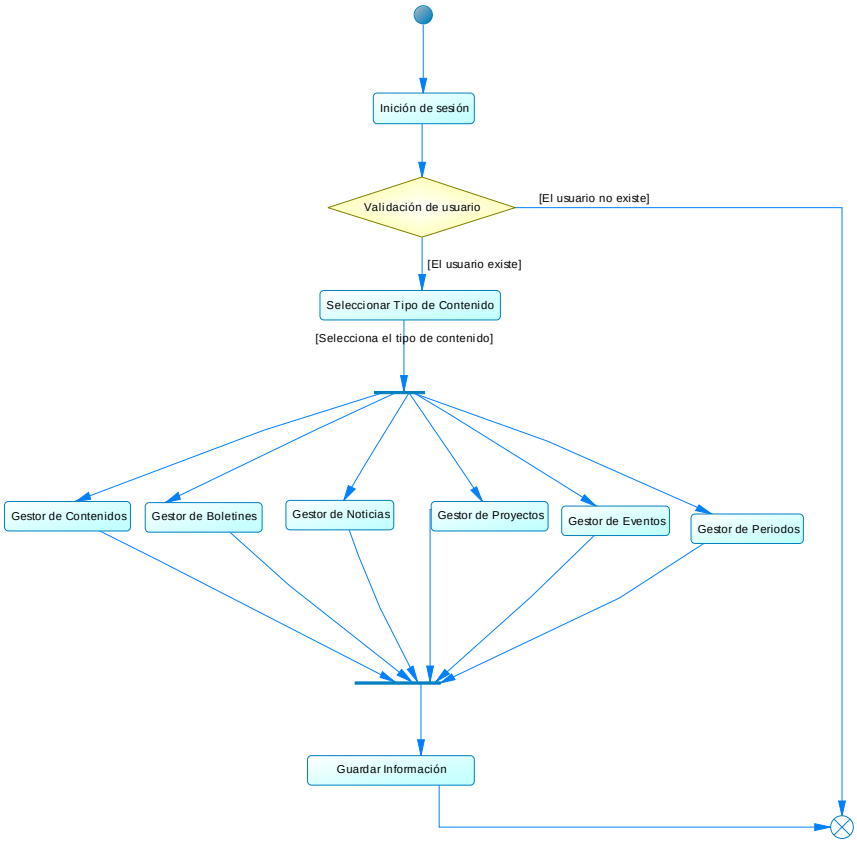


Figura 3.6 Diagrama de Actividades Administrador (Autoria)

3.2 DISEÑO

La finalidad de los diagramas es presentar diversas perspectivas de un sistema, a las cuales se les conoce como modelo.

El diseño se basa en la información adquirida con las personas que se encuentran trabajando en la institución; mancomunidad de la cuenca del rio mira, esto comprende en una administración para el levantamiento de toda la información.

Se utilizaron baria herramientas para el desarrollo visible de la aplicación en este caso tenemos programas de desarrollo de estilos de pantallas artister ayuda de photoshop e ilustrator, flash para el manejo de imágenes

3.3 DESARROLLO E IMPLEMENTACIÓN

3.3.1 Especificaciones Tecnológicas

Como se realizó con anterioridad todo el estudio de las herramientas para la implementación la aplicación se la realizó con la herramienta Aptana y el uso del lenguaje de programación Ruby y el Framework Rails ya que fue la mejor tecnología que se pudo aplicar para el desarrollo del aplicativo de la pagina web de la mancomunidad de la cuenca del rio mira

3.3.2 Arquitectura de Aplicación

La arquitectura para el desarrollo implementada es MVC. La rapidez en el desarrollo de proyectos con Rails está fundamentada en la idea de construir la aplicación **separando** de forma clara las capas de **Modelo** (Datos), **Vista** (Presentación) y **Controlador** (Funciones, métodos …) para reducir el acoplamiento entre la lógica de negocios y la de presentación. De modo que, antes de empezar a trabajar con Rails, no está de más tener claro los conceptos que engloba la **arquitectura Modelo Vista Controlador (MVC)**.

3.4 PRODUCTO

3.4.1 Instalación

1.) instalar rugby rubyinstaller-1.9.2-p290

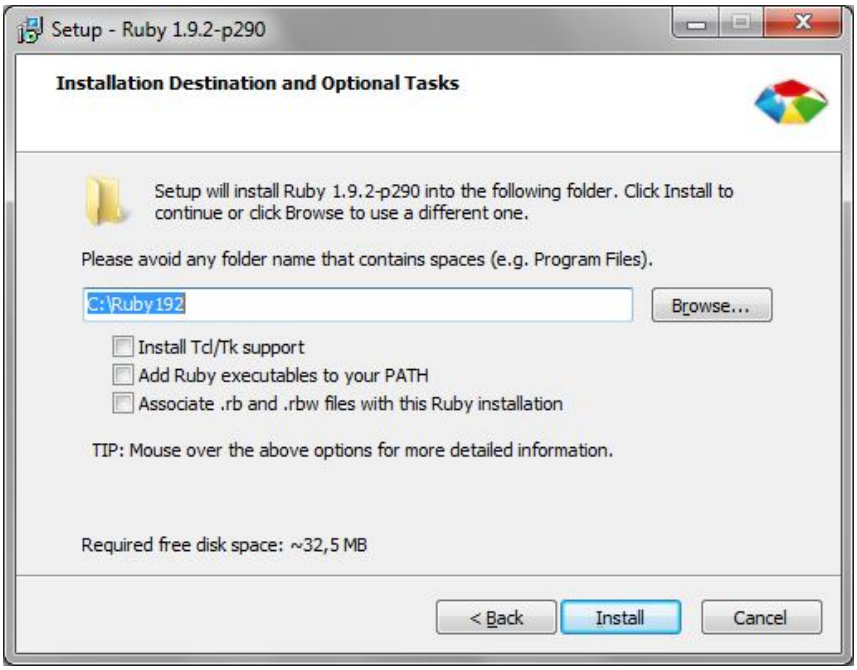


Figura 3.7 Instalacion de Ruby (Autoria)

2.) Desempaquetar devkit en el disco c

3.) Abrir una consola de comandos e ingresar al devkit ejecutar

ruby dk.rb init

ruby dk.rb install

4. Copiar la carpeta web_mcrm al disco c

Desde la consola entrar a la carpeta

Ejecutar: gem install rails

Ejecutar: bundle install

Ejecutar: gem update

Ejecutar: C:\web_mcrm\script\rails server

3.4.2 Descripción del Portal Web

3.4.2.1 Administrativa

En un browser ejecutar http://localhost:3000

Descripción del Sistema

Contenido

Lista todos los contenidos con las opciones de mostrar, editar y eliminar

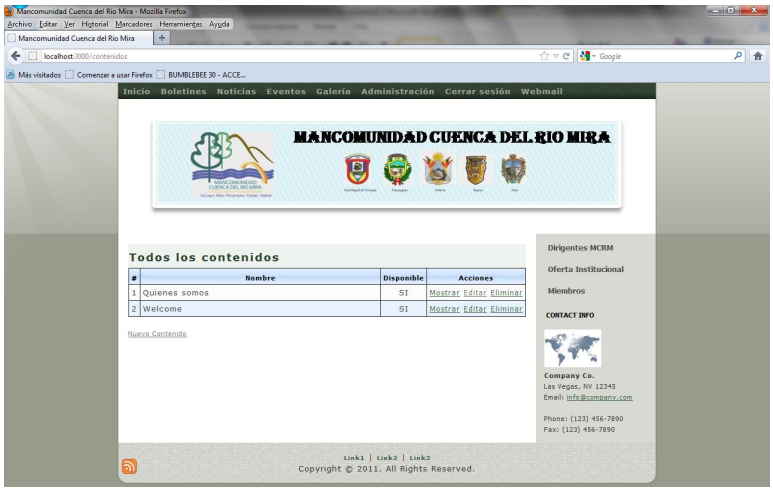


Figura 3.8 Ingreso contenidos (Autoria)

Editando contenido

Edita el contenido seleccionado y guardar para conservar cambios.

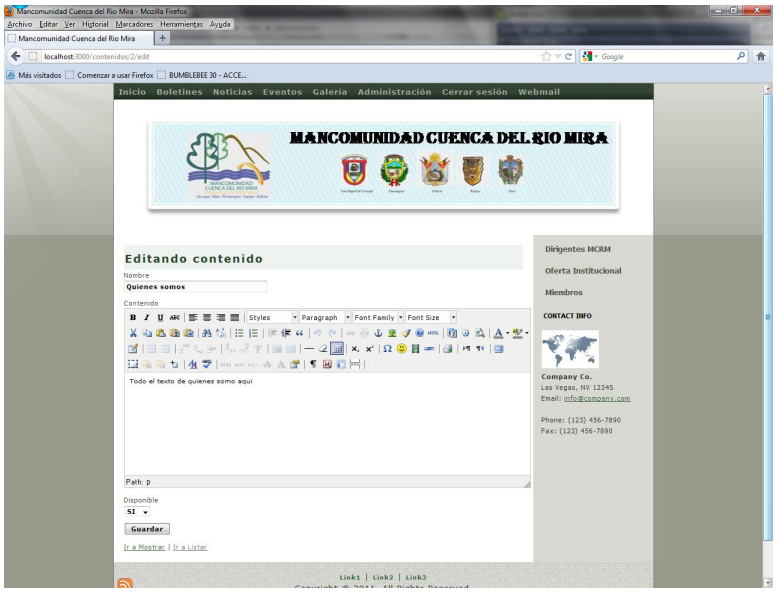


Figura 3.9 Editando contenidos (Autoria)

Boletines

Despliega la lista de boletines con las opciones de mostrar, editar y eliminar

Figura 3.10 Ingreso boletines (Autoria)

Editar Boletín

Edita el boletín seleccionado y luego opción guardar.

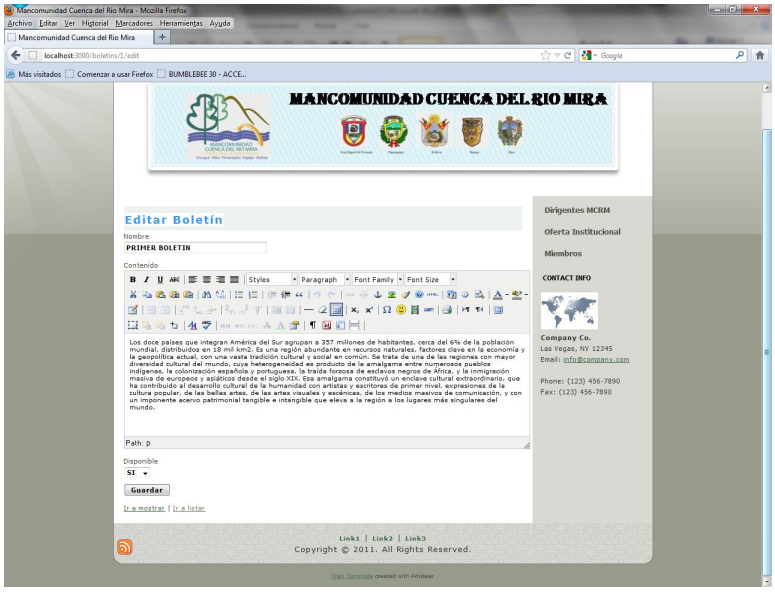


Figura 3.11 Editando boletines (Autoria)

Eventos

Despliega la lista de eventos con las opciones de mostrar, editar y eliminar.

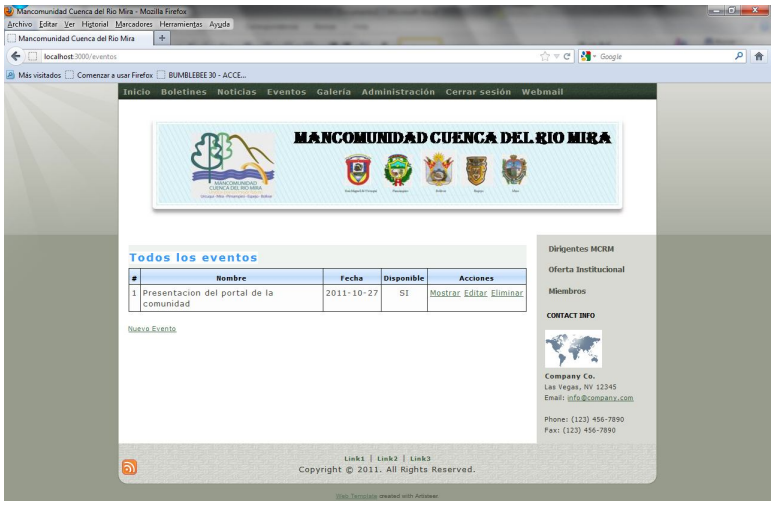


Figura 3.12 Ingreso eventos (Autoria)

Editar evento

Edita el evento seleccionado, luego la opción guardar.

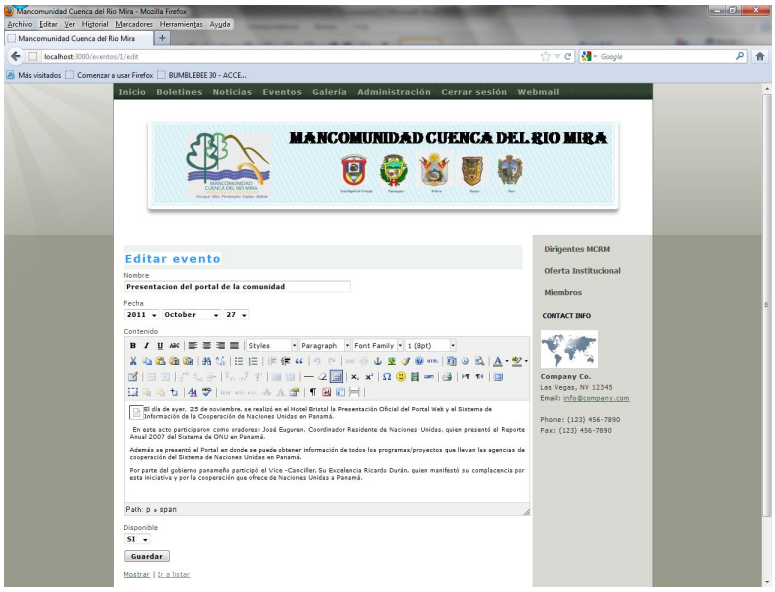


Figura 3.13 Editando eventos (Autoria)

Miembros

Despliega la lista de los miembros con las opciones de mostrar, editar y eliminar.

Figura 3.14 Ingreso miembros (Autoria)

Editando miembro

Edita el miembro seleccionado, luego la opción guardar.

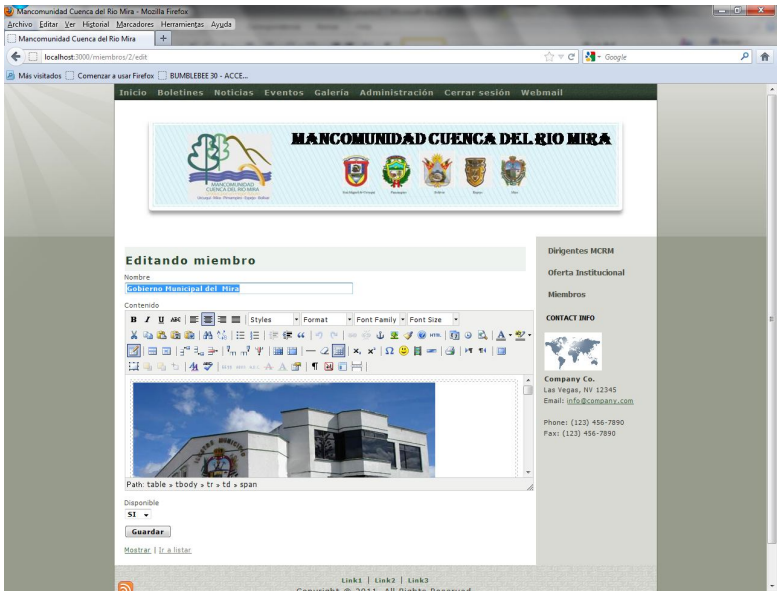


Figura 3.15 Editando miembros (Autoria)

Periodos

Despliega la información de todos los períodos con las opciones de mostrar, editar y eliminar.

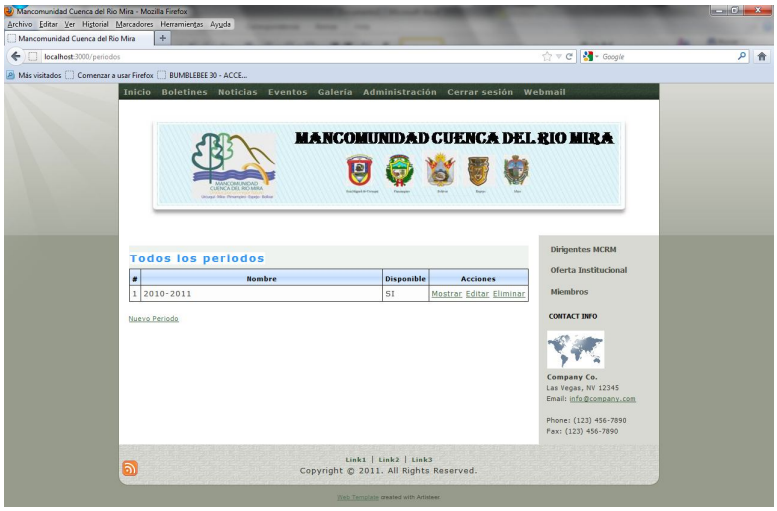


Figura 3.16 Ingreso periodos (Autoria)

Editando Periodo

Edita el período seleccionado, luego la opción guardar.

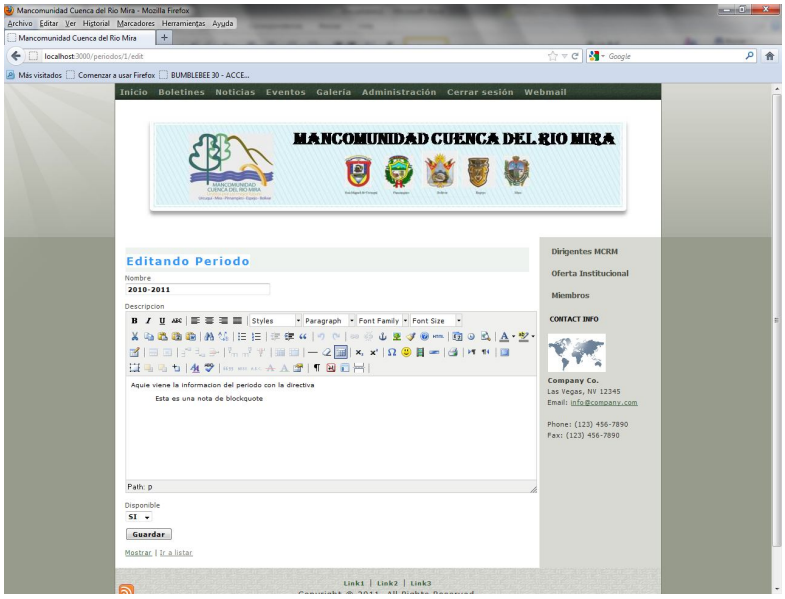


Figura 3.17 Editando periodos (Autoria)

Noticias

Lista todas las noticias con las opciones de mostrar, editar y eliminar.

Figura 3.18 Ingreso noticias (Autoria)

Editar Noticia

Edita la noticia seleccionada después de hacer los cambios la opción guardar.

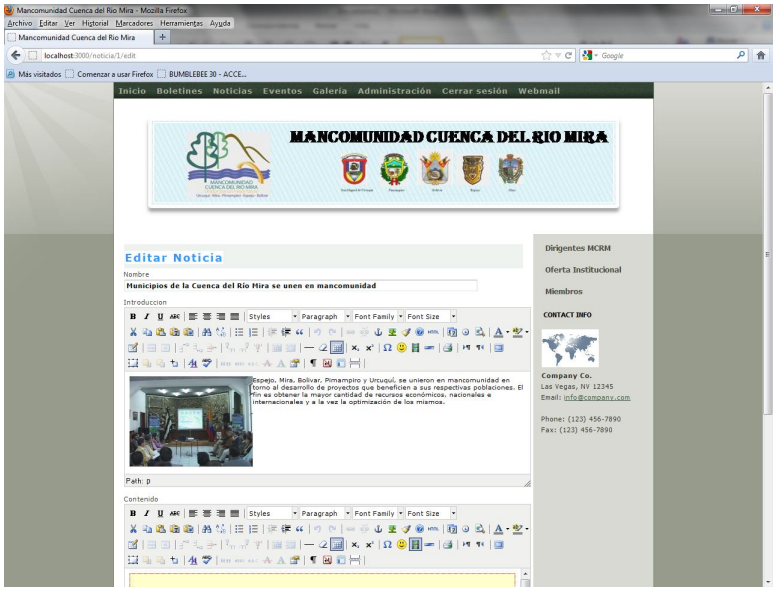


Figura 3.19 Editando noticias (Autoria)

Proyectos

Presenta una lista de todos los proyectos con las opciones de mostrar, editar y eliminar.

Figura 3.20 Ingreso proyecto (Autoria)

Editando proyecto

Edita el proyecto seleccionado con las herramientas de edición luego la opción guardar para conservar los cambios.

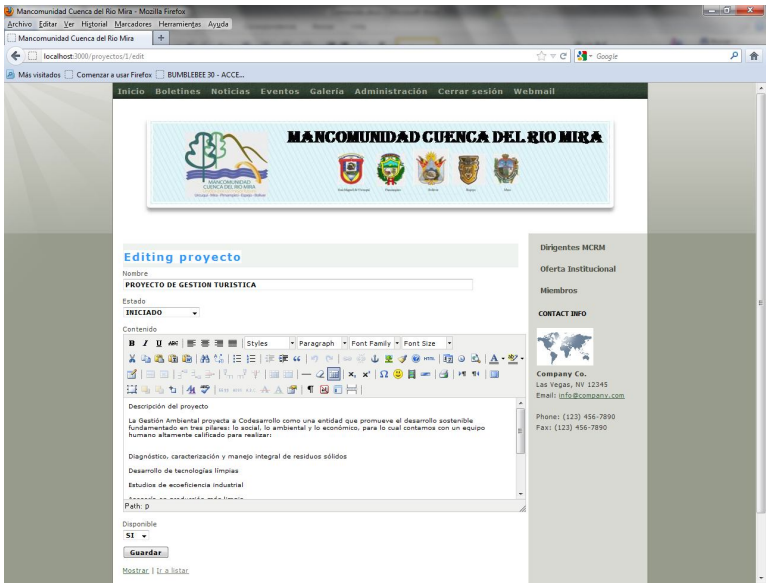


Figura 3.21 Editando proyecto (Autoria)

3.4.2.2 Evaluación La información ofrecida en el portal web de la mancomunidad de la cuenca del rio mira se encuentra evaluada y activa por parte de la dirigencia de dicho organismo para el uso y difusión de la información a todos sus miembros y a la comunidad de la web

3.5 Pruebas Finales

Realizada todas las pruebas pertinentes del caso el portal web se encuentra operativamente subida a la web para que toda la comunidad de la web pueda hacer uso de la difusión informativa que tiene el organismo de la mancomunidad de la cuenca del rio mira así accediendo de la siguiente manera www.mancomunidadcuencariomira.org

Figura 3.22 Web Mancomunidad cuenca del rio mira (Autoria)

CAPITULO IV

4. CONCLUSIONES Y RECOMENDACIONES

4. Conclusiones y Recomendaciones

4.1 Conclusiones

Como conclusiones son las siguientes:

- Se implemento exitosamente la pagina web de la mancomunidad de la cuenca del rio mira, usando el lenguaje de programación Ruby on Rails con la herramienta de desarrollo Aptana con el uso del patrón de diseño MVC que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos que son el MODELO que representa los datos, la VISTA se encarga de mostrar los datos que posee el modelo, pide los datos que va a necesitar y genera eventos a consecuencia del usuario y el CONTROLADOR recibe las llamadas de la vista y lo traduce en llamadas al modelo para obtener los datos y llamadas a la vista para mostrar los datos.
- La elaboración de distintos diagramas y artefactos siguiendo la metodología RUP proveen una fácil ejecución del proceso de elaboración de la pagina web
- En la parte de la subida a la web de la pagina tuvo un poco de dificultades ya que no existe un hostin que aloje el sitio web por lo que se realizo el servidor web y se hizo la compra de el nombre del dominio apuntando a una ip real la cual hace que se encuentre en la web
- El fortalecimiento de los conocimientos fue muy exitoso ya que estar basado en lenguajes de programación tradicionales los conocimientos se estancan y con el estudio del lenguaje de programación Ruby on Rails hizo que busque un nuevo lenguaje de programación

- La aplicación web como medio propuesto que fortalece a la institución para la divulgación de su información tanto a los miembros que la conforman como a la comunidad web.
- Para la elaboración del portal web de la mancomunidad de la cuenca del rio mira la herramienta que se utilizo fue aptana para la actualización de las gemas de Ruby y algunos componentes de Rails para la aplicación

4.2 Recomendaciones

- Profundizar el estudio más a fondo del lenguaje Ruby on Rails para aplicaciones de sistemas informáticos.
- Aprovechar la herramienta Aptana con el lenguaje Ruby on Rails ya que este lenguaje es netamente orientado a objetos y en las aplicaciones de sistemas va hacer más rápido y sencillo su elaboración.
- Profundizar y estudiar la herramienta y lenguaje Ruby on Rails en otros sistemas operativos tales como Linux y Mac.
- Constantes actualizaciones de conocimiento del lenguaje de programación ya que hoy en día la actualización de los conocimientos es sumamente importante para el desarrollo de la población dedicada al campo de la informática.
- Al implementar un Desarrollo Rápido de Aplicaciones web, es importante la utilización de Patrones, los cuales ya tienen una funcionalidad general y han sido predefinidos, y así contar con una base consistente y previamente elaborada para la implementación de la web y como no hablar de sistemas informáticos.

GLOSARIO

API

API(Application Program Interface). Interface entre Programa y Aplicación. Conjunto de normas que determinan como debe usarse una determinada función de un programa en una aplicación.

Snippets

(Programación): Es un término del idioma inglés utilizado en programación para referirse a pequeñas partes reusables de código fuente, código binario o texto. Comúnmente son definidas como unidades o métodos funcionales que se pueden integrar fácilmente en módulos mucho más grandes, aportando funcionalidad. También se utiliza la palabra para referirse a la práctica de minimizar el uso de código repetido que es común en muchas funciones, por medio del uso de un solo método que pueda ser reutilizado.

Singleton

El patrón Singleton garantiza que una clase sólo tenga una instancia y proporciona un punto de acceso global a ésta instancia.

Mixins

Un **mixin** es una clase que ofrece cierta funcionalidad para ser heredada por una subclase, pero no está ideada para ser autónoma. Heredar de un mixin no es una forma de especialización sino más bien un medio de obtener funcionalidad. Una subclase puede incluso escoger heredar gran parte o el total de su funcionalidad heredando de uno o más mixins mediante herencia múltiple.

Un mixin puede aplazar la definición y la vinculación de métodos hasta el tiempo de ejecución, aunque los atributos y los parámetros de instanciación siguen siendo definidos en tiempo de compilación. Esto se diferencia del enfoque más comúnmente utilizado, originario del lenguaje de programación Simula, en el que se definen todos los atributos, métodos e inicialización en tiempo de compilación.

Iteradores

Un iterador es un objeto que permite a un programador recorrer los elementos de una colección, sin tener en cuenta su implementación específica. Un iterador muchas veces es llamado "cursor", especialmente dentro del contexto de una base de datos.

Frameworks

Un framework es una estructura de soporte definida, en la cual otro proyecto de software puede ser organizado y desarrollado.

Plugins

Programa que puede anexarse a otro para aumentar sus funcionalidades (generalmente sin afectar otras funciones ni afectar la aplicación principal). No se trata de un parche ni de una actualización, es un módulo aparte que se incluye opcionalmente en una aplicación.

SQlite

ES un sistema de gestión de bases de datos relacional compatible con ACID, contenida en una relativamente pequeña (~275 kiB) biblioteca escrita en C. SQLite es un proyecto de dominio público creado por D. Richard Hipp

BitTorrent

Es un protocolo y programa creados para el intercambio de archivos entre iguales (peer to peer o P2P), creados por Bram Cohen, programador estadounidense. Específicamente el protocolo fue diseñado en abril de 2001, e implementado y lanzado el 2 de julio de ese año

BIBLIOGRAFÍA

LIBROS

[LIB 01] LUCAS CARLSON Y LEONARDO RICHARDSON, Curso de Ruby, imprenta ANAYA Multimedia/ 2007, 1024 páginas

SITIOS DE INTERNET

[URL01] WIKIPEDIA. Ruby. Revisado 2012 [en línea], from http://www.ruby-lang.org/es/

[URL 02] MARK SLAGEL. Ruby User Guide. Revisado 2011 [en línea], from: http://www.rubyist.net/~slagell/ruby/index.html

[URL 03] MARKMONITOR INC. Revisado 2012 [en línea], from http://peachep.wordpress.com/2007/06/07/la-arquitectura-mvc-model-control-view-en-rails/

[URL 04] Wikimedia Foundation, Inc. Revisado 2011 [en línea], from: http://es.wikipedia.org/wiki/Ruby_on_Rails

[URL 05] Aptana,Inc. Fashion Island Blvd, Suite 500, San Mateo, CA 94404 US. Revisado 2011 [en línea], from: http://www.aptana.com/

[URL 06] Wikimedia Foundation, Inc. Revisado 2011 [en línea], from: http://es.wikipedia.org/wiki/NetBeans

[URL 07] eNom, Inc. Revisado 2011 [en línea], from: http://railsplugins.org/plugins

[URL 08] NEW DREAM NETWORK, LLC. Revisado 2011 [en línea], from:

http://www.dokshor.com/las-mejores-herramientas-php-y-ruby-on-rails-desarrollo-web-profesional

[URL 09] GANDI SAS. Revisado 2011 [en línea], from: http://www.xdocs400.com/spip.php?article381&lang=fr

[URL 10] Wikimedia Foundation, Inc. Revisado 2011 [en línea], from: http://es.wikipedia.org/wiki/PostgreSQL

[URL 11] NETWORK SOLUTIONS, LLC. Revisado 2011 [en línea], from: http://www.postgresql.org.es/

[URL 12] NETWORK SOLUTIONS, LLC. Revisado 2011 [en línea], from: http://textmate.softonic.com/mac

[URL 13] Wikimedia Foundation, Inc. Revisado 2011 [en línea], from: http://es.wikipedia.org/wiki/TextMate

[URL 14] Andreas Schwarz. Revisado 2011 [en línea], from: http://www.ruby-forum.com/topic/144754

[URL 15] Stack Exchange, Inc. Revisado 2011 [en línea], from: http://stackoverflow.com/questions/3765843/aptana-vs-netbeans-for-ruby-development

[URL 16] NEW DREAM NETWORK, LLC. Revisado 2011 [en línea], from: http://sobrerailes.com/pages/en_marcha_con_rails/

[URL 17] GANDI SAS. Revisado 2011 [en línea], from: http://gemaroja.com/dando-los-primeros-pasos-en-ruby

[URL 18] GODADDY.COM, LLC. Revisado 2011 [en línea], from:

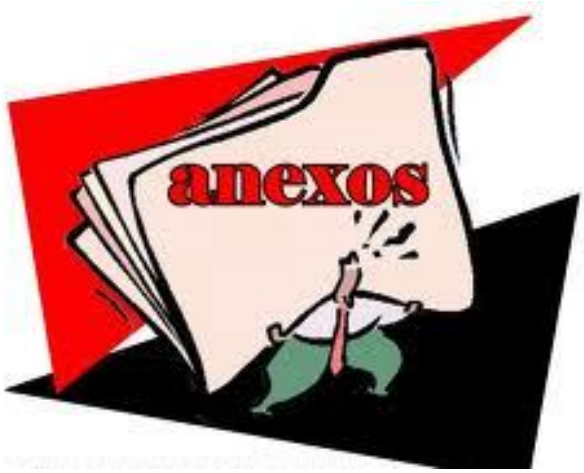
http://trisfera.com/atrium/general/node/359

[URL 19] Wikimedia Foundation, Inc. Revisado 2011 [en línea], from: http://es.wikibooks.org/wiki/Programaci%C3%B3n_en_Ruby

[URL 20] Wikimedia Foundation, Inc. Revisado 2011 [en línea], from: http://es.wikipedia.org/wiki/Eclipse_(software)

[URL 21] Wikimedia Foundation, Inc. Revisado 2011 [en línea], from: http://es.wikipedia.org/wiki/NetBeans

ANEXOS



ANTEPROYECTO DE TESIS
MANUAL TECNICO
MANUAL DE USUARIO

Universidad Técnica del Norte

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS
CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

PLAN DEL PROYECTO DE TITULACIÓN

Propuesto por: Christian Fernando Realpe Rosero	**Áreas Técnicas del Tema:** Aplicaciones Web
Director del Proyecto: Ing. Marco Pusdá	**Fecha:** 2010-07-02

1. Tema o Título del proyecto

Análisis y Estudio de Tecnología Ruby on Rails con bases de datos Postgres para Aplicaciones Web 2.0
Aplicativo: Implementación del Portal Web 2.0 para la Mancomunidad de la Cuenca del Río Mira

2. Antecedentes

La Mancomunidad de la Cuenca del Río Mira es una organización que genera un continuo desarrollo de actividades de proyectos hacia los diversos cantones de dicha organización y para que esto pueda plasmarse hacia la sociedad y porque no decir hacia el mundo, la información se la va a manejar en el Portal Web de la Mancomunidad de la Cuenca del Río Mira

La Mancomunidad de la Cuenca del Río Mira se define como una modalidad alternativa de gestión "que procura asumir responsabilidades compartidas con el concurso de varias administraciones locales, que quieren desarrollar estrategias comunes que favorezcan la eficiencia y eficacia de la gestión local". En otras palabras, "la Mancomunidad es un instrumento caracterizado por la voluntariedad de sus integrantes orientada a la gestión de competencias (cuando de por medio existe un proceso de descentralización), de servicios o la ejecución de obras (cuando éstos sobrepasan el territorio del gobierno local). La mancomunidad permite la administración conjunta, total o parcial de dos o más gobiernos seccionales".

En este caso la Mancomunidad de la Cuenca del Río Mira está conformado por cinco municipales de la provincia de Imbabura y Carchi (Urcuquí, Pimampiro, Mira, Espejo, Bolívar)

3. Problema

En la actualidad la tendencia de crecimiento de las instituciones se basan especialmente en el que se hacen conocer de todas las actividades que realizan y una de las mejores maneras es montando toda su información a través de la web y en este caso la Mancomunidad de la Cuenca del Río Mira carece de un Portal Web operativo que atienda los requerimientos de desarrollar acciones tendientes a garantizar el mejoramiento de la calidad de vida de la población de los respectivos cantones mediante la consecución de los fines propuestos como es el caso de Gestionar planes, programas y proyectos comunes; Unificar y potenciar los esfuerzos de las Entidades Asociadas; Cooperar para el eficaz cumplimiento de los fines de las Entidades; Realizar esfuerzos conjuntos y Fortalecer los procesos de gobernabilidad y ejercicio de la democracia.

Los cantones que conforman la Mancomunidad de la Cuenca del Río Mira no se encuentran informados naturalmente al instante de las actividades que realiza la organización, en efecto ya como nos encontramos en otra era y es la era del internet es bastante afectiva a la información, esto hará que los miembros pueda estar informadas de lo que se realiza en la Mancomunidad de la Cuenca del Río Mira con la información respectiva levantada hacia este Portal Web.

La Mancomunidad de la Cuenca del Río Mira al contar con el Portal Web se encontrara dispuesta a la divulgación de la información de sus actividades y tareas que se realizan dentro de la institución con el fin de realizar los proyectos sociales de la cual se encarga la mancomunidad y así no pasar desapercibida la institución ante comunidad Web.

El continuo desarrollo de actividades que realiza la Mancomunidad de la Cuenca del Río Mira tiene la necesidad de informar sus actividades en el Portal Web que se encontrará en forma de actualización permanente para la difusión de su información hacia la comunidad web y sus miembros de la institución.

4. Objetivos

Objetivo General

- Realizar un análisis y estudio de la tecnología Ruby on Rails y la base de datos Postgres
- Construir un Portal Web 2.0 aplicando tecnología Ruby on Rails y base de datos Postgres para la Mancomunidad de la Cuenca del Río Mira que brinde la información necesaria para la divulgación en la zona Norte del Ecuador.

Objetivos Específicos

- Realizar un estudio y diagnóstico del marco teórico y fortalecer los conocimientos
- Definir el entorno de desarrollo con la cuales se realizará la implementación del Portal Web
- Desarrollar un Portal Web 2.0 con Ruby On Rails
- Integrar, Validar e implementar el Portal Web 2.0

5. Alcance

El Portal Web para la Mancomunidad de la Cuenca del Río Mira contendrá 5 módulos, los cuales nos permitirán manejar información de gran importancia para la institución y su solidificación mediante la captación de apoyo interno y externo de instituciones públicas y privadas. (Ver Figura 1)

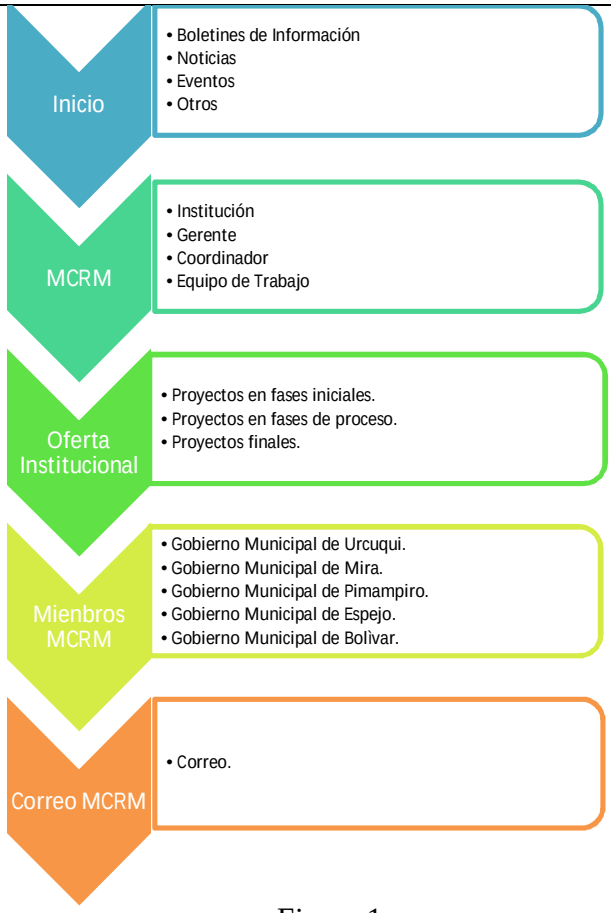


Figura 1

Inicio.- Presenta información principal del Portal Web, además de forma detallada se presentan los otros submenús para que le usuario se informe de manera clara y precisa.

MCRM.-Ofrece de la información de la institución en general.

Oferta Institucional.-Información de los procesos de proyectos que realiza la Mancomunidad de la Cuenca del Río Mira en sus tres fases.

Miembros MCRM.- Información acerca de los miembros que conforman la Mancomunidad de la Cuenca del Río Mira

Correo MCRM.- Correo correspondiente a la Mancomunidad de la Cuenca del Río Mira

Para llegar a cumplir el esquema anterior para la aplicación tenemos que armar la base de datos que servirá para almacenar la información necesaria para este cometido. Es importante el estudio del lenguaje de programación Ruby que será la base para llegar al desarrollo de la aplicación. (Ver Figura 2)

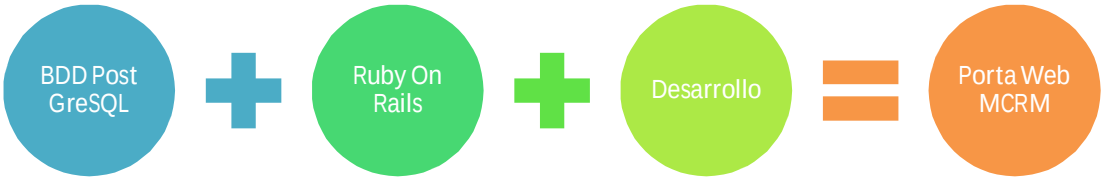


Figura 2

Ruby On Rails trabaja con MVC haciendo que la aplicación sea más segura y escalable, con proyección a futuro. (Ver Figura 3)

MVC
● Modelo
Responsable de mantener el "estado" de la aplicación.
● Vista
Responsable de presentar la interfaz y la información al usuario
● Controlador
Organiza la aplicación, Recibe eventos del exterior, interactúa con el modelo y actualiza la información de las vistas

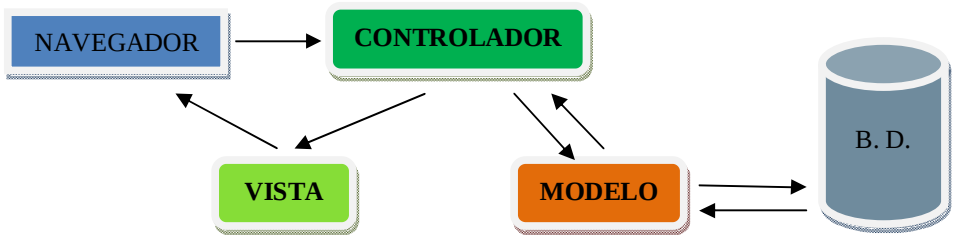


Figura 3

6. Justificación del Proyecto

La Mancomunidad de la Cuenca del Río Mira está buscando la forma de potencializar su trabajo en la llamada zona Norte del Ecuador y para estar acorde con el desarrollo de la tecnología y dar paso al adelanto de la misma; ve la necesidad y se acoge la oportunidad de elaborar un Portal Web para que este a su vez se convierta en una fortaleza interna, convirtiéndose además en la puerta de enlace para futuras negociaciones y apoyos por parte de otras instituciones privadas o públicas, nacionales o internacionales que trabaje de manera eficiente y eficaz acogiendo las actividades y necesidades de la institución.

Los miembros de la Mancomunidad Cuenca del Río Mira contarán con la información pertinente de las actividades que se están realizando en la institución a través del Portal

Web que se va a diseñar, el cual como ya lo dijimos es un medio informativo.

La Mancomunidad de la Cuenca del Río Mira contará con correo electrónico interno de la institución el que servirá tanto para la parte operativa como para los miembros, el uso ya es administración de los directivos de la institución y sus miembros.

La realización del Portal Web estará diseñada con una estructura dinámica y atractiva al usuario de fácil apertura para actualización y subida de información valiosa de cada uno de los miembros que lo conforman o que deseen ser parte con apoyo de todo tipo.

7. Temas Afines Realizados

Autor	Tema	Similitud	Diferencia
Vicente Alexander Guevara Vega / Marco Javier Carlosama Chicaiza	FASE DE DISEÑO, DESARROLLO, IMPLEMENTACION Y TRANSFERENCIA TECNICA DEL UNIPORTAL DE LA UNIVERSIDAD TECNICA DEL NORTE	La propuesta que se presentó por parte de el tema a fin es similar en la presentación de un Portal Web	La diferencia existente con el desarrollo de mi tesis es el estudio que se realiza para llegar al objetivo de presentar el trabajo terminado, el estudio que se realizó para el tema a fin es con Jomla y mi estudio es con Ruby on Rails y Base de Datos Postgres
Rea Enríquez Roldan Fernando	APLICACIONES WEB CON MVC USANDO TECNOLOGÍA JSP.	La similitud de la tesis propuesta es que se trabaja con patrones de diseño MCV	La diferencia que existe en la propuesta que presento, trabajaré con tecnologías Ruby on Rails y Base de Datos Postgres, lo que propone el tema a fin es con tecnologías JSP

8. Temario

1. Ruby on Rails y bases de datos Postgres

1.1. Qué es ruby?

1.1.1. Introducción
1.1.2. Características de ruby
1.1.3. Estudio de ruby

1.2. Herramientas de Desarrollo de Ruby on Rails

1.2.1 Introducción
1.2.2 Definición
1.2.3 Fusión Aplicaciones
1.2.4 Arquitectura de Desarrollo

1.3. Postgres y la fusion con ruby on rails

1.3.1. Introducción
1.3.2. La Base de Datos Postgres
1.3.3. Fusion de ruby on rails con postgres

2. Herramientas compatibles con Ruby on Rails

2.3. Descripción de herramientas afines

2.4. Compatibilidad y soporte

3. Análisis, Diseño e Implementación de "Portal Web 2.0 para la Mancomunidad de la Cuenca del Rio Mira"

3.3. Análisis de Requerimientos

3.3.1. Flujo de Trabajo
3.3.2. Casos de Uso
3.3.3. Diagramas de Clases
3.3.4. Infraestructura Tecnológica.

3.4. Diseño

3.5. Desarrollo e Implementación

3.5.1. Especificaciones Tecnológicas
3.5.2. Arquitectura de Aplicación

3.6. Producto

3.6.1. Instalación
3.6.2. Descripción del Portal Web
3.6.2.1. Administrativa
3.6.2.2. Evaluación

3.7. Pruebas Finales

4. Conclusiones y Recomendaciones

4.3. Conclusiones

4.4. Recomendaciones

9. Bibliografía

- http://www.tufuncion .com/ruby-tutorial-basico Tutorial básico de Ruby
- http://sobrerailes.com/pages/en_marcha_con_rails/ Sobre Rails
- http://www.rubycentral.com/book/index.html Ruby
- http://rubytutorial.wikidot.com/introduccion Ruby Tutorial

- Lucas Carlson y Leonard Richardson, CURSO DE RUBY, 1ra. Edición, Editorial ANAYA MULTIMEDIA, Mayo 2007

- Segaran Toby, INTELIGENCIA COLECTIVA. DESARROLLO DE APLICACIONES WEB 2.0, 1ra. Edición, Editorial ANAYA MULTIMEDIA, 2008

10. Cronograma de Actividades

Cronograma de Actividades

Nro.	ACTIVIDADES	DURACION	Duración en Meses																															
			Mes 1				Mes 2				Mes 3				Mes 4				Mes 5				Mes 6				Mes 7				Mes 8				Mes 9			
		En semanas	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	Investigación	2																																				
2	Recopilación de información	2																																				
3	Documento visión	2																																				
4	Plan de desarrollo del proyecto	2																																				
5	Listas de riesgos	2																																				
6	Modelos de casos de uso	2																																				
7	Desarrollar modelo entidad- relación	2																																				
8	Elaborar la arquitectura de la aplicación	2																																				
9	Construcción de aplicación	12																																				
10	Realización de pruebas	2																																				
11	Funcionamiento de la aplicación	2																																				
12	Redacción de conclusiones y recomendaciones	2																																				
13	Redacción de documento final	23																																				

11. Presupuesto

DETALLE		COSTO actual	COSTO REAL
HARDWARE	Equipo de Computación	1200.00	732.00
	Hosting BBDD PostgreSQL	85.00	85.00
	Impresora	80.00	0.00
SOFTWARE	Ruby on Rails	0.00	0.00
	Aptana	0.00	0.00
	Otros	300.00	300.00
CAPACITACION y BIBLIOGRAFIA	Libros	100.00	0.00
	Internet	240.00	0.00
	Cursos	500.00	500
MATERIALES	Reproducción de documentos	100.00	100.00
	Útiles de oficina	80.00	80.00
	Impresión de documentos	120.00	120.00
	Varios	250.00	250.00
SUBTOTAL		3055.00	1667.00
15%IMPREVISTOS		458.25	250,05
TOTAL		3513.25	1917,05

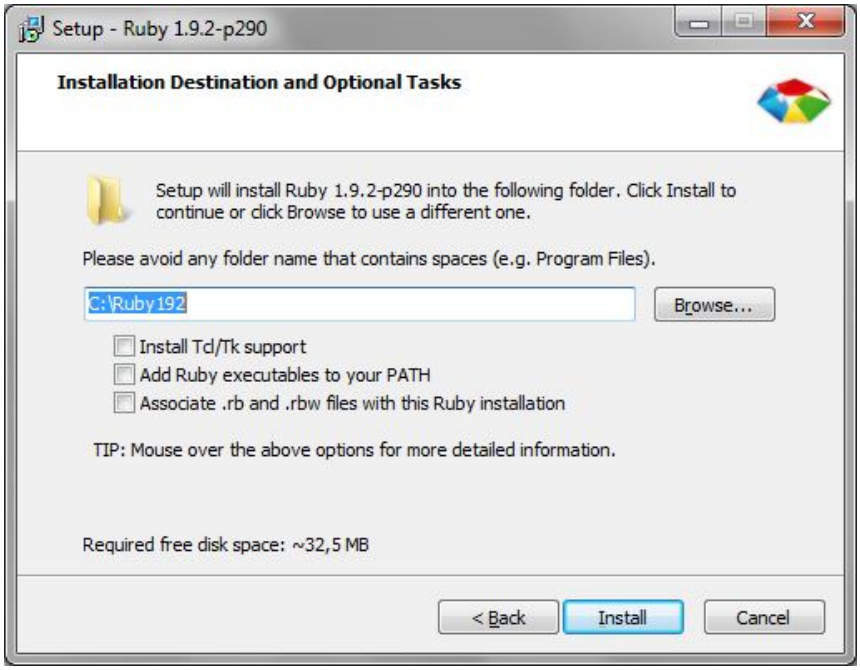
ESTUDIANTE 1

DIRECTOR DEL PROYECTO

MANUAL TECNICO

INSTALACIÓN

1.) instalar rugby rubyinstaller-1.9.2-p290



2.) Desempaquetar devkit en el disco c

3.) Abrir una consola de comandos e ingresar al devkit ejecutar

ruby dk.rb init

ruby dk.rb install

4. Copiar la carpeta web_mcrm al disco c

Desde la consola entrar a la carpeta

Ejecutar: gem install rails

Ejecutar: bundle install

Ejecutar: gem update

Ejecutar: C:\web_mcrm\script\rails server

CONTROLLERS

CONTROLADOR ADMINISTRADOR

La siguiente estructura que se encuentra a continuación se basa en el controlador para el ingreso como parte de administrador para en este caso poder subir, actualizar o borrar información para luego se entregada al usuario final en este caso a la publicación de sus datos

```
class AdministradorController < ApplicationController
  def index
  end

  def login
    @usuario = params[:usuario]
    @pass = params[:pass]
    @localusuario = Usuario.where("usuario=:usuario and
password=:pass",{:usuario=>@usuario,:pass=>@pass}).first
    #@pass = Digest::MD5.hexdigest(params[:pass])
    @error = ""
    if @localusuario == nil
      @error = "No existe el usuario"
    end
    session[:iniciado] = "tok123railstkf001"
  end

  def logout
    session[:iniciado] = nil
  end

end
```

CONTROLADOR APLICACIÓN

Inicializa todo lo eventos para hacer uso de los controlador de todas las acciones de los modulos a trabajar.

```
class ApplicationController < ActionController::Base
  protect_from_forgery
end
```

CONTROLADOR BOLETINES

Realiza el control de crear, editar, actualizar y eliminar datos de Boletines de información haciendo uso de la base de datos, para luego publicar en la View (vistas)

```
class BoletinsController < ApplicationController
  # GET /boletins
  # GET /boletins.json
```

Definición de controlador que interactúa con la View de la aplicación llamando al formulario índex

```
    def index
    @boletins = Boletin.all

    respond_to do |format|
      format.html # index.html.erb
      format.json { render json: @boletins }
    end
  end

  # GET /boletins/1
  # GET /boletins/1.json
```

Llama al formato de visualización para realizar las acciones que queremos realizar interactuando con la base de datos y la View

```
  def show
    @boletin = Boletin.find(params[:id])

    respond_to do |format|
      format.html # show.html.erb
      format.json { render json: @boletin }
```

```
  end
end

# GET /boletins/new
# GET /boletins/new.json
```

Realiza el llamamiento al formulario para realizar la acción de crear de un nuevo boletín

```
def new
  @boletin = Boletin.new

  respond_to do |format|
    format.html # new.html.erb
    format.json { render json: @boletin }
  end
end

# GET /boletins/1/edit
```

Realiza el llamamiento al formulario para realizar la acción de modificar o editar un boletín

```
def edit
  @boletin = Boletin.find(params[:id])
end

# POST /boletins
# POST /boletins.json
def create
  @boletin = Boletin.new(params[:boletin])

  respond_to do |format|
    if @boletin.save
      format.html { redirect_to @boletin, notice: 'Boletin was successfully created.' }
      format.json { render json: @boletin, status: :created, location: @boletin }
    else
      format.html { render action: "new" }
      format.json { render json: @boletin.errors, status: :unprocessable_entity }
    end
  end
end

# PUT /boletins/1
# PUT /boletins/1.json
```

Realiza el llamamiento al formulario para realizar la acción de actualizar un boletín

```
  def update
    @boletin = Boletin.find(params[:id])

    respond_to do |format|
      if @boletin.update_attributes(params[:boletin])
        format.html { redirect_to @boletin, notice: 'Boletin was successfully updated.' }
        format.json { head :ok }
      else
        format.html { render action: "edit" }
        format.json { render json: @boletin.errors, status: :unprocessable_entity }
      end
    end
  end

  # DELETE /boletins/1
  # DELETE /boletins/1.json
```

Realiza el llamamiento al formulario para realizar la acción de eliminar un boletín

```
  def destroy
    @boletin = Boletin.find(params[:id])
    @boletin.destroy

    respond_to do |format|
      format.html { redirect_to boletins_url }
      format.json { head :ok }
    end
  end
end
```

CONTROLADOR CONTENIDOS

Realiza el control de crear, editar, actualizar y eliminar datos de Contenidos de información haciendo uso de la base de datos, para luego publicar en la View (vistas)

```
class ContenidosController < ApplicationController
  # GET /contenidos
  # GET /contenidos.json
```

Definición de controlador que interactúa con la View de la aplicación llamando al formulario índex

```
  def index
    @contenidos = Contenido.all

    respond_to do |format|
      format.html # index.html.erb
      format.json { render json: @contenidos }
    end
  end

  # GET /contenidos/1
  # GET /contenidos/1.json
```

Llama al formato de visualización para realizar las acciones que queremos realizar interactuando con la base de datos y la View

```
  def show
    @contenido = Contenido.find(params[:id])

    respond_to do |format|
      format.html # show.html.erb
      format.json { render json: @contenido }
    end
  end

  # GET /contenidos/new
  # GET /contenidos/new.json
```

Realiza el llamamiento al formulario para realizar la acción de crear de un nuevo contenido

```
  def new
    @contenido = Contenido.new

    respond_to do |format|
      format.html # new.html.erb
      format.json { render json: @contenido }
    end
  end

  # GET /contenidos/1/edit
```

Realiza el llamamiento al formulario para realizar la acción de modificar o editar un contenido

```
  def edit
```

```
  @contenido = Contenido.find(params[:id])
 end

 # POST /contenidos
 # POST /contenidos.json
 def create
  @contenido = Contenido.new(params[:contenido])

  respond_to do |format|
   if @contenido.save
    format.html { redirect_to @contenido, notice: 'El contenido ha sido creado
satisfactoriamente.' }
    format.json { render json: @contenido, status: :created, location: @contenido }
   else
    format.html { render action: "new" }
    format.json { render json: @contenido.errors, status: :unprocessable_entity }
   end
  end
 end

 # PUT /contenidos/1
 # PUT /contenidos/1.json
```

Realiza el llamamiento al formulario para realizar la acción de actualizar un contenido

```
 def update
  @contenido = Contenido.find(params[:id])

  respond_to do |format|
   if @contenido.update_attributes(params[:contenido])
    format.html { redirect_to @contenido, notice: 'El contenido ha sido actualizado
satisfactoriamente.' }
    format.json { head :ok }
   else
    format.html { render action: "edit" }
    format.json { render json: @contenido.errors, status: :unprocessable_entity }
   end
  end
 end

 # DELETE /contenidos/1
 # DELETE /contenidos/1.json
```

Realiza el llamamiento al formulario para realizar la acción de eliminar un contenido

```
  def destroy
    @contenido = Contenido.find(params[:id])
    @contenido.destroy

    respond_to do |format|
      format.html { redirect_to contenidos_url }
      format.json { head :ok }
    end
  end
end
```

CONTROLADOR EVENTOS

Realiza el control de crear, editar, actualizar y eliminar datos de eventos haciendo uso de la base de datos, para luego publicar en la View (vistas)

```
class EventosController < ApplicationController
  # GET /eventos
  # GET /eventos.json
```

Definición de controlador que interactúa con la View de la aplicación llamando al formulario índex

```
  def index
    @eventos = Evento.all

    respond_to do |format|
      format.html # index.html.erb
      format.json { render json: @eventos }
    end
  end

  # GET /eventos/1
  # GET /eventos/1.json
```

Llama al formato de visualización para realizar las acciones que queremos realizar interactuando con la base de datos y la View

```
  def show
    @evento = Evento.find(params[:id])

    respond_to do |format|
```

```
    format.html # show.html.erb
    format.json { render json: @evento }
  end
end

# GET /eventos/new
# GET /eventos/new.json
```

Realiza el llamamiento al formulario para realizar la acción de crear de un nuevo evento

```
def new
  @evento = Evento.new

  respond_to do |format|
    format.html # new.html.erb
    format.json { render json: @evento }
  end
end

# GET /eventos/1/edit
```

Realiza el llamamiento al formulario para realizar la acción de modificar o editar un evento

```
def edit
  @evento = Evento.find(params[:id])
end

# POST /eventos
# POST /eventos.json
```

Definición para crear el nuevo evento

```
def create
  @evento = Evento.new(params[:evento])

  respond_to do |format|
    if @evento.save
      format.html { redirect_to @evento, notice: 'Evento was successfully created.' }
      format.json { render json: @evento, status: :created, location: @evento }
    else
      format.html { render action: "new" }
      format.json { render json: @evento.errors, status: :unprocessable_entity }
    end
  end
```

```
  end

  # PUT /eventos/1
  # PUT /eventos/1.json
```

Realiza el llamamiento al formulario para realizar la acción de actualizar un evento

```
  def update
    @evento = Evento.find(params[:id])

    respond_to do |format|
      if @evento.update_attributes(params[:evento])
        format.html { redirect_to @evento, notice: 'Evento was successfully updated.' }
        format.json { head :ok }
      else
        format.html { render action: "edit" }
        format.json { render json: @evento.errors, status: :unprocessable_entity }
      end
    end
  end

  # DELETE /eventos/1
  # DELETE /eventos/1.json
```

Realiza el llamamiento al formulario para realizar la acción de eliminar un evento

```
  def destroy
    @evento = Evento.find(params[:id])
    @evento.destroy

    respond_to do |format|
      format.html { redirect_to eventos_url }
      format.json { head :ok }
    end
  end
end
```

CONTOLADOR FRONTEND

Realiza el control de crear, editar, actualizar y eliminar datos de información haciendo uso de la base de datos, para luego publicar en la View (vistas) llamada de todas los frontend de la aplicación

```
class FrontendController < ApplicationController
 def boletines
  @boletins = Boletin.where("disponible='SI'")

  respond_to do |format|
   format.html # index.html.erb
   format.json { render json: @boletins }
  end
 end

 def boletin_show
  @boletin = Boletin.find(params[:id])

  respond_to do |format|
   format.html # show.html.erb
   format.json { render json: @boletin }
  end
 end

 def noticias
  @noticia = Noticium.where("disponible='SI'")

  respond_to do |format|
   format.html # index.html.erb
   format.json { render json: @noticia }
  end
 end

 def noticia_show
  @noticium = Noticium.find(params[:id])

  respond_to do |format|
   format.html # show.html.erb
   format.json { render json: @noticium }
  end
 end

 def proyectos
  @proyectos = Proyecto.where("disponible='SI'")

  respond_to do |format|
   format.html # index.html.erb
   format.json { render json: @proyectos }
```

```
  end
end

def proyecto_show
  @proyecto = Proyecto.find(params[:id])

  respond_to do |format|
    format.html # show.html.erb
    format.json { render json: @proyecto }
  end
end

def eventos
  @eventos = Evento.where("disponible='SI'")

  respond_to do |format|
    format.html # index.html.erb
    format.json { render json: @eventos }
  end
end

def periodos
  @periodos = Periodo.where("disponible='SI'")

  respond_to do |format|
    format.html # index.html.erb
    format.json { render json: @periodos }
  end
end

def periodo_show
  @periodo = Periodo.find(params[:id])

  respond_to do |format|
    format.html # show.html.erb
    format.json { render json: @periodo }
  end
end

def galeria

end

def miembros
```

```
    @miembros = Miembro.where("disponible='SI'")

    respond_to do |format|
      format.html # index.html.erb
      format.json { render json: @miembros }
    end
  end

  def miembro_show
    @miembro = Miembro.find(params[:id])

    respond_to do |format|
      format.html # show.html.erb
      format.json { render json: @miembro }
    end
  end

end
```

CONTROLADOR INICIO

Realiza el control de crear, editar, actualizar y eliminar datos de inicio de información haciendo uso de la base de datos, para luego publicar en la View (vistas)

```
class InicioController < ApplicationController
  def index
    @contenido = Contenido.find(1)

    respond_to do |format|
      format.html # show.html.erb
      format.json { render json: @contenido }
    end
  end

end
```

CONTROLADOR MIEMBROS

Realiza el control de crear, editar, actualizar y eliminar datos de Miembros de información haciendo uso de la base de datos, para luego publicar en la View (vistas)

```
class MiembrosController < ApplicationController
```

```
# GET /miembros
# GET /miembros.json
```

Definición de controlador que interactúa con la View de la aplicación llamando al formulario índex

```
def index
  @miembros = Miembro.all

  respond_to do |format|
    format.html # index.html.erb
    format.json { render json: @miembros }
  end
end

# GET /miembros/1
# GET /miembros/1.json
```

Llama al formato de visualización para realizar las acciones que queremos realizar interactuando con la base de datos y la View

```
def show
  @miembro = Miembro.find(params[:id])

  respond_to do |format|
    format.html # show.html.erb
    format.json { render json: @miembro }
  end
end

# GET /miembros/new
# GET /miembros/new.json
```

Realiza el llamamiento al formulario para realizar la acción de crear de un nuevo miembro

```
def new
  @miembro = Miembro.new

  respond_to do |format|
    format.html # new.html.erb
    format.json { render json: @miembro }
  end
end
```

```
  # GET /miembros/1/edit
```

Realiza el llamamiento al formulario para realizar la acción de modificar o editar un miembro

```
  def edit
    @miembro = Miembro.find(params[:id])
  end

  # POST /miembros
  # POST /miembros.json
  def create
    @miembro = Miembro.new(params[:miembro])

    respond_to do |format|
      if @miembro.save
        format.html { redirect_to @miembro, notice: 'Miembro was successfully created.' }
        format.json { render json: @miembro, status: :created, location: @miembro }
      else
        format.html { render action: "new" }
        format.json { render json: @miembro.errors, status: :unprocessable_entity }
      end
    end
  end

  # PUT /miembros/1
  # PUT /miembros/1.json
```

Realiza el llamamiento al formulario para realizar la acción de actualizar un miembro

```
  def update
    @miembro = Miembro.find(params[:id])

    respond_to do |format|
      if @miembro.update_attributes(params[:miembro])
        format.html { redirect_to @miembro, notice: 'Miembro was successfully updated.' }
        format.json { head :ok }
      else
        format.html { render action: "edit" }
        format.json { render json: @miembro.errors, status: :unprocessable_entity }
      end
    end
  end

  # DELETE /miembros/1
```

```
  # DELETE /miembros/1.json
```

Realiza el llamamiento al formulario para realizar la acción de eliminar un miembro

```
  def destroy
    @miembro = Miembro.find(params[:id])
    @miembro.destroy

    respond_to do |format|
      format.html { redirect_to miembros_url }
      format.json { head :ok }
    end
  end
end
```

CONTROLADOR NOTICIA

Realiza el control de crear, editar, actualizar y eliminar datos de Noticias de información haciendo uso de la base de datos, para luego publicar en la View (vistas)

```
class NoticiaController < ApplicationController
  # GET /noticia
  # GET /noticia.json
```

Definición de controlador que interactúa con la View de la aplicación llamando al formulario índex

```
  def index
    @noticia = Noticium.all

    respond_to do |format|
      format.html # index.html.erb
      format.json { render json: @noticia }
    end
  end

  # GET /noticia/1
  # GET /noticia/1.json
```

Llama al formato de visualización para realizar las acciones que queremos realizar interactuando con la base de datos y la View

```
  def show
```

```
  @noticium = Noticium.find(params[:id])

  respond_to do |format|
    format.html # show.html.erb
    format.json { render json: @noticium }
  end
 end

 # GET /noticia/new
 # GET /noticia/new.json
```

Realiza el llamamiento al formulario para realizar la acción de crear de un nuevo noticia

```
 def new
  @noticium = Noticium.new

  respond_to do |format|
    format.html # new.html.erb
    format.json { render json: @noticium }
  end
 end



 # GET /noticia/1/edit
```

Realiza el llamamiento al formulario para realizar la acción de modificar o editar un noticia

```
 def edit
  @noticium = Noticium.find(params[:id])
 end

 # POST /noticia
 # POST /noticia.json
 def create
  @noticium = Noticium.new(params[:noticium])

  respond_to do |format|
    if @noticium.save
      format.html { redirect_to @noticium, notice: 'Noticium was successfully created.' }
      format.json { render json: @noticium, status: :created, location: @noticium }
    else
      format.html { render action: "new" }
      format.json { render json: @noticium.errors, status: :unprocessable_entity }
```

```
    end
  end
end

# PUT /noticia/1
# PUT /noticia/1.json
```

Realiza el llamamiento al formulario para realizar la acción de actualizar un noticia

```
def update
  @noticium = Noticium.find(params[:id])

  respond_to do |format|
    if @noticium.update_attributes(params[:noticium])
      format.html { redirect_to @noticium, notice: 'Noticium was successfully updated.' }
      format.json { head :ok }
    else
      format.html { render action: "edit" }
      format.json { render json: @noticium.errors, status: :unprocessable_entity }
    end
  end
end

# DELETE /noticia/1
# DELETE /noticia/1.json
```

Realiza el llamamiento al formulario para realizar la acción de eliminar un noticia

```
def destroy
  @noticium = Noticium.find(params[:id])
  @noticium.destroy

  respond_to do |format|
    format.html { redirect_to noticia_url }
    format.json { head :ok }
  end
end
end
```

CONTROLADOR PERIODOS

Realiza el control de crear, editar, actualizar y eliminar datos de Periodos de información haciendo uso de la base de datos, para luego publicar en la View (vistas)

```
class PeriodosController < ApplicationController
  # GET /periodos
  # GET /periodos.json
```

Definición de controlador que interactúa con la View de la aplicación llamando al formulario índex

```
  def index
    @periodos = Periodo.all

    respond_to do |format|
      format.html # index.html.erb
      format.json { render json: @periodos }
    end
  end

  # GET /periodos/1
  # GET /periodos/1.json
```

Llama al formato de visualización para realizar las acciones que queremos realizar interactuando con la base de datos y la View

```
  def show
    @periodo = Periodo.find(params[:id])

    respond_to do |format|
      format.html # show.html.erb
      format.json { render json: @periodo }
    end
  end

  # GET /periodos/new
  # GET /periodos/new.json
```

Realiza el llamamiento al formulario para realizar la acción de crear de un nuevo periodo

```
  def new
    @periodo = Periodo.new

    respond_to do |format|
      format.html # new.html.erb
      format.json { render json: @periodo }
    end
  end
```

```
  # GET /periodos/1/edit
```

Realiza el llamamiento al formulario para realizar la acción de modificar o editar un periodo

```
  def edit
    @periodo = Periodo.find(params[:id])
  end

  # POST /periodos
  # POST /periodos.json


  def create
    @periodo = Periodo.new(params[:periodo])

    respond_to do |format|
      if @periodo.save
        format.html { redirect_to @periodo, notice: 'Periodo was successfully created.' }
        format.json { render json: @periodo, status: :created, location: @periodo }
      else
        format.html { render action: "new" }
        format.json { render json: @periodo.errors, status: :unprocessable_entity }
      end
    end
  end

  # PUT /periodos/1
  # PUT /periodos/1.json
```

Realiza el llamamiento al formulario para realizar la acción de actualizar periodo

```
  def update
    @periodo = Periodo.find(params[:id])

    respond_to do |format|
      if @periodo.update_attributes(params[:periodo])
        format.html { redirect_to @periodo, notice: 'Periodo was successfully updated.' }
        format.json { head :ok }
      else
        format.html { render action: "edit" }
        format.json { render json: @periodo.errors, status: :unprocessable_entity }
      end
    end
```

```
  end

  # DELETE /periodos/1
  # DELETE /periodos/1.json
```

Realiza el llamamiento al formulario para realizar la acción de eliminar un periodo

```
  def destroy
    @periodo = Periodo.find(params[:id])
    @periodo.destroy

    respond_to do |format|
      format.html { redirect_to periodos_url }
      format.json { head :ok }
    end
  end
end
```

CONTROLADOR PROYECTOS

Realiza el control de crear, editar, actualizar y eliminar datos de Proyectos de información haciendo uso de la base de datos, para luego publicar en la View (vistas)

```
class ProyectosController < ApplicationController
  # GET /proyectos
  # GET /proyectos.json
```

Definición de controlador que interactúa con la View de la aplicación llamando al formulario índex

```
  def index
    @proyectos = Proyecto.all

    respond_to do |format|
      format.html # index.html.erb
      format.json { render json: @proyectos }
    end
  end

  # GET /proyectos/1
  # GET /proyectos/1.json
```

Llama al formato de visualización para realizar las acciones que queremos realizar interactuando con la base de datos y la View

```
def show
  @proyecto = Proyecto.find(params[:id])

  respond_to do |format|
    format.html # show.html.erb
    format.json { render json: @proyecto }
  end
end

# GET /proyectos/new
# GET /proyectos/new.json
```

Realiza el llamamiento al formulario para realizar la acción de crear de un nuevo proyecto

```
def new
  @proyecto = Proyecto.new

  respond_to do |format|
    format.html # new.html.erb
    format.json { render json: @proyecto }
  end
end

# GET /proyectos/1/edit
```

Realiza el llamamiento al formulario para realizar la acción de modificar o editar un proyecto

```
def edit
  @proyecto = Proyecto.find(params[:id])
end

# POST /proyectos
# POST /proyectos.json
def create
  @proyecto = Proyecto.new(params[:proyecto])

  respond_to do |format|
    if @proyecto.save
      format.html { redirect_to @proyecto, notice: 'Proyecto was successfully created.' }
      format.json { render json: @proyecto, status: :created, location: @proyecto }
    else
      format.html { render action: "new" }
      format.json { render json: @proyecto.errors, status: :unprocessable_entity }
```

```
    end
  end
end

# PUT /proyectos/1
# PUT /proyectos/1.json
```

Realiza el llamamiento al formulario para realizar la acción de actualizar un proyecto

```
def update
  @proyecto = Proyecto.find(params[:id])

  respond_to do |format|
    if @proyecto.update_attributes(params[:proyecto])
      format.html { redirect_to @proyecto, notice: 'Proyecto was successfully updated.' }
      format.json { head :ok }
    else
      format.html { render action: "edit" }
      format.json { render json: @proyecto.errors, status: :unprocessable_entity }
    end
  end
end

# DELETE /proyectos/1
# DELETE /proyectos/1.json
```

Realiza el llamamiento al formulario para realizar la acción de eliminar un proyecto

```
def destroy
  @proyecto = Proyecto.find(params[:id])
  @proyecto.destroy

  respond_to do |format|
    format.html { redirect_to proyectos_url }
    format.json { head :ok }
  end
end
end
```

VISTAS

VISTA ADMINISTRADOR

Llamada formulario de administrador

```
<%= cabecera "Administrador"%>
<%= render "formlogin" %>
<%= pie%>
```

Gráfica formulario para ingreso de usuario y administrador para el ingreso, modificar, actualizar y eliminar información

```
<% if session[:iniciado]==nil %>
<h3>Iniciar sesion</h3>
<form method="get" action="/administrador/login/">
	<table class="adminform">
		<tr>
			<th>Usuario:</th>
			<td><input type="text" name="usuario" id="usuario" /></td>
		</tr>
		<tr>
			<th>Password:</th>
			<td><input type="Password" name="pass" id="pass" /></td>
		</tr>
		<tr>
			<td colspan="2"><input type="submit" value="Enviar" /></td>
		</tr>
	</table>
	<br>
</form>
<% else %>
<h3>Cerrar sesion</h3>
<form method="get" action="/administrador/logout/">
	<input type="submit" value="Cerrar sesion" />
</form>
<% end %>
```

Si el administrador es correcto o está incorrecto ingresa al siguiente formulario

```
<%= cabecera "Administrador"%>
<h3>Se ha inicado sesion</h3>
<%=@error%>
<%= pie %>
```

Si el administrador cierra sección ingresa al siguiente formulario

```
<%= cabecera "Administrador"%>
<h3>Se ha cerrado sesion</h3>
```

```
<%= pie %>
```

VISTA BOLETIN

Ingresa al formulario principal para decidir que tarea vamos a realiza es la vista index

```
<%= cabecera "Todos los boletines" %>
<table class="adminlist" style="width:100%">
  <tr>
        <th style="width:10px">#</th>
    <th>Nombre</th>
    <th style="width:20px">Disponible</th>
    <th style="width:20px">Acciones</th>
  </tr>

<%
        i = 1
        @boletins.each do |boletin|
                fila = "row0"
                if (i % 2) == 0
                        fila = "row1"
                end
%>
  <tr class="<%=fila%>">
        <td><%=i%></td>
    <td><%= boletin.nombre %></td>
    <td style="text-align: center"><%= boletin.disponible %></td>
    <td nowrap>
        <%= link_to 'Mostrar', boletin %>
        <%= link_to 'Editar', edit_boletin_path(boletin) %>
        <%= link_to 'Eliminar', boletin, confirm: 'Are you sure?', method: :delete %>
    </td>
  </tr>
<% end %>
</table>

<br />

<%= link_to 'Nuevo BoletÃn', new_boletin_path %>
<%= pie %>
```

Grafica todos los datos que se encuentra ingresados con sus respectivos links para poder realizar lo que el administrador necesita realizar

```
<%= cabecera "Mostrar BoletÃn"%>
<p id="notice"><%= notice %></p>

<p>
  <b>Nombre:</b>
  <%= @boletin.nombre %>
</p>

<p>
  <b>Contenido:</b>
  <%= raw @boletin.contenido %>
</p>

<p>
  <b>Disponible:</b>
  <%= @boletin.disponible %>
</p>


<%= link_to 'Editar', edit_boletin_path(@boletin) %> |
<%= link_to 'Ir a listar', boletins_path %>
<%= pie %>
```

Grafica el boletín seleccionado hacer creado, modificado o actualizado

```
<%= form_for(@boletin) do |f| %>
  <% if @boletin.errors.any? %>
    <div id="error_explanation">
      <h2><%= pluralize(@boletin.errors.count, "error") %> prohibited this boletin from being
saved:</h2>

      <ul>
      <% @boletin.errors.full_messages.each do |msg| %>
        <li><%= msg %></li>
      <% end %>
      </ul>
    </div>
  <% end %>

  <div class="field">
    <%= f.label :nombre %><br />
    <%= f.text_field :nombre %>
  </div>
  <div class="field">
```

```
    <%= f.label :contenido %><br />
    <%= f.text_area :contenido %>
  </div>
  <div class="field">
    <%= f.label :disponible %><br />
    <%= comboselect_sino "boletin",@boletin.disponible %>
  </div>
  <div class="actions">
    <%= f.submit "Guardar"%>
  </div>
<% end %>
<%= tinymce_init%>
```

Enlace para crear un nuevo boletín

```
<%= cabecera "Nuebo BoletÃn"%>
<%= render 'form' %>

<%= link_to 'Ir a listar', boletins_path %>
<%= pie %>
```

Enlace para editar un nuevo boletín

```
<%= cabecera "Editar BoletÃn"%>

<%= render 'form' %>

<%= link_to 'Ir a mostrar', @boletin %> |
<%= link_to 'Ir a listar', boletins_path %>
<%= pie %>
```

VISTA BOLETIN

Ingresa al formulario principal para decidir que tarea vamos a realiza es la vista index

```
<%= cabecera "Todos los contenidos"%>
<table class="adminlist" style="width:100%">
  <tr>
        <th style="width:10px">#</th>
    <th>Nombre</th>
    <th style="width:20px">Disponible</th>
    <th style="width:20px">Acciones</th>
```

```
  </tr>

<%
        i = 1
        @contenidos.each do |contenido|
                fila = "row0"
                if (i % 2) == 0
                        fila = "row1"
                end
%>
  <tr class="<%=fila%>">
        <td><%=i%></td>
    <td><%= contenido.nombre %></td>
    <td style="text-align: center"><%= contenido.disponible %></td>
    <td nowrap>
        <%= link_to "Mostrar", contenido %>
        <%= link_to 'Editar', edit_contenido_path(contenido) %>
        <%= link_to 'Eliminar', contenido, confirm: 'EstÃ¡ seguro?', method: :delete %>
    </td>
  </tr>
<%
                i = i + 1
        end
%>

</table>

<br />

<%= link_to 'Nuevo Contenido', new_contenido_path %>
<%= pie %>
```

Gráfica todos los datos que se encuentra ingresados con sus respectivos links para poder realizar lo que el administrador necesita realizar

```
<%= cabecera "Mostrar contenido"%>
<p id="notice"><%= notice %></p>

<p>
  <b>Nombre:</b>
  <%= @contenido.nombre %>
</p>

<p>
```

```
  <b>Contenido:</b>
  <%= raw @contenido.contenido %>
</p>

<p>
  <b>Disponible:</b>
  <%= @contenido.disponible %>
</p>
<%= link_to 'Ir a Editar', edit_contenido_path(@contenido) %> |
<%= link_to 'Ir a Listar', contenidos_path %>
<%= pie %>
```

Grafica el boletín seleccionado hacer creado, modificado o actualizado

```
<%= form_for(@contenido) do |f| %>
  <% if @contenido.errors.any? %>
    <div id="error_explanation">
      <h2><%= pluralize(@contenido.errors.count, "error") %> prohibited this contenido from
being saved:</h2>

      <ul>
      <% @contenido.errors.full_messages.each do |msg| %>
        <li><%= msg %></li>
      <% end %>
      </ul>
    </div>
  <% end %>

  <div class="field">
    <%= f.label :nombre %><br />
    <%= f.text_field :nombre %>
  </div>
  <div class="field">
    <%= f.label :contenido %><br />
    <%= f.text_area :contenido,:rows => 20, :cols => 90  %>
  </div>
  <div class="field">
    <%= f.label :disponible %><br />
    <%= comboselect_sino "contenido",@contenido.disponible %>
  </div>
  <div class="actions">
    <%= f.submit "Guardar" %>
  </div>
<% end %>
```

```
<%= tinymce_init%>

Enlace para crear un nuevo contenido
<%= cabecera "Nuevo Contenido"%>
<%= render 'form' %>
<%= link_to 'Ir a Listar', contenidos_path %>
<%= pie%>

Enlace para editar un nuevo contenido
<%= cabecera "Editando contenido"%>
<%= render 'form' %>
<%= link_to 'Ir a Mostrar', @contenido %> |
<%= link_to 'Ir a Listar', contenidos_path %>
<%= pie%>
```

VISTA EVENTOS

Ingresa al formulario principal para decidir que tarea vamos a realiza es la vista index

```
<%= cabecera "Todos los eventos"%>

<table class="adminlist">
  <tr>
        <th style="width:10px">#</th>
    <th>Nombre</th>
    <th style="width:20px">Fecha</th>
    <th style="width:20px">Disponible</th>
    <th style="width:20px">Acciones</th>
  </tr>

<%
        i = 1
        @eventos.each do |evento|
                fila = "row0"
                if (i % 2) == 0
                        fila = "row1"
                end
%>
  <tr class="<%=fila%>">
        <td><%=i%></td>
```

```
    <td><%= evento.nombre %></td>
    <td><%= evento.fecha %></td>
    <td style="text-align: center"><%= evento.disponible %></td>
    <td nowrap>
          <%= link_to 'Mostrar', evento %>
          <%= link_to 'Editar', edit_evento_path(evento) %>
          <%= link_to 'Eliminar', evento, confirm: 'Are you sure?', method: :delete %>
    </td>
  </tr>
<% end %>
</table>

<br />

<%= link_to 'Nuevo Evento', new_evento_path %>
<%= pie %>
```

Grafica todos los datos que se encuentra ingresados con sus respectivos links para poder realizar lo que el administrador necesita realizar

```
<%= cabecera "Mostrar evento"%>
<p id="notice"><%= notice %></p>

<p>
  <b>Nombre:</b>
  <%= @evento.nombre %>
</p>

<p>
  <b>Fecha:</b>
  <%= @evento.fecha %>
</p>

<p>
  <b>Contenido:</b>
  <%= raw @evento.contenido %>
</p>

<p>
  <b>Disponible:</b>
  <%= @evento.disponible %>
</p>
```

```
<%= link_to 'Editar', edit_evento_path(@evento) %> |
<%= link_to 'Ir a listar', eventos_path %>
<%= pie %>
```

Grafica el boletín seleccionado hacer creado, modificado o actualizado

```
<%= form_for(@evento) do |f| %>
  <% if @evento.errors.any? %>
    <div id="error_explanation">
      <h2><%= pluralize(@evento.errors.count, "error") %> prohibited this evento from being
saved:</h2>

      <ul>
      <% @evento.errors.full_messages.each do |msg| %>
        <li><%= msg %></li>
      <% end %>
      </ul>
    </div>
  <% end %>

  <div class="field">
    <%= f.label :nombre %><br />
    <%= f.text_field :nombre,:size=>50 %>
  </div>
  <div class="field">
    <%= f.label :fecha %><br />
    <%= f.date_select :fecha %>
  </div>
  <div class="field">
    <%= f.label :contenido %><br />
    <%= f.text_area :contenido %>
  </div>
  <div class="field">
    <%= f.label :disponible %><br />
    <%= comboselect_sino "evento",@evento.disponible%>
  </div>
  <div class="actions">
    <%= f.submit "Guardar"%>
  </div>
<% end %>
<%= tinymce_init%>
```

Enlace para crear un nuevo evento

```
<%= cabecera "Nuevo Evento"%>

<%= render 'form' %>

<%= link_to 'Ir a listar', eventos_path %>

<%= pie %>
```

Enlace para editar un nuevo evento

```
<%= cabecera "Editar evento"%>

<%= render 'form' %>

<%= link_to 'Mostrar', @evento %> |
<%= link_to 'Ir a listar', eventos_path %>

<%= pie%>
```

VISTA FRONTEND

Realiza la llamada principal de todos los datos ingresados de la pagina web para su presentación

Boletín

```
<h1>Frontend#boletin_show</h1>
<p>Find me in app/views/frontend/boletin_show.html.erb</p>

<% @boletins.each do |boletin| %>
        <%= cabecera boletin.nombre %>
        <strong>Fecha:</strong> <%= boletin.created_at%><br>
        <%= raw boletin.contenido %>
   <%= pie %>
<% end %>
```

Eventos

```
<%      @eventos.each do |evento|%>
        <%= cabecera evento.nombre%>
   <strong>Fecha:</strong> <%= evento.fecha %><br>
   <%= raw evento.contenido%>
   <%= pie%>
<% end %>
```

Galería

```
<%= cabecera "GalerÃa de ImÃ¡genes"%>
<script type="text/javascript" src="swfobject.js"></script>
<script type="text/javascript">
var flashvars = {};
var params = {};
params.scale = "noscale";
params.salign = "tl";
params.wmode = "transparent";
params.allowScriptAccess = "always";
params.allowFullScreen = "true";
var attributes = {};
swfobject.embedSWF("DockGalleryFX.swf", "DivDockGalleryFX", "100%", "400", "9.0.0",
false, flashvars, params, attributes);
</script>
<div id="DivDockGalleryFX"></div>
<%= pie %>
```

Miembros

```
<%= cabecera @miembro.nombre%>
<%= raw @miembro.contenido%>
<%= pie%>
<%=cabecera "Miembros de la MCRM"%>

<h2>Los miembros del MCRM son:</h2>
<ul>
        <li>Gobierno Municipal de Urcuqui</li>
        <li>Gobierno Municipal de Mira</li>
        <li>Gobierno Municipal de Espejo</li>
        <li>Gobierno Municipal de Bolivar</li>
        <li>Gobierno Municipal de Pimampiro</li>
</ul>
<%=pie%>
```

Noticias

```
<%= cabecera @noticium.nombre %>
<%= raw @noticium.introduccion%>
<%= raw @noticium.contenido%>
<%= pie %>

<% @noticia.each do |noticium| %>
        <%= cabecera noticium.nombre%>
        <%= raw noticium.introduccion%>
   <%= leermas "noticia_show",noticium.id%>
   <%= pie%>
  </tr>
<% end %>
```

Periodos

```
<%     @periodos.each do |periodo| %>
  <%= cabecera periodo.nombre%>
  <%= raw periodo.descripcion%>
  <%=pie%>
<% end %>
```

Proyectos

```
<%= cabecera @proyecto.nombre%>
<p>
  <b>Estado:</b>
  <%= @proyecto.estado %>
</p>
  <%= raw @proyecto.contenido %>
<%= pie%>

<% @proyectos.each do |proyecto| %>
        <%= cabecera proyecto.nombre%>
                <strong>Estado: </strong><br><%= proyecto.estado%><%= leermas
"proyecto_show",proyecto.id%>
        <%= pie%>
<% end %>
```

Index

```
<%= cabecera @contenido.nombre%>
```

```
<%= raw @contenido.contenido%>
<%= pie%>
```

VISTA GALERIA

Ingresa al formulario principal para decidir que tarea vamos a realiza es la vista index

```
<h1>Listing galeria</h1>

<table>
  <tr>
    <th>Nombre</th>
    <th>Descripcion</th>
    <th>Imagen</th>
    <th>Disponible</th>
    <th></th>
    <th></th>
    <th></th>
  </tr>

<% @galeria.each do |galerium| %>
  <tr>
    <td><%= galerium.nombre %></td>
    <td><%= galerium.descripcion %></td>
    <td><%= galerium.imagen %></td>
    <td><%= galerium.disponible %></td>
    <td><%= link_to 'Show', galerium %></td>
    <td><%= link_to 'Edit', edit_galerium_path(galerium) %></td>
    <td><%= link_to 'Destroy', galerium, confirm: 'Are you sure?', method: :delete %></td>
  </tr>
<% end %>
</table>

<br />

<%= link_to 'New Galerium', new_galerium_path %>
```

Grafica todos los datos que se encuentra ingresados con sus respectivos links para poder realizar lo que el administrador necesita realizar

```
<p id="notice"><%= notice %></p>
```

```
<p>
  <b>Nombre:</b>
  <%= @galerium.nombre %>
</p>

<p>
  <b>Descripcion:</b>
  <%= @galerium.descripcion %>
</p>

<p>
  <b>Imagen:</b>
  <%= @galerium.imagen %>
</p>

<p>
  <b>Disponible:</b>
  <%= @galerium.disponible %>
</p>


<%= link_to 'Edit', edit_galerium_path(@galerium) %> |
<%= link_to 'Back', galeria_path %>
```

Grafica el boletín seleccionado hacer creado, modificado o actualizado

```
<%= form_for(@galerium) do |f| %>
  <% if @galerium.errors.any? %>
    <div id="error_explanation">
      <h2><%= pluralize(@galerium.errors.count, "error") %> prohibited this galerium from
being saved:</h2>

      <ul>
      <% @galerium.errors.full_messages.each do |msg| %>
        <li><%= msg %></li>
      <% end %>
      </ul>
    </div>
  <% end %>

  <div class="field">
    <%= f.label :nombre %><br />
    <%= f.text_field :nombre %>
```

```
  </div>
  <div class="field">
    <%= f.label :descripcion %><br />
    <%= f.text_area :descripcion %>
  </div>
  <div class="field">
    <%= f.label :imagen %><br />
    <%= f.text_field :imagen %>
  </div>
  <div class="field">
    <%= f.label :disponible %><br />
    <%= f.text_field :disponible %>
  </div>
  <div class="actions">
    <%= f.submit %>
  </div>
<% end %>
```

Enlace para crear una nueva galería

```
<h1>New galerium</h1>

<%= render 'form' %>

<%= link_to 'Back', galeria_path %>
```

Enlace para editar una nueva galería

```
<h1>Editing galerium</h1>

<%= render 'form' %>

<%= link_to 'Show', @galerium %> |
<%= link_to 'Back', galeria_path %>
```

VISTA INICIO

Inicio de la página web

```
<%= cabecera @contenido.nombre%>
<%= raw @contenido.contenido%>
<%= pie%>
```

VISTA LOYOUTS

Se encarga de la graficación de toda la pagina web

```
<!DOCTYPE html>
<html>
<head>
  <title><%= SITE_NAME%></title>
  <%= stylesheet_link_tag    "application" %>
 <script type="text/javascript" src="/assets/jquery.js">
 </script>
  <%= stylesheet_link_tag    "style" %>
  <%= javascript_include_tag "script" %>
  <%= javascript_include_tag "application" %>
  <%= csrf_meta_tags %>
</head>
<body>
<div id="art-page-background-glare">
       <div id="art-page-background-glare-image">
    <div id="art-main">
       <div class="art-sheet">
          <div class="art-sheet-tl"></div>
          <div class="art-sheet-tr"></div>
          <div class="art-sheet-bl"></div>
          <div class="art-sheet-br"></div>
          <div class="art-sheet-tc"></div>
          <div class="art-sheet-bc"></div>
          <div class="art-sheet-cl"></div>
          <div class="art-sheet-cr"></div>
          <div class="art-sheet-cc"></div>
          <div class="art-sheet-body">
             <div class="art-nav">
                  <div class="l"></div>
                  <div class="r"></div>
                  <ul class="art-menu">
                           <li>
                                    <a href="/inicio/index" ><span class="l"></span><span
class="r"></span><span class="t">Inicio</span></a>
                           </li>
                           <li>
                                    <a href="/frontend/boletines"><span class="l"></span><span
class="r"></span><span class="t">Boletines</span></a>
```

```
                    </li>
                    <li>
                        <a href="/frontend/noticias"><span class="l"></span><span
class="r"></span><span class="t">Noticias</span></a>
                    </li>
                    <li>
                        <a href="/frontend/eventos"><span class="l"></span><span
class="r"></span><span class="t">Eventos</span></a>
                    </li>
                    <% if session[:iniciado]!=nil %>
                    <li>
                        <a href="#"><span class="l"></span><span
class="r"></span><span class="t">AdministraciÃ³n</span></a>
                        <ul>
                            <li><a href="/contenidos">Contenidos</a></li>
                            <li><a href="/boletins">Boletines</a></li>
                            <li><a href="/eventos">Eventos</a></li>
                            <li><a href="/miembros">Miembros</a></li>
                            <li><a href="/periodos">Periodos</a></li>
                            <li><a href="/noticia">Noticias</a></li>
                            <li><a href="/proyectos">Proyectos</a></li>
                        </ul>
                    </li>
                    <li>
                        <a href="/administrador/logout"><span class="l"></span><span
class="r"></span><span class="t">Cerrar sesiÃ³n</span></a>
                    </li>
                    <% end %>
                    <li>
                        <a href="#"><span class="l"></span><span
class="r"></span><span class="t">About</span></a>
                    </li>
                </ul>
            </div>
            <div class="art-header">
               <div class="art-header-center">
                  <div class="art-header-jpeg"></div>
               </div>
               <div class="art-logo">
                <h1 id="name-text" class="art-logo-name"><a href="#">TÃtulo</a></h1>
                <h2 id="slogan-text" class="art-logo-text">Texto del Eslogan</h2>
               </div>
            </div>
            <div class="art-content-layout">
```

```
            <div class="art-content-layout-row">
              <div class="art-layout-cell art-content">
<%= yield %>


               <div class="cleared"></div>
              </div>
              <div class="art-layout-cell art-sidebar1">
               <div class="art-vmenublock">
                  <div class="art-vmenublock-body">
                        <div class="art-vmenublockcontent">
                           <div class="art-vmenublockcontent-body">
                                 <ul class="art-vmenu">
                                  <li class="active">
                                       <a href="/frontend/periodos/"><span
class="l"></span><span class="r"></span><span class="t">Dirigentes MCRM</span></a>
                                  </li>
                                  <li>
                                       <a href="/frontend/proyectos/"><span
class="l"></span><span class="r"></span><span class="t">Oferta Institucional</span></a>
                                  </li>
                                  <li>
                                       <a href="/frontend/miembros/"><span
class="l"></span><span class="r"></span><span class="t">Miembros</span></a>
                                       <ul>
                                            <li><a
href="/frontend/miembro_show/?id=1">Gob.Muni. de Urcuqui</a></li>
                                            <li><a
href="/frontend/miembro_show/?id=2">Gob.Muni. de Mira</a></li>
                                            <li><a
href="/frontend/miembro_show/?id=3">Gob.Muni. de Espejo</a></li>
                                            <li><a
href="/frontend/miembro_show/?id=4">Gob.Muni. de BolÃvar</a></li>
                                            <li><a
href="/frontend/miembro_show/?id=5">Gob.Muni. de Pimampiro</a></li>

                                       </ul>
                                  </li>
                                 </ul>

                                  <div class="cleared"></div>
                           </div>
                        </div>
                     <div class="cleared"></div>
```

```
                    </div>
                  </div>

                  <div class="art-block">
                    <div class="art-block-body">
                        <div class="art-blockheader">
                          <h3 class="t">Contact Info</h3>
                        </div>
                        <div class="art-blockcontent">
                          <div class="art-blockcontent-body">
                              <div>
                                  <%= image_tag
"/assets/images/contact.jpg",:width=>95 %>
                                <br />
                                <b>Company Co.</b><br />
                                Las Vegas, NV 12345<br />
                                Email: <a
href="mailto:info@company.com">info@company.com</a><br />
                                <br />
                                Phone: (123) 456-7890 <br />
                                Fax: (123) 456-7890
                                </div>

                                 <div class="cleared"></div>
                          </div>
                        </div>
                            <div class="cleared"></div>
                    </div>
                  </div>
                  <div class="cleared"></div>
                </div>
              </div>
            </div>
            <div class="cleared"></div>
            <div class="art-footer">
              <div class="art-footer-t"></div>
              <div class="art-footer-l"></div>
              <div class="art-footer-b"></div>
              <div class="art-footer-r"></div>
              <div class="art-footer-body">
                  <a href="#" class="art-rss-tag-icon" title="RSS"></a>
                <div class="art-footer-text">
                    <p><a href="#">Link1</a> | <a href="#">Link2</a> | <a
href="#">Link3</a></p><p>Copyright &amp;copy; 2011. All Rights Reserved.</p>
```

```
                </div>
                        <div class="cleared"></div>
                    </div>
                </div>
                    <div class="cleared"></div>
            </div>
        </div>
        <div class="cleared"></div>
        <p class="art-page-footer"><a href="http://www.artisteer.com/">Web Template</a>
created with Artisteer.</p>
    </div>
        </div>
    </div>

</body>
</html>
```

VISTA MIEMBROS

Ingresa al formulario principal para decidir que tarea vamos a realiza es la vista index

```
<%= cabecera "Listado de miembros"%>

<table class="adminlist" style="width:100%">
  <tr>
        <th style="width:10px">#</th>
    <th>Nombre</th>
    <th style="width:20px">Disponible</th>
    <th style="width:20px">Acciones</th>
  </tr>

<%
        i = 1
        @miembros.each do |miembro|
                fila = "row0"
                if (i % 2) == 0
                        fila = "row1"
                end
%>
  <tr class="<%=fila%>">
        <td><%=i%></td>
    <td><%= miembro.nombre %></td>
    <td style="text-align: center"><%= miembro.disponible %></td>
```

```
    <td nowrap>
          <%= link_to 'Mostrar', miembro %>
          <%= link_to 'Editar', edit_miembro_path(miembro) %>
          <%= link_to 'Eliminar', miembro, confirm: 'EstÃ¡ seguro?', method: :delete %>

    </td>
  </tr>
<% end %>
</table>

<br />

<%= link_to 'Nuevo Miembro', new_miembro_path %>
<%= pie %>
```

Grafica todos los datos que se encuentra ingresados con sus respectivos links para poder realizar lo que el administrador necesita realizar

```
<%= cabecera "Mostrar Miembro"%>
<p id="notice"><%= notice %></p>

<p>
  <b>Nombre:</b>
  <%= @miembro.nombre %>
</p>

<p>
  <b>Contenido:</b>
  <%= raw @miembro.contenido %>
</p>

<p>
  <b>Disponible:</b>
  <%= @miembro.disponible %>
</p>


<%= link_to 'Editar', edit_miembro_path(@miembro) %> |
<%= link_to 'Ir a listar', miembros_path %>
<%= pie %>
```

Grafica el boletín seleccionado hacer creado, modificado o actualizado

```
<%= form_for(@miembro) do |f| %>
  <% if @miembro.errors.any? %>
    <div id="error_explanation">
      <h2><%= pluralize(@miembro.errors.count, "error") %> prohibited this miembro from
being saved:</h2>

      <ul>
      <% @miembro.errors.full_messages.each do |msg| %>
        <li><%= msg %></li>
      <% end %>
      </ul>
    </div>
  <% end %>

  <div class="field">
    <%= f.label :nombre %><br />
    <%= f.text_field :nombre, :size=>50 %>
  </div>
  <div class="field">
    <%= f.label :contenido %><br />
    <%= f.text_area :contenido %>
  </div>
  <div class="field">
    <%= f.label :disponible %><br />
    <%= comboselect_sino "miembro",@miembro.disponible %>
  </div>
  <div class="actions">
    <%= f.submit "Guardar"%>
  </div>
<% end %>
<%= tinymce_init %>
```

Enlace para crear un nuevo miembro

```
<%= cabecera "Nuevo Miembro"%>

<%= render 'form' %>

<%= link_to 'Ir a listar', miembros_path %>

<%= pie %>
```

Enlace para editar un nuevo miembro

```
<%= cabecera "Editando miembro"%>

<%= render 'form' %>

<%= link_to 'Mostrar', @miembro %> |
<%= link_to 'Ir a listar', miembros_path %>

<%= pie %>
```

VISTA NOTICIAS

Ingresa al formulario principal para decidir que tarea vamos a realiza es la vista index

```
<%= cabecera "Todas las noticias"%>
<table class="adminlist" style="width:100%">
  <tr>
        <th style="width:10px">#</th>
    <th>Nombre</th>
    <th style="width:20px">Disponible</th>
    <th style="width:20px">Acciones</th>
  </tr>

<%
        i = 1
        @noticia.each do |noticium|
                fila = "row0"
                if (i % 2) == 0
                        fila = "row1"
                end
%>
  <tr class="<%=fila%>">
        <td><%=i%></td>
    <td><%= noticium.nombre %></td>
    <td style="text-align: center"><%= noticium.disponible %></td>
    <td nowrap>
        <%= link_to 'Mostrar', noticium %>
        <%= link_to 'Editar', edit_noticium_path(noticium) %>
        <%= link_to 'Eliminar', noticium, confirm: 'EstÃ¡ seguro?', method: :delete %>
```

```
    </td>
  </tr>
<%
        i = i+1
        end
%>
</table>

<br />

<%= link_to 'Nueva Noticia', new_noticium_path %>
<%= pie%>
```

Grafica todos los datos que se encuentra ingresados con sus respectivos links para poder realizar lo que el administrador necesita realizar

```
<%= cabecera "Mostrar Noticia"%>
<p id="notice"><%= notice %></p>

<p>
  <b>Nombre:</b>
  <%= @noticium.nombre %>
</p>

<p>
  <b>Introduccion:</b>
  <%= raw @noticium.introduccion %>
</p>

<p>
  <b>Contenido:</b>
  <%= raw @noticium.contenido %>
</p>

<p>
  <b>Disponible:</b>
  <%= @noticium.disponible %>
</p>


<%= link_to 'Editar', edit_noticium_path(@noticium) %> |
<%= link_to 'Ir a listar', noticia_path %>
<%= pie%>
```

Grafica el boletín seleccionado hacer creado, modificado o actualizado

```
<%= form_for(@noticium) do |f| %>
  <% if @noticium.errors.any? %>
    <div id="error_explanation">
      <h2><%= pluralize(@noticium.errors.count, "error") %> prohibited this noticium from
being saved:</h2>

      <ul>
      <% @noticium.errors.full_messages.each do |msg| %>
        <li><%= msg %></li>
      <% end %>
      </ul>
    </div>
  <% end %>

  <div class="field">
    <%= f.label :nombre %><br />
    <%= f.text_field :nombre,:size=>80 %>
  </div>
  <div class="field">
    <%= f.label :introduccion %><br />
    <%= f.text_area :introduccion %>
  </div>
  <div class="field">
    <%= f.label :contenido %><br />
    <%= f.text_area :contenido %>
  </div>
  <div class="field">
    <%= f.label :disponible %><br />
    <%= comboselect_sino "noticium",@noticium.disponible%>
  </div>
  <div class="actions">
    <%= f.submit "Guardar"%>
  </div>
<% end %>
<%= tinymce_init%>
```

Enlace para crear una nueva noticia

```
<h1>New noticium</h1>
```

```
<%= render 'form' %>

<%= link_to 'Back', noticia_path %>
```

Enlace para editar una nueva noticia

```
<%= cabecera "Editar Noticia"%>

<%= render 'form' %>

<%= link_to 'Ir a mostrar', @noticium %> |
<%= link_to 'Ir a listar', noticia_path %>

<%= pie %>
```

VISTA PERIODOS

Ingresa al formulario principal para decidir que tarea vamos a realiza es la vista index

```
<%= cabecera "Todos los periodos"%>

<table class="adminlist" style="width:100%">
  <tr>
        <th style="width:10px">#</th>
    <th>Nombre</th>
    <th style="width:20px">Disponible</th>
    <th style="width:20px">Acciones</th>
  </tr>


<%
        i = 1
        @periodos.each do |periodo|
                fila = "row0"
                if (i % 2) == 0
                        fila = "row1"
                end
%>
  <tr class="<%=fila%>">
        <td><%=i%></td>
    <td><%= periodo.nombre %></td>
    <td><%= periodo.disponible %></td>
```

```
    <td nowrap>
          <%= link_to 'Mostrar', periodo %>
          <%= link_to 'Editar', edit_periodo_path(periodo) %>
          <%= link_to 'Eliminar', periodo, confirm: 'EstÃ¡ seguro?', method: :delete %>
    </td>
  </tr>
<% end %>
</table>

<br />

<%= link_to 'Nuevo Periodo', new_periodo_path %>
<%= pie %>
```

Grafica todos los datos que se encuentra ingresados con sus respectivos links para poder realizar lo que el administrador necesita realizar

```
<%= cabecera "Mostrar Periodo"%>
<p id="notice"><%= notice %></p>

<p>
  <b>Nombre:</b>
  <%= @periodo.nombre %>
</p>

<p>
  <b>Descripcion:</b>
  <%= raw @periodo.descripcion %>
</p>

<p>
  <b>Disponible:</b>
  <%= @periodo.disponible %>
</p>


<%= link_to 'Editar', edit_periodo_path(@periodo) %> |
<%= link_to 'Ir a listar', periodos_path %>

<%= pie %>
```

Grafica el boletín seleccionado hacer creado, modificado o actualizado

```
<%= form_for(@periodo) do |f| %>
  <% if @periodo.errors.any? %>
    <div id="error_explanation">
      <h2><%= pluralize(@periodo.errors.count, "error") %> prohibited this periodo from being
saved:</h2>

      <ul>
      <% @periodo.errors.full_messages.each do |msg| %>
        <li><%= msg %></li>
      <% end %>
      </ul>
    </div>
  <% end %>

  <div class="field">
    <%= f.label :nombre %><br />
    <%= f.text_field :nombre %>
  </div>
  <div class="field">
    <%= f.label :descripcion %><br />
    <%= f.text_area :descripcion %>
  </div>
  <div class="field">
    <%= f.label :disponible %><br />
    <%= comboselect_sino "periodo",@periodo.disponible%>
  </div>
  <div class="actions">
    <%= f.submit "Guardar"%>
  </div>
<% end %>
<%= tinymce_init %>
```

Enlace para crear un nuevo periodo

```
<%= cabecera "Nuevo Periodo"%>

<%= render 'form' %>

<%= link_to 'Ir a listar', periodos_path %>

<%= pie %>
```

Enlace para editar un nuevo periodo

```
<%= cabecera "Editando Periodo"%>

<%= render 'form' %>

<%= link_to 'Mostrar', @periodo %> |
<%= link_to 'Ir a listar', periodos_path %>
<%= pie %>
```

VISTA PROYECTOS

Ingresa al formulario principal para decidir que tarea vamos a realiza es la vista index

```
<%= cabecera "Todos los contenidos"%>
<table class="adminlist" style="width:100%">
 <tr>
        <th style="width:10px">#</th>
  <th>Nombre</th>
  <th style="width:20px">Estado</th>
  <th style="width:20px">Disponible</th>
  <th style="width:20px">Acciones</th>
 </tr>

<%
        i = 1
        @proyectos.each do |proyecto|
                fila = "row0"
                if (i % 2) == 0
                        fila = "row1"
                end
%>
 <tr class="<%=fila%>">
        <td><%=i%></td>
  <td><%= proyecto.nombre %></td>
  <td style="text-align: center"><%= proyecto.estado %></td>
  <td style="text-align: center"><%= proyecto.disponible %></td>
  <td nowrap>
        <%= link_to 'Mostrar', proyecto %>
        <%= link_to 'Editar', edit_proyecto_path(proyecto) %>
        <%= link_to 'Eliminar', proyecto, confirm: 'Are you sure?', method: :delete %>
  </td>
 </tr>
<%
```

```
                i = i + 1
        end
%>
</table>

<br />

<%= link_to 'Nuevo Proyecto', new_proyecto_path %>
<%= pie%>
```

Grafica todos los datos que se encuentra ingresados con sus respectivos links para poder realizar lo que el administrador necesita realizar

```
<%= cabecera "Mostrar Proyecto"%>
<p id="notice"><%= notice %></p>

<p>
  <b>Nombre:</b>
  <%= @proyecto.nombre %>
</p>

<p>
  <b>Estado:</b>
  <%= @proyecto.estado %>
</p>

<p>
  <b>Disponible:</b>
  <%= @proyecto.disponible %>
</p>
<%= link_to 'Editar', edit_proyecto_path(@proyecto) %> |
<%= link_to 'Ir a listar', proyectos_path %>
<p>
  <b>Contenido:</b>
  <%= raw @proyecto.contenido %>
</p>

<%= pie%>
```

Grafica el boletín seleccionado hacer creado, modificado o actualizado

```
<%= form_for(@proyecto) do |f| %>
```

```
 <% if @proyecto.errors.any? %>
  <div id="error_explanation">
    <h2><%= pluralize(@proyecto.errors.count, "error") %> prohibited this proyecto from
being saved:</h2>

    <ul>
    <% @proyecto.errors.full_messages.each do |msg| %>
      <li><%= msg %></li>
    <% end %>
    </ul>
  </div>
 <% end %>

 <div class="field">
  <%= f.label :nombre %><br />
  <%= f.text_field :nombre,:size=>80 %>
 </div>
 <div class="field">
  <%= f.label :estado %><br />
  <%= comboselect_estado "proyecto",@proyecto.estado%>
 </div>
 <div class="field">
  <%= f.label :contenido %><br />
  <%= f.text_area :contenido %>
 </div>
 <div class="field">
  <%= f.label :disponible %><br />
  <%= comboselect_sino "proyecto",@proyecto.disponible%>
 </div>
 <div class="actions">
  <%= f.submit "Guardar"%>
 </div>
<% end %>
<%= tinymce_init%>
```

Enlace para crear un nuevo proyecto

```
<%= cabecera "Nuevo proyecto"%>

<%= render 'form' %>

<%= link_to 'Ir a listar', proyectos_path %>

<%= pie%>
```

Enlace para editar un nuevo proyecto

```
<%= cabecera "Editing proyecto"%>

<%= render 'form' %>

<%= link_to 'Mostrar', @proyecto %> |
<%= link_to 'Ir a listar', proyectos_path %>
<%= pie%>
```

MODELOS

La acción que realizan es la interpretación con la base de datos para poder ingresar, modifica, eliminar, de la base de datos

```
class Boletin < ActiveRecord::Base
end

class Contenido < ActiveRecord::Base
end

class Evento < ActiveRecord::Base
end

class Galerium < ActiveRecord::Base
end

class Miembro < ActiveRecord::Base
end

class Noticium < ActiveRecord::Base
end

class Periodo < ActiveRecord::Base
end

class Proyecto < ActiveRecord::Base
  has_many :directivas
end
```

```
class Usuario < ActiveRecord::Base
end
```

MANUAL DE USUARIO

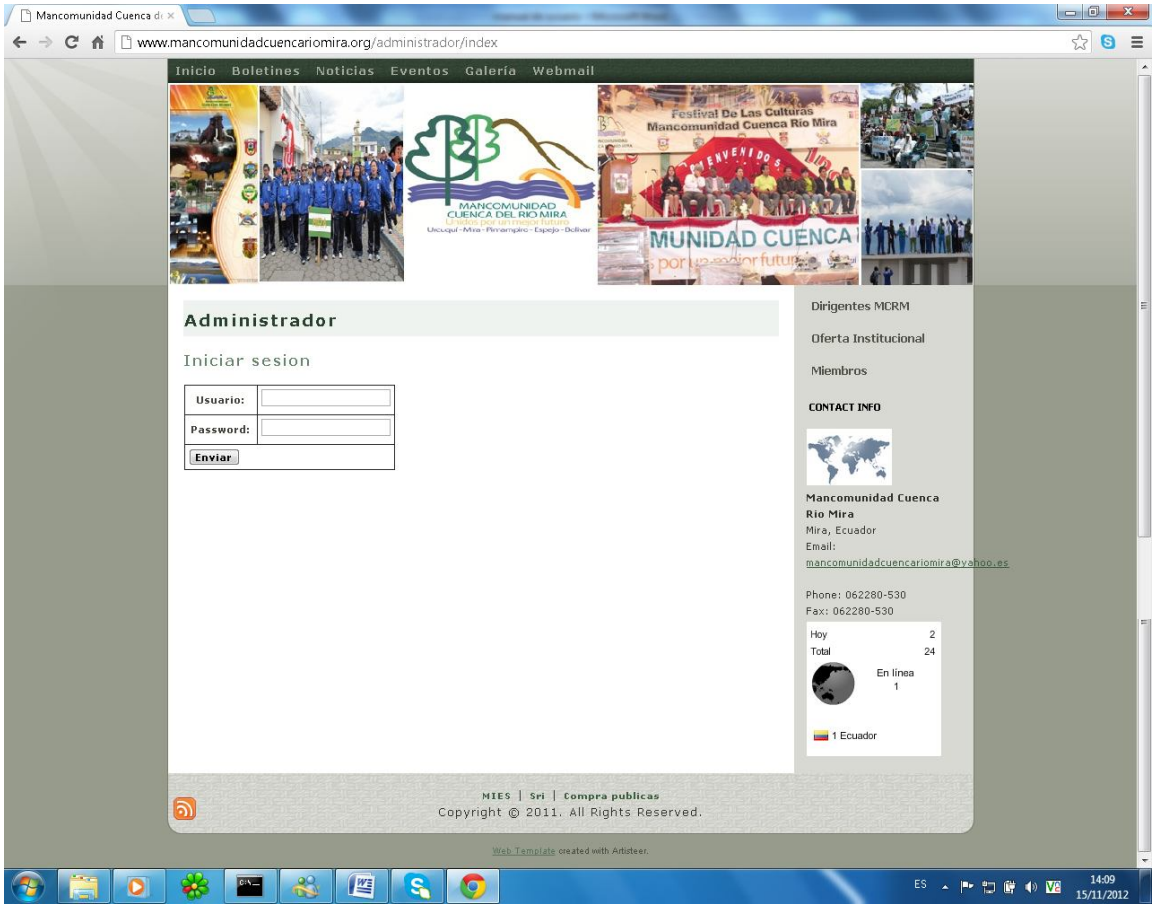
Ingreso al portal web de la Mancomunidad de la Cuenca del Río Mira

www.mancomunidadcuencariomira.org

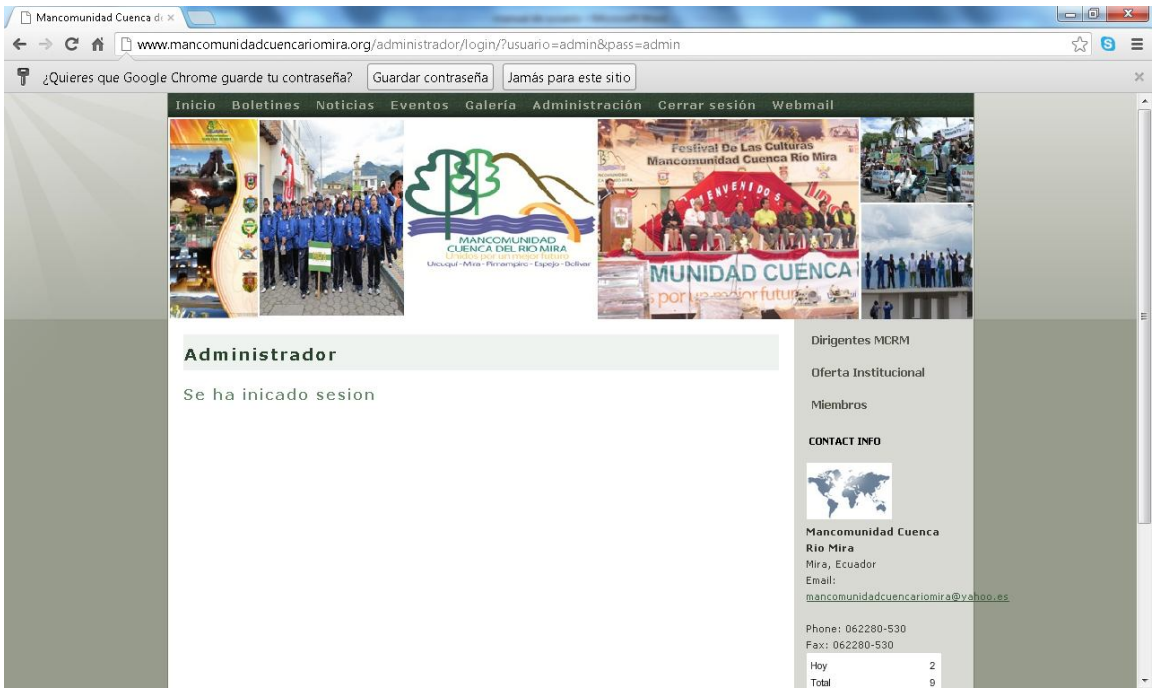
Podemos navegar por la web dentro de nuestra aplicación e ingresar a todos según la dependencia o los accesos a navegar dentro de nuestro sitio web podemos navegar por los boletines de información, noticias, eventos, galería, dirigente de la mancomunidad de la cuenca del rio mira, oferta institucional, y miembros que conforman la mancomunidad de la cuenca del rio mira. Además tenemos tres enlaces principales los cuales acceden en forma rápida y sencilla con solo un clic tenemos el enlace a la pagina del Mies, SRI y compras públicas.

La página tiene un contador de visitantes que acceden a nuestra página

Ingreso como administrador para ingresar, modificar, actualizar o eliminar información tenemos el usuario y el password



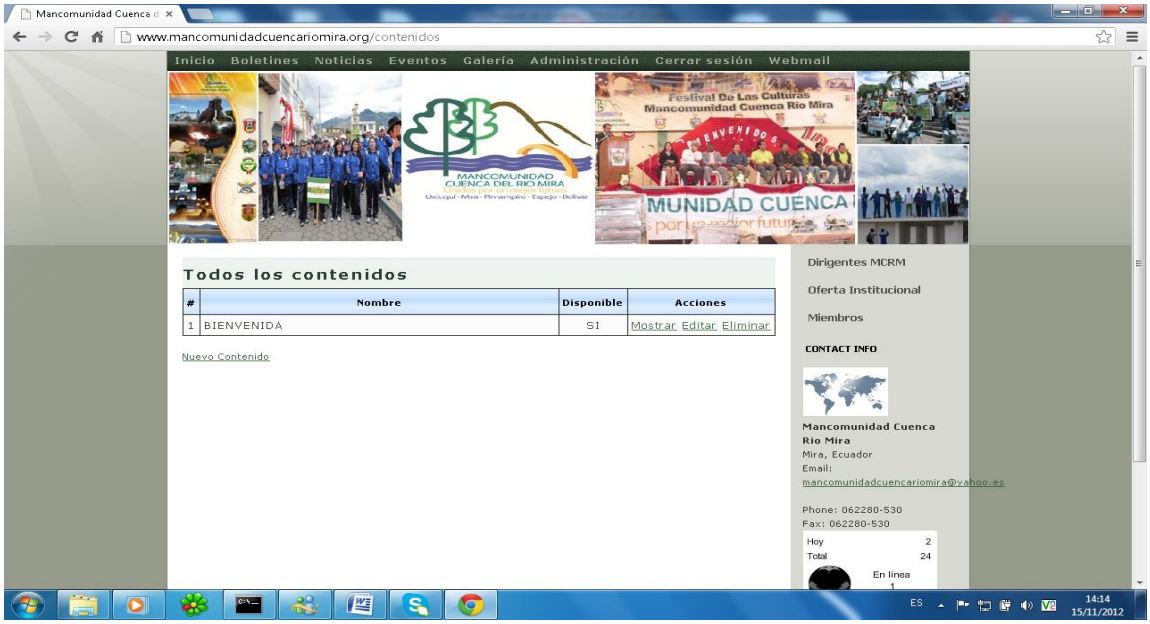
A continuación una vez ingresado el usuario y el password correctamente puedo hacer uso el menú de administración



Se habilita en el menú la opción de administrador para realizar acciones de ingreso, modificación, eliminación y nuevo

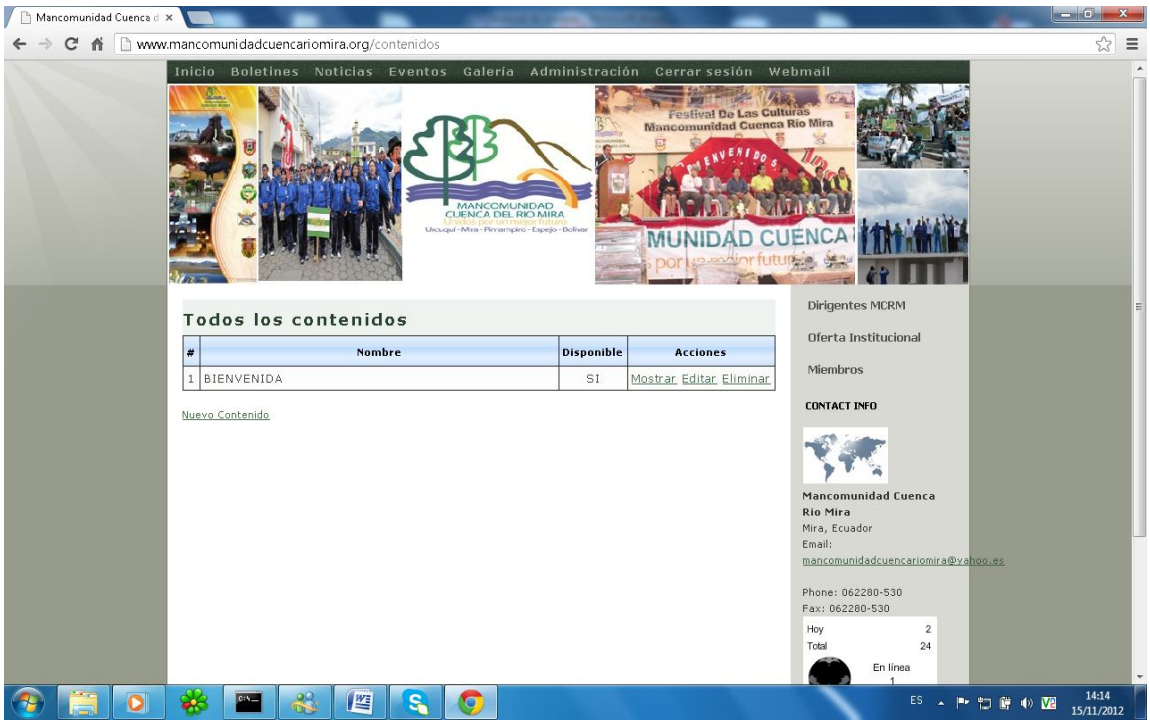
Editando contenido

Edita el contenido seleccionado y guardar para conservar cambios.



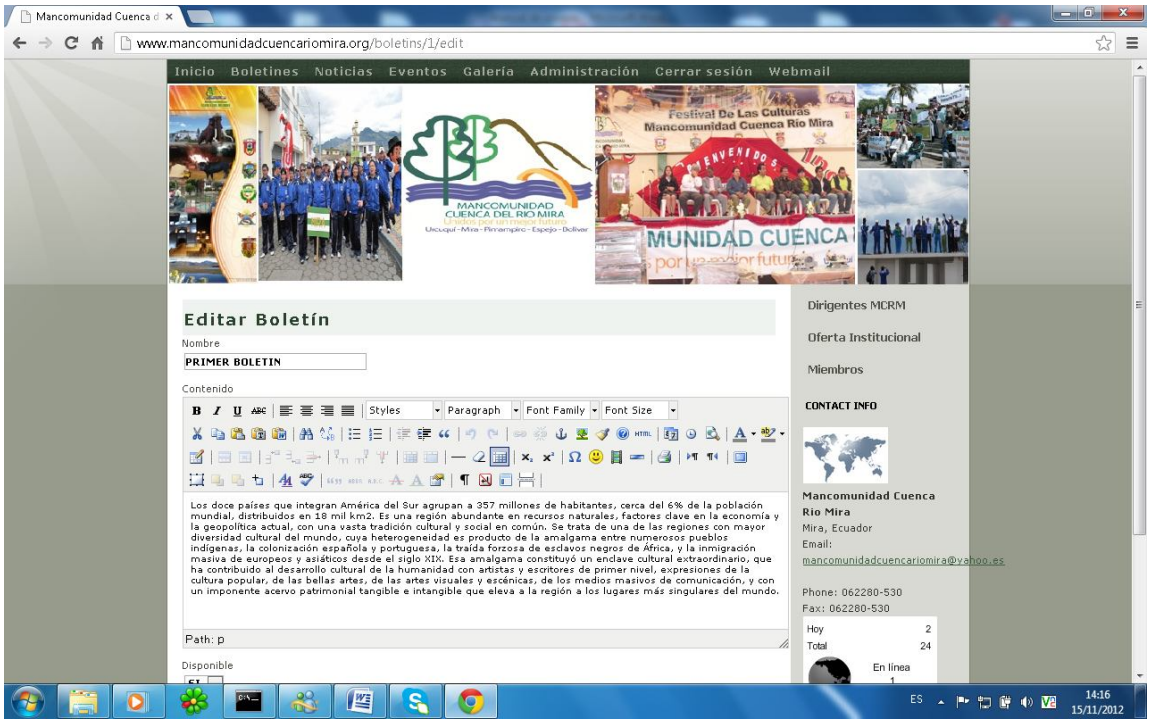
Boletines

Despliega la lista de boletines con las opciones de mostrar, editar y eliminar



Editar Boletín

Edita el boletín seleccionado y luego opción guardar.

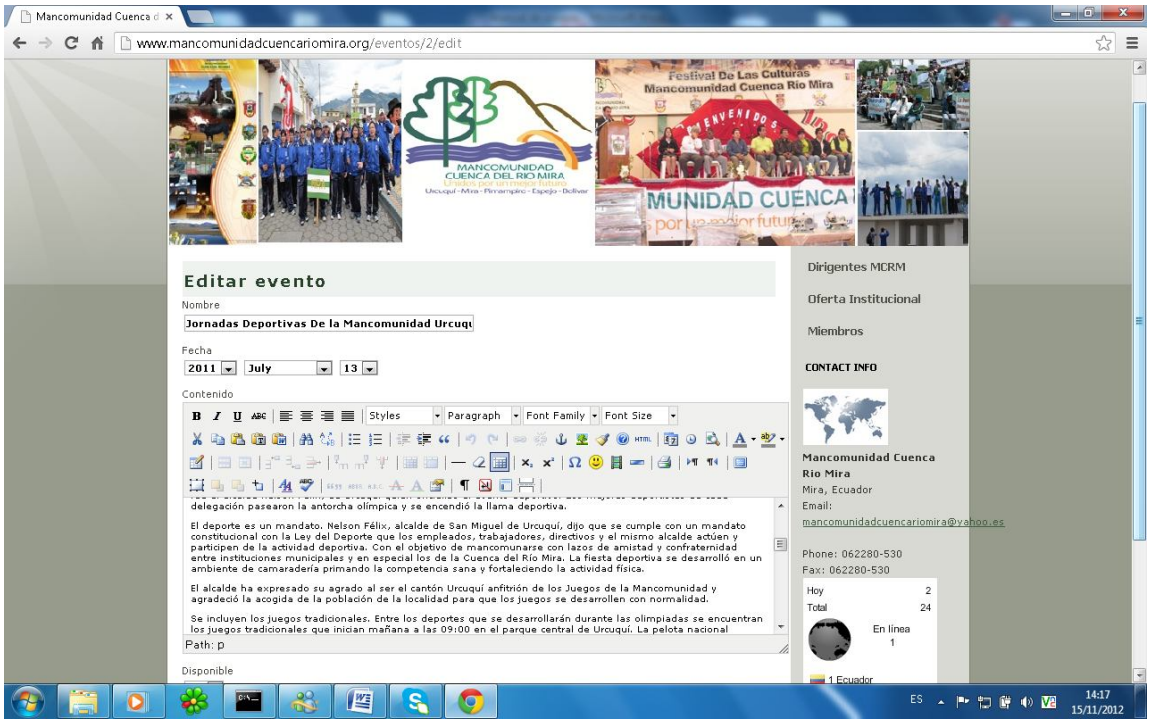


Eventos

Despliega la lista de eventos con las opciones de mostrar, editar y eliminar.

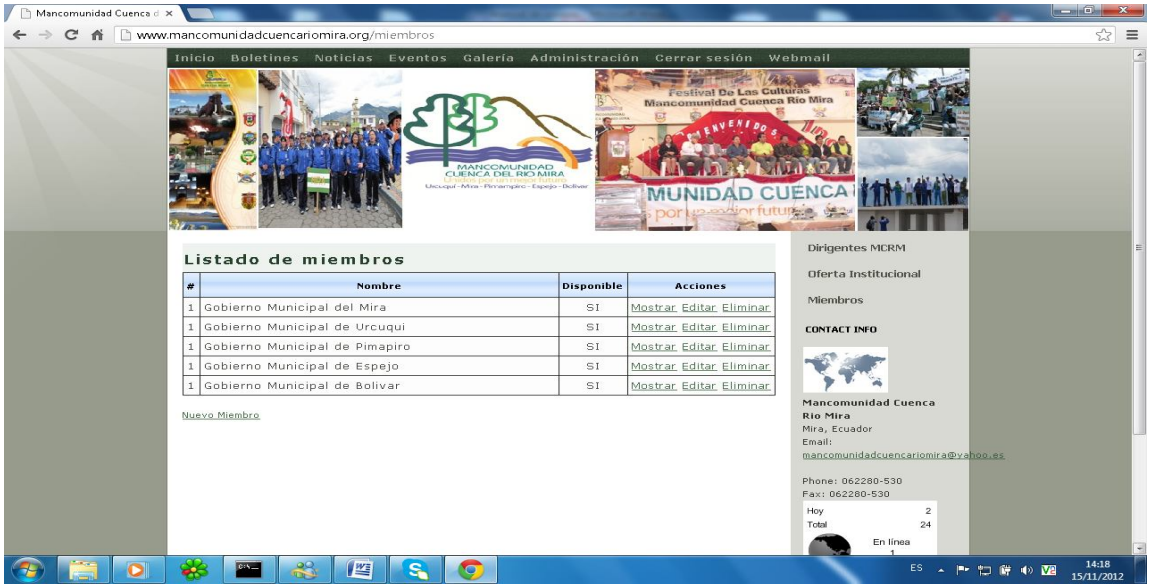
Editar evento

Edita el evento seleccionado, luego la opción guardar.



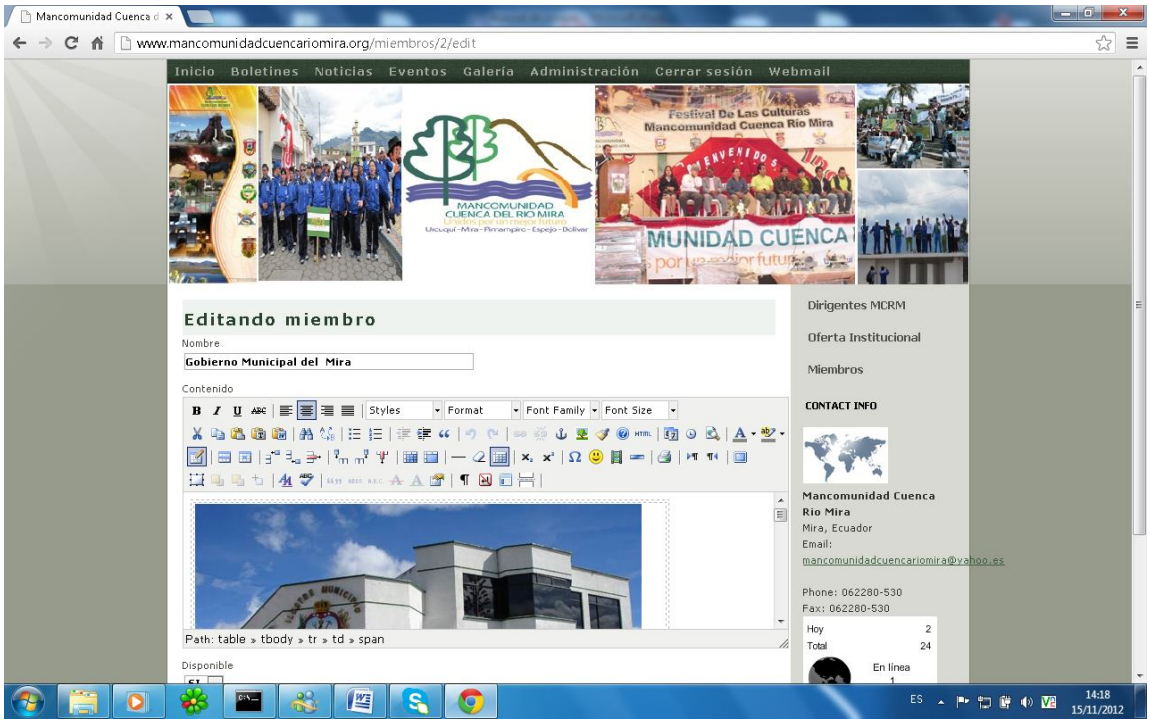
Miembros

Despliega la lista de los miembros con las opciones de mostrar, editar y eliminar.



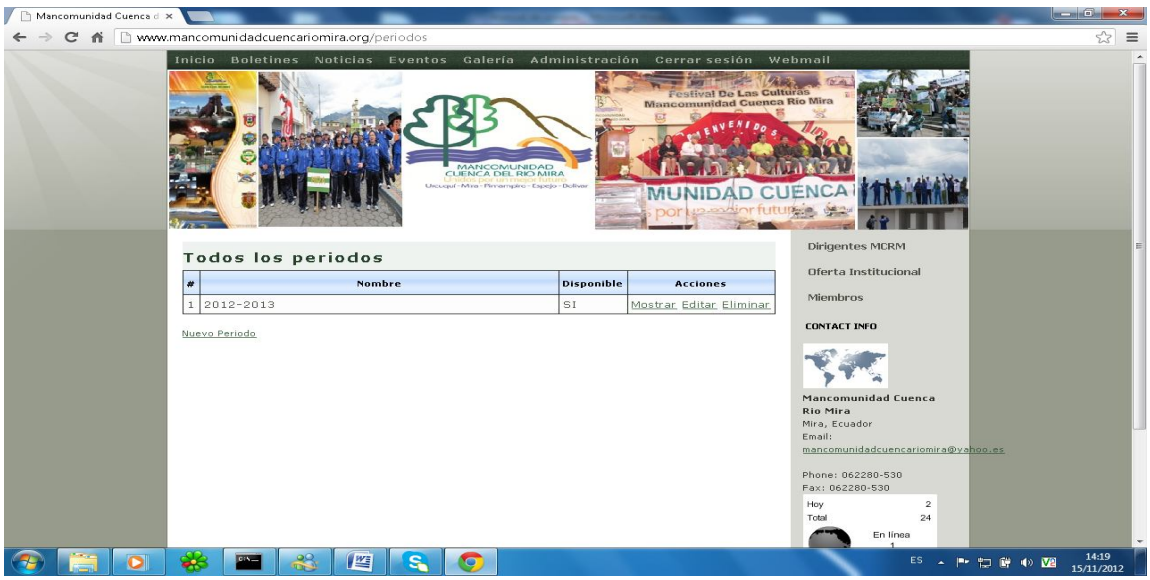
Editando miembro

Edita el miembro seleccionado, luego la opción guardar.



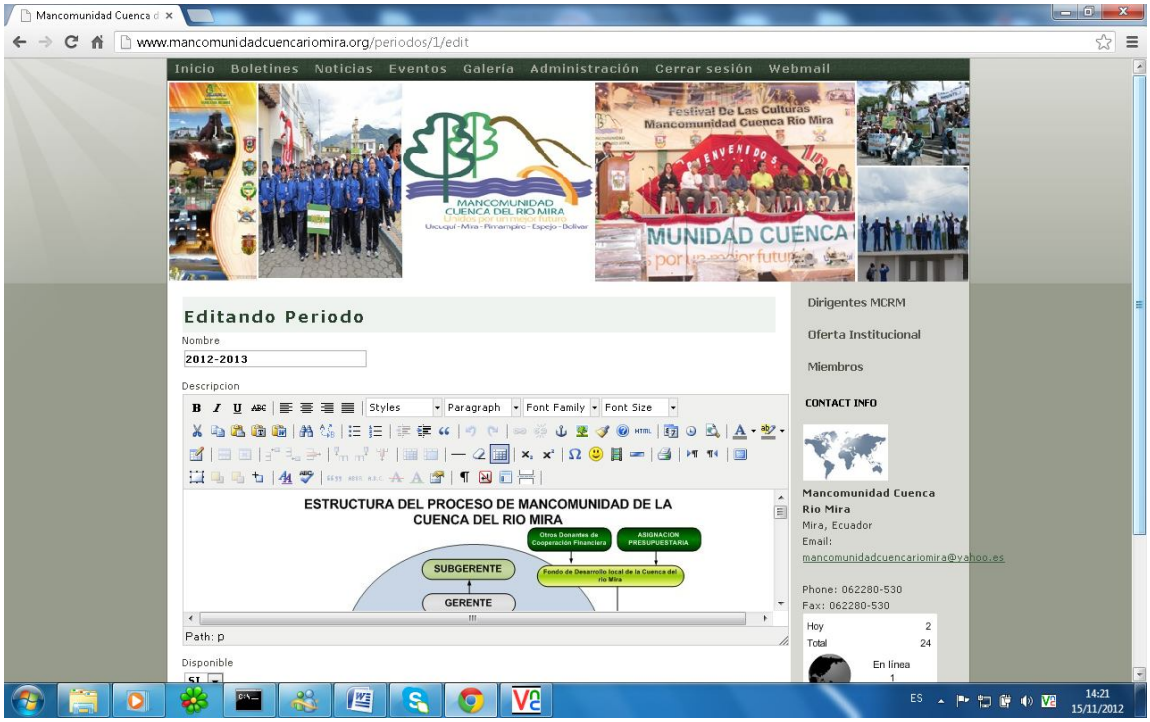
Periodos

Despliega la información de todos los períodos con las opciones de mostrar, editar y eliminar.



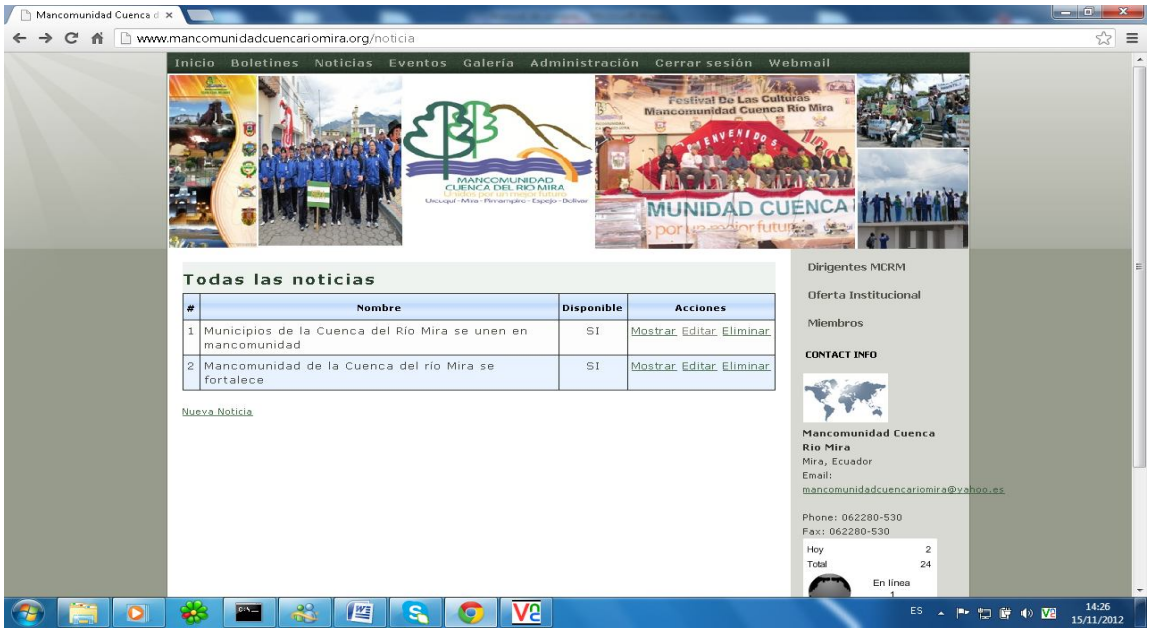
Editando Periodo

Edita el período seleccionado, luego la opción guardar.



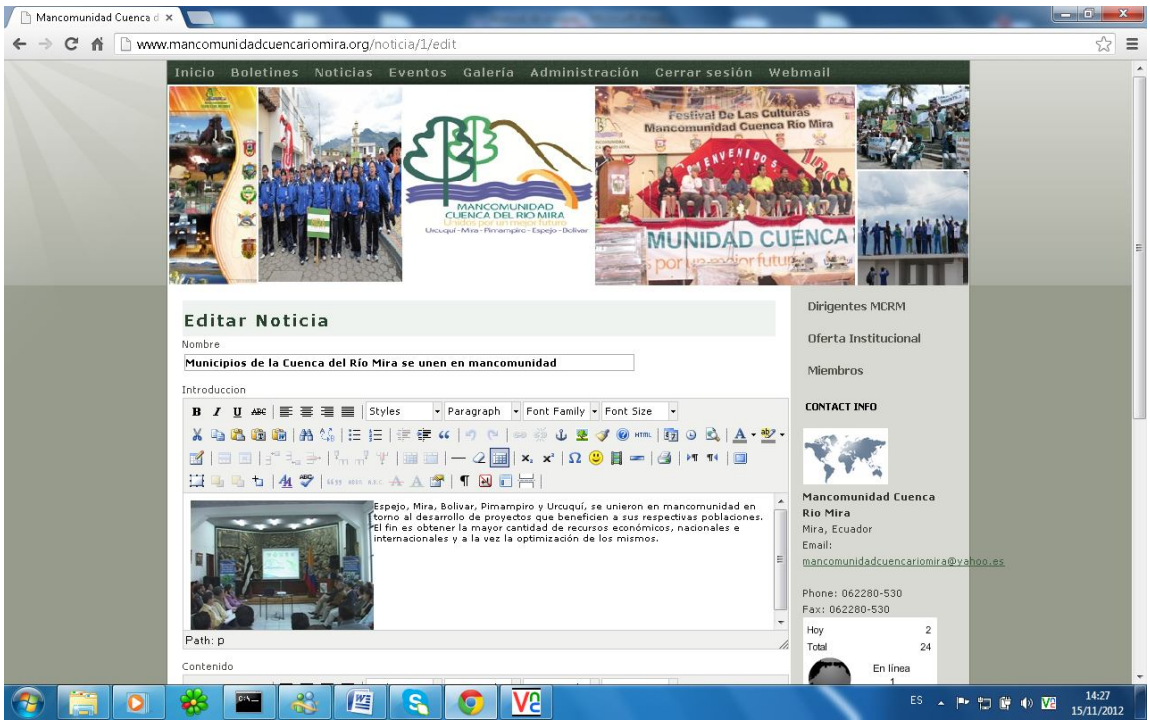
Noticias

Lista todas las noticias con las opciones de mostrar, editar y eliminar.



Editar Noticia

Edita la noticia seleccionada después de hacer los cambios la opción guardar.

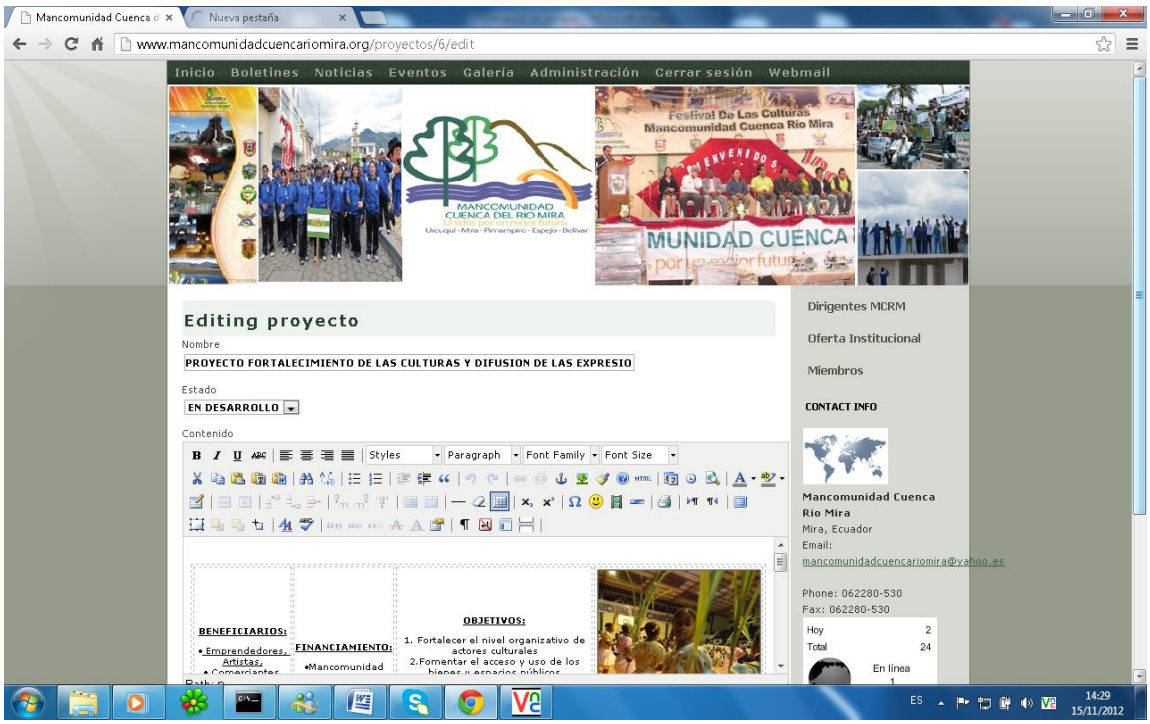


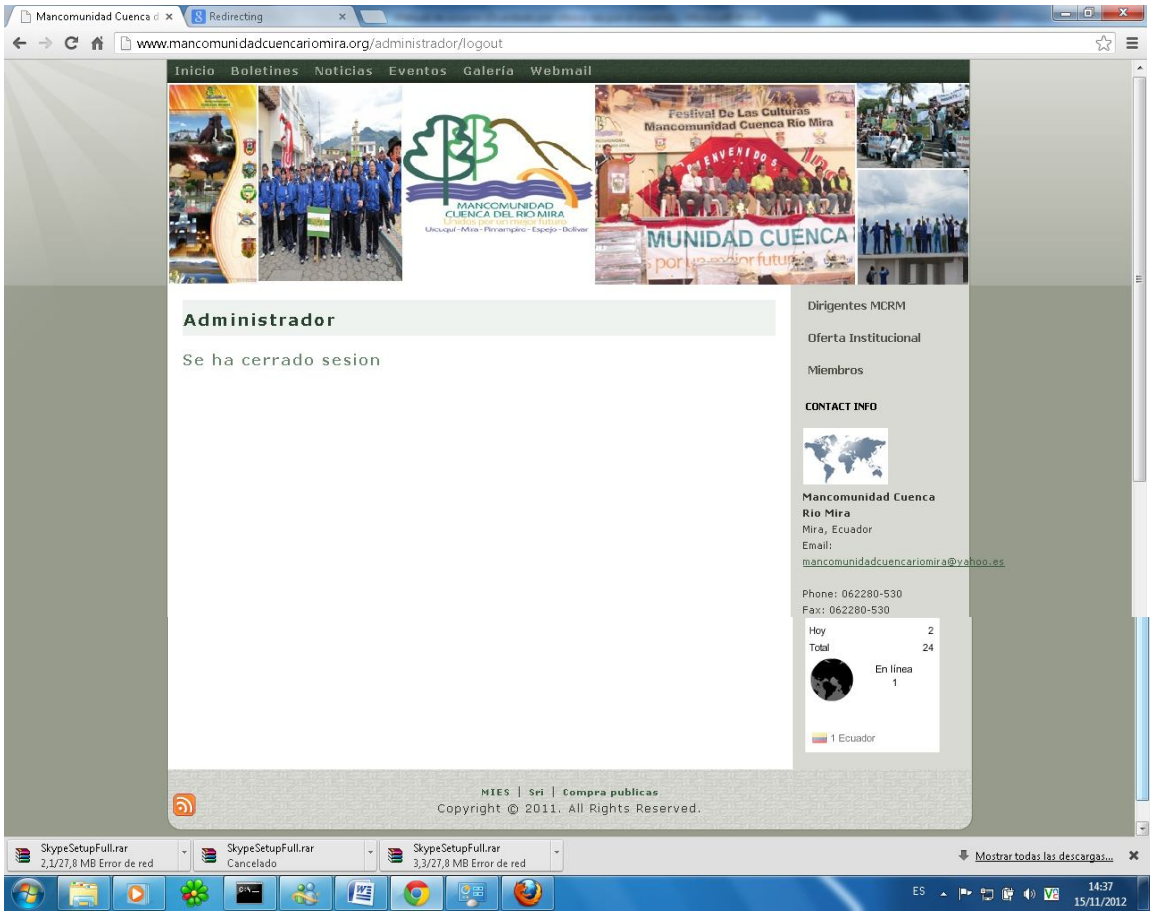
Proyectos

Presenta una lista de todos los proyectos con las opciones de mostrar, editar y eliminar.

Editando proyecto

Edita el proyecto seleccionado con las herramientas de edición luego la opción guardar para conservar los cambios.





Evaluación

La información ofrecida en el portal web de la mancomunidad de la cuenca del rio mira se encuentra evaluada y activa por parte de la dirigencia de dicho organismo para el uso y difusión de la información a todos sus miembros y a la comunidad de la web

Pruebas Finales

Realizada todas las pruebas pertinentes del caso el portal web se encuentra operativamente subida a la web para que toda la comunidad de la web pueda hacer uso de la difusión

informativa que tiene el organismo de la mancomunidad de la cuenca del rio mira así accediendo de la siguiente manera www.mancomunidadcuencariomira.org