



ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA EN SISTEMAS

**“ESTUDIO COMPARATIVO DE LAS TECNOLOGÍAS RUBY ON
RAILS Y MONO PARA EL DESARROLLO DE APLICACIONES
WEB. CASO PRÁCTICO: CONSEJO PROVINCIAL DE
CHIMBORAZO.”**

TESIS DE GRADO

**PREVIA LA OBTENCIÓN DEL TÍTULO DE:
INGENIERO EN SISTEMAS INFORMÁTICOS**

**PRESENTADO POR:
WILIAN XAVIER SÁNCHEZ LABRE**

**RIOBAMBA – ECUADOR
2008**

A la Escuela Superior Politécnica de Chimborazo por ser la Institución que cristalizó mi formación, a sus Autoridades y Docentes, en especial a la Ing. Ivonne Rodríguez Directora de Tesis y al Ing. Wladimir Castro Miembro del Tribunal quienes con su constante ayuda y colaboración me permitieron culminar con el desarrollo este trabajo.

El presente trabajo va dedicado a toda mi familia, quienes con su apoyo incondicional se han convertido en los pilares fundamentales en este largo camino sembrando en mí la dedicación, la fé, el esfuerzo y de quienes he aprendido que la perseverancia y la superación son necesarias para ver realizadas nuestras metas ya que con tanto sacrificio y afán han hecho de mi sueño, el sueño de ellos propios; especialmente a la memoria de mi padre Orlando Sánchez (+), a mi tío Marcelo Sánchez (+), quienes a pesar de no estar con nosotros, sus enseñanzas, sus espíritus siempre están presentes en nuestras mentes y en vuestros corazones, a mi madre querida Elena Labre quien ha estado siempre ayudándome de una u otra forma a cumplir todas mis anhelos, a mi esposa Ximena Herrera que con su perseverancia, amor y paciencia ha estado opoyándome en los momentos más difíciles de mi vida, a mis hijas Emily y Eliana quienes son el pilar de mi vida, a mis hermanos Lucía, Daniel, Roberto, a mis abuelos Miguel, Julia, Laura, cuñados, sobrinos y a mis suegros Víctor, Martha.

Wilian Xavier Sánchez Labre.

FIRMAS DE RESPONSABILIDAD

NOMBRE	FIRMA	FECHA
Dr. Romeo Rodríguez DECANO DE LA FACULTAD DE INFORMÁTICA Y ELECTRÓNICA
Ing. Iván Menes DIRECTOR DE LA ESCUELA DE INGENIERÍA EN SISTEMAS
Ing. Ivonne Rodríguez DIRECTORA DE TESIS
Ing. Wladimir Castro MIEMBRO DEL TRIBUNAL
Tlgo. Carlos Rodríguez DIRECTOR DEL CENTRO DE DOCUMENTACIÓN
NOTA DE LA TESIS	

“Yo, **WILIAN XAVIER SÁNCHEZ LABRE**, soy responsable de las ideas, doctrinas y resultados expuestos en esta tesis y el patrimonio intelectual de la Tesis de Grado pertenece a la **ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO**”

Wilian Xavier Sánchez Labre

ÍNDICE DE ABREVIATURAS

HCPCH	Honorable Consejo Provincial de Chimborazo.
AOT	Ahead-Of-Time (Antes-de-Tiempo).
ASP	Active Server Pages (Páginas de Servidor Activa).
BCL	Bass Class Library (Librería de Clase Básica).
BD	Base de Datos.
CD	Disk Compact (Disco Compacto).
CGI	Common Gateway Interface (Interfaz externa Común).
CLI	Lenguaje Intermedio Común.
CLR	Common Language Runtime (Lenguaje Común en Tiempo de Ejecución).
COC	Convención sobre configuración.
CRUD	Crear – Leer – Actualizar – Eliminar.
MS-DOS	Microsoft Disk Operating System (Sistema Operativo de Disco de Microsoft).
DRY	Don't repeat yourself (No te repitas).
E/S	Entrada/Salida.
ECMA	European Computer Manufacture's Association (Asociación fabricante de Computadora Euporea).
ESPOCH	Escuela Superior Politécnica de Chimborazo.
FTP	File Transfer Protocol. (Protocolo de Transferencia de Archivos).
GNOME	GNU Network Object Model Environment.
GNU	Es un acrónimo recursivo que significa "GNU No es Unix".

GPL	General Public License (Licencia General Pública).
GTK	Gimp Tool Kit.
GUI	Interfaz Grafica de Usuario.
HTML	HiperText Markup Lenguaje. (Lenguaje de Marcado de Hipertexto).
HTTP	HyperText Transfer Protocol. (Protocolo de Transferencia de Hipertexto).
IBM	International Business Machines Corporation (Corporación de Máquinas de Negocios Internacionales).
IP	Protocol Internet (Protocolo de Internet).
ISO	Sistema Internacional para le Estandarización.
JIT	Just-In-Time (Justo-a-Tiempo).
MCS	Mono Compiler Suite (Compilador de Mono).
MIT	Instituto de Tecnología de Massachussets.
MVC	Model View Controller.
ORM	Object Relational Mapping (Mapa de Objeto Relacional).
PHP	Personal Hypertext Preprocessor (Personal Hipertext Procesor).
RDBMS	Sistema de Manejo de base de Datos Relacional.
RoR	Ruby on Rails.
SEPOA	Sistema de Evaluación de Proyectos Operacional Anual.
SEPROCH	Seguimiento y Evaluación de Proyectos en Chimborazo.
SQL	Language Query Structure (Lenguaje de Consultas Estructurado).
URL	Uniform Resource Locutor (Localizador Uniforme de Recurso).
WEBrick	Servidor de Prueba de Ruby on Rails.
XSP	eXtensible Server Pages (Páginas de Servidor extensibles).

ÍNDICE GENERAL

PORTADA

AGRADECIMIENTO

DEDICATORIA

ÍNDICE DE ABREVIATURAS

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

CAPÍTULO I

MARCO REFERENCIAL

1.1. Problematicación	21
1.1.1. Formulaci3n del Problema	22
1.1.2. Sistematizaci3n del Problema	22
1.2. Objetivos	23
1.2.1. Objetivo General.....	23
1.2.2. Objetivos Específicos	23
1.3. Justificaci3n.....	23
1.4. Planteamiento de la Hip3tesis	25

CAPÍTULO II

MARCO TE3RICO

2.1. Introducci3n	27
2.2. Defini3n de Software Libre.....	31

2.2.1. ¿Qué no es Software Libre?	32
2.2.2. Clasificación de software libre	33
2.2.3. Características Generales del Software Libre	37
2.3. Tecnologías para el desarrollo de Aplicaciones Web.....	38
2.4. Sistemas Operativos	40
2.4.1. Tipos de Sistemas Operativos	40
2.4.2. Clasificación de Sistemas Operativos.....	42
2.5. Base de Datos Mysql	44
2.5.1. ¿Que es MySql?.....	45
2.5.2. Base de Datos Relacionales	45
2.5.3. Tipos de Datos en MySql.....	46
2.5.4. Lenguaje SQL.....	47

CAPÍTULO III

ANÁLISIS COMPARATIVO DE LAS TECNOLOGÍAS RUBY ON RAILS Y MONO

3.1. Introducción	49
3.2. Estudio de las Tecnologías Ruby on Rails y Mono.....	49
3.2.1. Tecnología Ruby on Rails.....	50
3.2.1.1. ¿Qué es Ruby on Rails?	51
3.2.1.2. Características de Ruby on Rails	53
3.2.1.3. Arquitectura de Rails	55
3.2.1.4. Licencias de la Tecnología Ruby on Rails.....	62
3.2.1.5. Entorno de Programación de Ruby.....	62
3.2.2. Tecnología Mono.....	78

3.2.2.1. ¿Qué es Proyecto Mono?	79
3.2.2.2. Características de Mono.....	80
3.2.2.3. Arquitectura de Mono	80
3.2.2.4. Licencias de la tecnología Mono	82
3.2.2.5. Entorno de Programación de la tecnología Mono	83
3.3. Determinación de los Parámetros de Comparación.....	92
3.4. Desarrollo de módulos de prueba para el Análisis de Comparación.....	96
3.4.1. Módulo en Windows.....	98
3.4.2. Módulo en Linux	106
3.5. Análisis Comparativo de las Tecnologías Ruby on Rails y Mono.....	108
3.5.1. Acceso a Base de Datos	108
3.5.2. Líneas de Código.....	111
3.5.3. Portabilidad	113
3.5.4. Interfaz de Usuario	116
3.5.5. Puntajes Alcanzados	119
3.6. Resultado del Análisis Comparativo	122
3.6.1. Ventajas y Desventajas de Ruby on Rails.....	123
3.6.2. Ventajas y Desventajas de la Tecnología Mono	125

CAPÍTULO IV

DESARROLLO DEL SISTEMA DE SEGUIMIENTO Y MONITOREO DE PROYECTOS EN EL CONSEJO PROVINCIAL DE CHIMBORAZO

4.1. Introducción	127
4.2. Secuencia de Etapas o Fases	128
4.3. FASE I – Ingeniería de la Información.....	128

4.3.1. Definición del Ámbito	128
4.3.2. Requerimientos.....	129
4.3.3. Análisis Institucional	129
4.3.4. Estudio de Factibilidad	132
4.4. FASE II – Análisis del Sistema.....	135
4.4.1 Casos de Uso del Sistema	135
4.4.2. Detalle de los Casos de Uso Identificados	135
4.4.3. Diagrama de los Casos de Uso	137
4.4.4. Diagramas de Secuencia	140
4.4.5. Diagramas de Colaboración	142
4.4.6. Diagramas de Calles	143
4.4.7. Diagramas de Clases.....	144
4.4.8. Diagramas de Base de Datos.....	145
4.4.9. Diagramas de Despliegue.....	145
4.5. FASE III – Diseño	146
4.5.1. Implementación	150
4.5.2 Pruebas	150

CONCLUSIONES

RESUMEN

SUMARY

GLOSARIO

ANEXOS

BIBLIOGRAFÍA

ÍNDICE DE TABLAS

CAPÍTULO I

MARCO REFERENCIAL

Tabla N° I. 1. Operacionalización Conceptual.....	25
Tabla N° I. 2. Operacionalización Metodológica.....	26

CAPÍTULO II

MARCO TEÓRICO

Tabla N° II. 1. Tipos de datos numéricos en Mysql.....	46
Tabla N° II. 2. Tipo de datos Cadena en Mysql.....	46
Tabla N° II. 3. Tipos de datos fecha y hora en Mysql.....	47

CAPÍTULO III

ANÁLISIS COMPARATIVO DE LAS TECNOLOGÍAS RUBY ON RAILS Y MONO

Tabla N° III. 1. Operadores en Ruby	64
Tabla N° III. 2. Funciones de Cadenas	66
Tabla N° III. 3. Caracteres especiales en expresiones regulares	66
Tabla N° III. 4. Clases de variables	74
Tabla N° III. 5. Variables de Sistema	75
Tabla N° III. 6. Control de Excepciones.....	76
Tabla N° III. 7. Estructura de Directorios de Rails	77
Tabla N° III. 8. Funciones de Cadenas	88

Tabla N° III. 9. Control de Excepciones en Mono	92
Tabla N° III. 10. Determinación de los Criterios de Comparación	93
Tabla N° III. 11. Variable del Parámetro de Comparación Acceso a Base de Datos.....	93
Tabla N° III. 12. Variable del Parámetro de Comparación Líneas de Código	94
Tabla N° III. 13. Variable del Parámetro de Comparación Portabilidad.....	94
Tabla N° III. 14. Variable del Parámetro de Comparación Acceso a Base de Datos.....	94
Tabla N° III. 15. Escala de Evaluación para las tecnologías	94
Tabla N° III. 16. Pesos en el parámetro para las variables	95
Tabla N° III. 17. Interpretación de Resultados en cada tecnología	96
Tabla N° III. 18. Calificación de Tecnologías	96
Tabla N° III. 19. Pesos de las variables para las dos tecnologías	108
Tabla N° III. 20. Parámetro Acceso a Base de Datos Tecnología Ruby on Rails.....	108
Tabla N° III. 21. Parámetro Acceso a Base de Datos Tecnología Mono	108
Tabla N° III. 22. Resultados del Parámetro Acceso a Base de Datos	109
Tabla N° III. 23. Pesos de las variables para las dos tecnologías	111
Tabla N° III. 24. Parámetro Líneas de Código Tecnología Ruby on Rails	111
Tabla N° III. 25. Parámetro Líneas de Código Tecnología Mono	111
Tabla N° III. 26. Resultado Parámetro Líneas de Código	112
Tabla N° III. 27. Pesos de las variables para las dos tecnologías	113
Tabla N° III. 28. Parámetro Potabilidad Tecnología Ruby on Rails	114
Tabla N° III. 29. Parámetro Portabilidad Tecnología Mono.....	114
Tabla N° III. 30. Resultado Parámetro Portabilidad.....	115
Tabla N° III. 31. Pesos de las variables para el parámetro Interfaz	116
Tabla N° III. 32. Parámetro Interfaz Tecnología Ruby on Rails.....	116

Tabla N° III. 33. Parámetro Interfaz Tecnología Mono	117
Tabla N° III. 34. Resultado Parámetro Interfaz	118
Tabla N° III. 35. Tabla General de Resultados	120
Tabla N° III. 36. Calificación final de las Tecnologías Ruby on Rails y Mono	123

CAPÍTULO IV

DESARROLLO DEL SISTEMA DE SEGUIMIENTO Y MONITOREO DE PROYECTOS EN EL CONSEJO PROVINCIAL DE CHIMBORAZO

Tabla N° IV. 1. Caso de Uso Autenticación de usuarios	135
Tabla N° IV. 2. Caso de Uso Ingreso de Información.....	136
Tabla N° IV. 3. Caso de Uso Reportes	136
Tabla N° IV. 4. Caso de Uso Cuentas de Usuarios	137

ÍNDICE DE FIGURAS

CAPÍTULO II

MARCO TEÓRICO

Figura N° II. 1. Logo de Ruby on Rails.....	28
Figura N° II. 2. Logo de Mono.....	28
Figura N° II. 3. Logo de Phyton.....	29
Figura N° II. 4. Logo de Perl.....	29
Figura N° II. 5. Logo de PHP.....	30
Figura N° II. 6. Logo de MySql.....	30
Figura N° II. 7. Estructura Conceptual del Software Libre.....	32
Figura N° II. 8. Tablas Relacionadas.....	45

CAPÍTULO III

ANÁLISIS COMPARATIVO DE LAS TECNOLOGÍAS RUBY ON RAILS Y MONO

Figura N° III. 1. Evolución de Ruby.....	50
Figura N° III. 2. Evolución de Ruby on Rails.....	51
Figura N° III. 3. Modelo – Vista – Controlador.....	52
Figura N° III. 4. Arquitectura de Ruby on Rails.....	55
Figura N° III. 5. Patrón Modelo – Vista – Controlador (MVC) Simple.....	56
Figura N° III. 6. Petición de una Vista.....	57
Figura N° III. 7. Patrón Modelo – Vista – Controlador – RoR Aplicación Web.....	59
Figura N° III. 8. Estructura de Directorios de Rails.....	78

Figura N° III. 9. Estructura de Mono.....	82
Figura N° III. 10. Versiones de Ruby – Rails – Gem.....	97
Figura N° III. 11. Versiones Mono.....	97
Figura N° III. 12. Versiones de Motor de Base de Datos MySql.....	97
Figura N° III. 13. Versiones de Ruby – Rails – Gem.....	97
Figura N° III. 14. Versiones Mono.....	98
Figura N° III. 15. Versiones de Motor de Base de Datos MySql.....	98
Figura N° III. 16. Usuarios para la Base de Datos prototipo	99
Figura N° III. 17. Creación de la aplicación (Ruby on Rails).....	99
Figura N° III. 18. Conexión a la base de datos	100
Figura N° III. 19. Scaffold – Creación de archivos rhtml.....	100
Figura N° III. 20. Servidor de Prueba WEBrick	101
Figura N° III. 21. Datos de Base de Datos (Tabla datos)	101
Figura N° III. 22. Prototipo de la tecnología RoR.....	101
Figura N° III. 23. Servidor de prueba XSP	105
Figura N° III. 24. Prototipo de la tecnología Mono	105
Figura N° III. 25. Servidor WEBrick en Linux para la tecnología Ruby on Rails.....	106
Figura N° III. 26. Servidor XSP en Linux para la Tecnología Mono.....	106
Figura N° III. 27. Ejecución del Prototipo en Linux – Tecnología Ruby on Rails	107
Figura N° III. 28. Ejecución del Prototipo en Linux – Tecnología Mono.....	107
Figura N° III. 29. Parámetro Acceso a Datos.....	110
Figura N° III. 30. Parámetro Líneas de Código	113
Figura N° III. 31. Parámetro Portabilidad.....	115
Figura N° III. 32. Parámetro Interfaz.....	118

Figura N° III. 33. Diagrama General de Resultados.....	120
Figura N° III. 34. Resultado Final	121

CAPÍTULO IV

DESARROLLO DEL SISTEMA DE SEGUIMIENTO Y MONITOREO DE PROYECTOS EN EL CONSEJO PROVINCIAL DE CHIMBORAZO

Figura N° IV. 1. Secuencia de Etapas o Fases	128
Figura N° IV. 2. Organigrama Estructural de la Organización.....	132
Figura N° IV. 3. Cronograma de Desarrollo	134
Figura N° IV. 4. Diagrama de Casos de Uso General	137
Figura N° IV. 5. Diagrama de Caso de Uso de Autenticación de usuarios	138
Figura N° IV. 6. Diagrama de Caso de Uso de Ingreso de información	138
Figura N° IV. 7. Diagrama de Caso de Uso de Reportes.....	139
Figura N° IV. 8. Diagrama de Caso de Uso Cuentas de Usuarios	139
Figura N° IV. 9. Diagrama de secuencia Autenticación de usuario	140
Figura N° IV. 10. Diagrama de secuencia Ingreso de información.....	140
Figura N° IV. 11. Diagrama de secuencia Reportes.....	141
Figura N° IV. 12. Diagrama de secuencia Cuentas de Usuarios.....	141
Figura N° IV. 13. Diagrama de Colaboración Autenticación de usuarios.....	142
Figura N° IV. 14. Diagrama de Colaboración Ingreso de información.....	142
Figura N° IV. 15. Diagrama de Colaboración Reportes	142
Figura N° IV. 16. Diagrama de Colaboración Cuentas de Usuarios	143
Figura N° IV. 17. Diagrama de Calles.....	143
Figura N° IV. 18. Diagrama de Clases	144

Figura N° IV. 19. Diagrama de Base de Datos	145
Figura N° IV. 20. Diagrama de Despliegue	145
Figura N° IV. 21. Diagrama de nodos	146
Figura N° IV. 22. Ventana General	146
Figura N° IV. 23. Menú Administrador.....	147
Figura N° IV. 24. Menú Referencias	147
Figura N° IV. 25. Menú Ubicación	148
Figura N° IV. 26. Menú Proyectos	148
Figura N° IV. 27. Menú Reportes	149
Figura N° IV. 28. Cerrar Sesión	149

INTRODUCCIÓN

La rápida evolución de las tecnologías y con ello el desarrollo de Aplicaciones Web mediante código abierto día a día adquieren más popularidad en todo el mundo, por esta razón el desarrollo de Aplicaciones Web, los servicios y las soluciones es un área de trabajo que ofrece un amplio rango de investigación para los desarrolladores.

El presente tema de investigación se proyecta hacia las tecnologías de software libre que en la actualidad existen en el mercado para desarrollar Aplicaciones Web, seguras y fáciles de usar; y totalmente económicas. Estas nuevas soluciones informáticas pueden ser aprovechadas por los usuarios en general con el fin de disminuir los costos y maximizar el aprovechamiento de recursos en una Empresa Pública o Privada.

Las Aplicaciones Web permiten a los diversos ordenadores comunicarse a través de una red, Internet, formada por la interconexión de diversas redes, mediante estas Aplicaciones Web hay muchos servicios que permite acceder a multitud de prestaciones y funciones, así como a infinidad de servicios, programas, tiendas, etc.

Para el Desarrollo de Aplicaciones Web mediante tecnologías de software libre, entre ellas Ruby on Rails y Mono; permite una implementación simple y sencilla de un sistema de comunicaciones que nos permite enviar cualquier tipo de información de una forma fácil, permitiendo que los servidores atiendan miles de peticiones y reduzcan los costes de desarrollo.

Para realizar este trabajo de investigación se ha decidido dividir el presente trabajo en cuatro capítulos los cuales se detallan a continuación:

- En el **primer capítulo** de este trabajo, se presenta los objetivos e hipótesis de la investigación.
- En el **segundo capítulo**, se realiza el Marco teórico para las aplicaciones de Software Libre, Sistemas Operativos, MySQL, Ruby on Rails y Mono.
- En el **tercer capítulo**, se enfoca al análisis comparativo entre las tecnologías Ruby on Rails y Mono.
- En el **cuarto capítulo**, se enfoca al desarrollo de la aplicación Web para el Seguimiento y Evaluación de Proyectos en el Departamento de Planificación del Honorable Consejo Provincial de Chimborazo (HCPCH), de esta manera se logra vincular la teoría con la práctica y culminar el trabajo investigativo.

Finalmente, este trabajo concluye emitiendo las conclusiones a las que se llegó luego de realizar el presente trabajo de investigación así como las debidas recomendaciones para la implantación del sistema que se desarrolló.

CAPÍTULO I

MARCO REFERENCIAL

1.1. Problematización

El Departamento de Planificación del HCPCH, lleva un Control de Proyectos que se están realizando a nivel de la Provincia de Chimborazo en forma manual mediante hojas electrónicas de Excel donde se ingresa toda la información (objetivos, metas, indicadores, actividades, costos) necesarias sobre un proyecto.

De igual manera en el momento en que alguna persona no autorizada pueda manipular la información que se encuentra ingresada en las Hojas de Excel puede ocasionar perdidas de datos o actualizaciones incorrectas.

Por tal razón, hay dificultades en muchos aspectos puesto que si necesitan un informe sobre el presupuesto, avance (porcentaje) de un proyecto, programa y eje, la información no puede ser entregada a tiempo por parte de la persona encargada de dicha información.

Por tal razón el almacenamiento de información en forma clasificada y organizada ayuda a disminuir el problema, por lo cual se realizará un sistema que utilice una base de datos donde se almacenará la información necesaria la cual posteriormente podrá ser procesada y dar reportes de inmediato según la necesidad del personal administrativo.

En sí el sistema propuesto trata de eliminar el almacenamiento de información en hojas de Excel dando a la vez una mejor seguridad a la información donde solo podrán manipular personas autorizadas en el sistema.

1.1.1. Formulación del Problema

¿Cuál es la tecnología más apropiada para el desarrollo de aplicaciones Web que se puede utilizar para el proceso de Seguimiento y Monitoreo de Proyectos en el Honorable Consejo Provincial de Chimborazo?

1.1.2. Sistematización del Problema

¿Cuáles serían las ventajas y desventajas de automatizar el proceso de Seguimiento y Monitoreo de Proyectos?

¿Qué tipo de tecnología existe en el mercado actual para implementar una solución fiable y factible en la HCPCH?

¿Existen sistemas de similares características implantadas en otras instituciones públicas o privadas?

1.2. Objetivos

1.2.1. Objetivo General

- Realizar un estudio comparativo de las tecnologías de software libre RUBY ON RAILS y MONO para el Desarrollo de Aplicaciones Web aplicado al Seguimiento y Monitoreo de Proyectos en el Consejo Provincial de Chimborazo.

1.2.2. Objetivos Específicos

- Definir criterios de comparación de las tecnologías RUBY ON RAILS Y MONO.
- Comparar las tecnologías de software libre RUBY ON RAILS Y MONO para establecer la tecnología más adecuada en el desarrollo de aplicaciones Web.
- Estudiar las ventajas y desventajas de las tecnologías de software libre RUBY ON RAILS Y MONO.
- Desarrollar una Aplicación Web en el Departamento de Planificación del Consejo Provincial de Chimborazo para el Seguimiento y Monitoreo de Proyectos utilizando la tecnología seleccionada en el estudio comparativo.

1.3. Justificación

El presente tema de investigación se proyecta hacia las tecnologías de software libre que en la actualidad existen en el mercado para desarrollar Aplicaciones Web, seguras y fáciles de usar; y totalmente económicas.

La rápida evolución de las tecnologías y con ello el desarrollo de Aplicaciones Web mediante código abierto día a día adquieren más popularidad en todo el mundo, por

esta razón el desarrollo de Aplicaciones Web, los servicios y las soluciones es un área de trabajo que ofrece un amplio rango de investigación para los desarrolladores.

Estas nuevas soluciones informáticas pueden ser aprovechadas por los usuarios en general con el fin de disminuir los costos y maximizar el aprovechamiento de recursos en una Empresa Pública o Privada.

Las Aplicaciones Web permiten a los diversos ordenadores comunicarse a través de una red, Internet, formada por la interconexión de diversas redes, mediante estas Aplicaciones Web hay muchos servicios que permite acceder a multitud de prestaciones y funciones, así como a infinidad de servicios, programas, tiendas, etc.

Con el avance de las tecnologías y en la actualidad las Aplicaciones Web desarrolladas en su gran mayoría por medio de software libre permite una implementación simple y sencilla de un sistema de comunicaciones que nos permite enviar cualquier tipo de información de una forma fácil, permitiendo que los servidores atiendan a dichas peticiones y reduzcan los costes de desarrollo.

El Honorable Consejo Provincial de Chimborazo (HCPCH) se ha visto con la necesidad de desarrollar un nuevo Sistema para el Seguimiento y Monitoreo de Proyectos que se realizan en la todo la Provincia de Chimborazo, el mismo que permitirá tener un control de los proyectos realizados; por ende se administrará de una mejor forma la información de cada proyecto, brindando al personal administrativo del Departamento de Planificación saber en cuanto avanzado un proyecto en particular.

En la actualidad HCPCH lleva un Control de Proyectos en forma manual mediante hojas electrónicas de Excel donde se ingresa toda la información necesaria para el mismo, por ende con la realización de una aplicación Web en el Departamento de Planificación se lograría beneficios para los siguientes aspectos:

- Un control del estado en que se encuentran los proyectos.
- Información actualizada del avance (porcentaje) de un proyecto, programa y eje.
- Presupuestos asignados a cada proyecto, programa y eje.

1.4. Planteamiento de la Hipótesis

El análisis comparativo de las Tecnologías de Software Libre RUBY ON RAILS y MONO, permitirá seleccionar la tecnología más adecuada para la implementación de una aplicación Web la misma que ayudará en la Administración de información durante el Seguimiento y Monitoreo de Proyectos en el Departamento de Planificación del Consejo Provincial de Chimborazo.

DETERMINACIÓN DE VARIABLES:

Sistema Seguimiento y Monitoreo

Administración de Información HCPCH

OPERACIONALIZACION DE VARIABLES

Tabla N° I. 1. Operacionalización Conceptual

VARIABLES	CONCEPTOS
Sistema Seguimiento y Monitoreo	Sistema de computadora que ayudara al personal administrativo del Departamento de Planificación a Administrar la información de Proyectos.
Administración de Información HCPCH	Mejor aprovechamiento de presupuestos y avances (porcentajes) de actividades de cada proyecto, programa y eje.

Tabla N° I. 2. Operacionalización Metodológica

VARIABLE	INDICADOR	TÉCNICA	FUENTE DE VERIFICACIÓN
Sistema Seguimiento y Monitoreo	Eficiencia	Recopilación de información	Comunidades Departamento de Planificación Base de Datos
	N° de Proyectos	Revisión de presupuesto	
Administración de Información HCPCH	Eficacia	Revisión de presupuesto	Comunidades Personal interno HCPCH
	N° de Proyectos	Observación Recursos disponibles.	

CAPÍTULO II

MARCO TEÓRICO

2.1. Introducción

El movimiento de Software Libre tuvo sus orígenes en el Instituto de Tecnología de Massachussets (MIT). En los laboratorios informáticos del MIT en la década del '70 empezó a trabajar Richard Stallman, quien es considerado "el creador del movimiento".

El "software libre" es un tipo particular de software que se basa en que el código fuente de un programa debe estar disponible, para luego poder modificarlo, mejorarlo y distribuirlo libremente, es decir es un asunto de libertad, no de precio, se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software.

Ruby es el lenguaje hecho para programar con el menor código posible, seguro, Orientado a Objetos, trabaja con MySQL, PostgreSQL, SQLite, Oracle, SQL Server, DB2, entre otros. Ruby es un lenguaje de programación reflexivo y orientado a objetos creado por el programador japonés Yukihiro "Matz" Matsumoto en 1993.

Rails es un framework Model View Controller (MVC) de reciente creación, especialmente adecuado para el desarrollo Web utilizando metodologías ágiles. Ruby on Rails es open source, esta basado en un potente lenguaje (Ruby) y está optimizado para permitir el desarrollo veloz de aplicaciones Web 2.0 conservando al mismo tiempo un alto nivel de calidad.



Figura N° II. 1. Logo de Ruby on Rails

Mono es el nombre de un proyecto de código abierto iniciado por Ximian y actualmente impulsado por Novell (tras su adquisición de Ximian) para crear un grupo de herramientas libres, basadas en GNU/Linux y compatibles con .NET según lo especificado por el ECMA. Mono es la respuesta del mundo Open Source a la plataforma .Net de Microsoft. Mono permite desarrollar en el lenguaje C# para la plataforma Windows y Linux.

El proyecto mono nace el 9 de Julio del 2001 liderado por Miguel de Icaza (fundador del proyecto GNOME), ya que pensaba que esta tecnología podría ayudarle con los problemas que había tenido a la hora de desarrollar GNOME.



Figura N° II. 2. Logo de Mono

Python es un lenguaje muy expresivo, es decir, los programas Python son muy compactos, un programa Python suele ser bastante más corto que su equivalente en

lenguajes como C. Python es muy legible. La sintaxis de Python es muy elegante y permite la escritura de programas cuya lectura resulta más fácil que si utilizáramos otros lenguajes de programación. El entorno de ejecución de Python detecta muchos de los errores de programación que escapan al control de los compiladores y proporciona información muy rica para detectarlos y corregirlos. Python puede usarse como lenguaje imperativo procedimental o como lenguaje orientado a objetos.



Figura N° II. 3. Logo de Python

Perl es un lenguaje de propósito general originalmente desarrollado para la manipulación de texto y que ahora es utilizado para un amplio rango de tareas incluyendo administración de sistemas, Desarrollo Web, programación en red, desarrollo de GUI y más. Sus principales características son que es fácil de usar, soporta tanto la programación estructurada como la programación orientada a objetos y la programación funcional, tiene incorporado un poderoso sistema de procesamiento de texto y una enorme colección de módulos disponibles.



Figura N° II. 4. Logo de Perl

PHP (acronimo de "PHP: Hypertext Preprocessor") es un lenguaje interpretado de alto nivel embebido en páginas HTML y ejecutado en el servidor. Quizás la característica

más potente y destacable de PHP es su soporte para una gran cantidad de bases de datos. Escribir un interfaz vía Web para una base de datos es una tarea simple con PHP.

Antes de las bases de datos se utilizaban archivos secuenciales como almacenes de datos. Estos daban un acceso muy rápido pero sólo de forma secuencial (para acceder a una posición, se debía recorrer el archivo entero).



Figura N° II. 5. Logo de PHP

Con las bases de datos también aparecieron los archivos indexados, donde el acceso ya podía ser aleatorio (acceder de una vez a la posición deseada del archivo). Según Henry F. Korth autor del libro “Fundamentos de Bases de Datos” se define una base de datos como una serie de datos organizados y relacionados entre sí, y un conjunto de programas que permitan a los usuarios acceder y modificar esos datos.

Uno de los propósitos principales de un sistema de base de datos es proporcionar a los usuarios una visión abstracta de los datos. Es decir, el sistema esconde ciertos detalles de cómo se almacenan y mantienen los datos. MySQL es un sistema gestor de bases de datos, la misma que se trata de una de las bases de datos más rápida actualmente. MySQL no se trata de una base de datos de escritorio (como Microsoft Access), MySQL es un servidor de bases de datos sobre una red TCP/IP. El puerto donde escucha el servidor MySQL es el 3306 (TCP)



Figura N° II. 6. Logo de MySQL

2.2. Definición de Software Libre

Software libre es la denominación del software que una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente.

Según la Free Software Foundation, el software libre se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software; de modo más preciso, se refiere a cuatro libertades de los usuarios del software:

1. La libertad de usar el programa, con cualquier propósito.

"Usar el programa con cualquier propósito". Es decir, el ejercicio de esta libertad implica que lo podemos utilizar con cualquier fin, ya sea educativo, cultural, comercial, político, social, etc. Esta libertad deriva de que hay ciertas licencias que restringen el uso del software a un determinado propósito, o que prohíben su uso para determinadas actividades.

2. La libertad de estudiar cómo funciona el programa, y adaptarlo a sus necesidades. El acceso al código fuente es una condición previa para esto.

"Estudiar como funciona el programa, y adaptarlo a sus necesidades". Significa que podemos estudiar su funcionamiento (al tener acceso al código fuente) lo que nos va a permitir, entre otras cosas: descubrir funciones ocultas, averiguar como realiza determinada tarea, descubrir que otras posibilidades tiene, que es lo que le falta para hacer algo, etc.

3. La libertad de distribuir copias, con lo que se ayuda a otros.

"Distribuir copias". Quiere decir que soy libre de redistribuir el programa, ya sea gratis o con algún costo, ya sea por email, FTP o en CD, ya sea a una persona o a varias, ya sea a un vecino o a una persona que vive en otro país, etc.

4. La libertad de mejorar el programa y hacer públicas las mejoras a los demás, de modo que toda la comunidad se beneficie.

"Mejorar el programa, y liberar las mejoras al público". Tengo la libertad de hacer mejor el programa, o sea que puedo: hacer menores los requerimientos de hardware para funcionar, que tenga mayores prestaciones, que ocupe menos espacio, que tenga menos errores, etc.

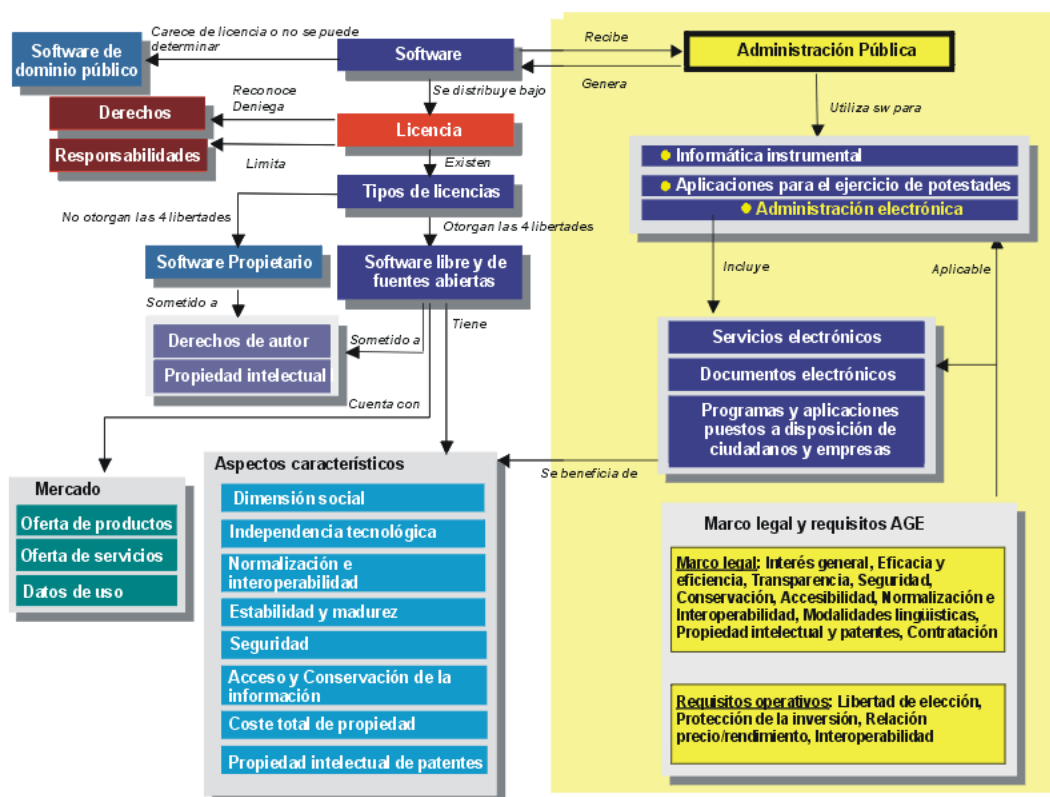


Figura Nº II. 7. Estructura Conceptual del Software Libre

2.2.1. ¿Qué no es Software Libre?

- **Software regalado:** o de costo cero, pero sin el código fuente. Es el que normalmente viene en los CD's de revistas de computación o que se consigue en sitios freeware.

- **Software con el código fuente:** esto quiere expresar que el software se provee con su código fuente, pero no necesariamente brinda las libertades del Software Libre.
- **Software de dominio publico:** este tipo de software no tienen licencias de uso, por lo tanto corre el peligro de dejar de serlo si alguien lo utiliza con el fin de apropiárselo.

2.2.2. Clasificación de software libre

- **De acuerdo a su costo**

De costo cero: también conocido como software gratis o gratuito. Es aquel software cuyo costo de adquisición es nulo, es decir, no hace falta efectuar un desembolso de dinero para poder usarlo.

De costo mayor a cero: también se conoce como software "comercial o de pago". Es el software desarrollado por una entidad que tiene la intención de hacer dinero con su uso.

- **De acuerdo a la apertura de su código fuente**

De código fuente abierto: también llamado "de fuente abierta" u "Open Source". Es aquel software que permite tener acceso a su código fuente a través de cualquier medio (ya sea acompañado con el programa ejecutable, a través de Internet, a través del abono de una suma de dinero, etc.). A la idea esencial del Open Source, ofrecer programas con acceso al código fuente, van unidas una serie de conceptos:

- **Flexibilidad.** Si el código fuente está disponible, los desarrolladores pueden modificar los programas de acuerdo a sus necesidades. Además, se produce un flujo constante de ideas que mejora la calidad de los programas.
 - **Fiabilidad y seguridad** Con muchos programadores a la vez escrutando el mismo trabajo, los errores se detectan y corrigen antes, por lo que el producto resultante es más fiable y eficaz que el comercial.
 - **Rapidez de desarrollo** Las actualizaciones y ajustes se realizan a través de una comunicación constante vía Internet.
 - **Relación con el usuario.** El programador se acerca mucho más a las necesidades reales de su cliente, y puede crear un producto específico para él.
- a. **De código fuente cerrado:** también llamado "software cerrado". Es el software que no tiene disponible su código fuente disponible por ningún medio, ni siquiera pagando. Generalmente tiene esta característica cuando su creador desea proteger su propiedad intelectual.
- i. **Software Semilibre:** No es un software libre pero posee una con autorización que permite usarlo, copiarlo, distribuirlo y modificarlo (incluyendo la distribución de versiones modificadas) sin fines de lucro.
 - ii. **Programa Freeware:** Es un tipo de software que permite la libre redistribución (incluso la incentiva) del mismo pero no su modificación y su código fuente no está disponible.

- iii. **Programa Shareware:** Es un software con autorización para redistribuir copias, pero con un tiempo limitado y si se requiere pagar por su uso se debe pagar por la licencia.
 - iv. **Software Propietario (Privativo/Cerrado):** Es un tipo de software donde su uso, su redistribución o su modificación está prohibida porque su código fuente está "cerrado" o se requiere de una autorización para leerlo o bien se encuentra tan restringido que no se lo puede hacer libre de un modo efectivo.
 - v. **Software Comercial.** Es un software desarrollado por una organización que lucra a través del uso del mismo. Software comercial y propietario (privativo/cerrado) no son la misma cosa. La mayoría del software comercial es propietario, sin embargo existe Software Libre Comercial y hay Software no Libre Comercial.
- **De acuerdo a su protección**
 - a. **De dominio publico:** es el software que no esta protegido por ningún tipo de licencia. Cualquiera puede tomarlo y luego de modificarlo, hacerlo propio.
 - b. **Protegido por licencias:** es el tipo de software protegido con una licencia de uso. Dentro de este grupo tenemos:
 - i. **Protegido con copyright:** es decir, con derechos de autor (o de copia). El usuario no puede adquirirlo para usarlo y luego vender copias (salvo con la autorización de su creador).
 - ii. **Protegido con copyleft GPL (General Public License):** es aquel cuyos términos de distribución no permiten a los redistribuidores agregar

ninguna restricción adicional. Quiere decir que cada copia del software, aun modificada, sigue siendo como era antes.

- **De acuerdo a su "legalidad"**

- a. **Legal:** es aquel software que se posee o circula sin contravenir ninguna norma. Por ejemplo, si tengo un software con su respectiva licencia original y con su certificado de autenticidad, o si lo tengo instalado en una sola computadora (porque la licencia solo me permite hacer eso).

- b. **Ilegal:** es el software que se posee o circula violando una norma determinada. Por ejemplo: tengo licencia para usarlo en una sola computadora pero lo instalo en mas de una, no tengo la licencia pero lo puedo utilizar mediante artificios (cracks, patches, loaders, key generators, números de serie duplicados, etc)

- **De acuerdo a su "filosofía"**

- a. **Propietario:** es aquel software que refleja el hecho de que su propiedad absoluta permanece en manos de quien tiene sus derechos y no del usuario, quien únicamente puede utilizarlo bajo ciertas condiciones. Su uso, redistribución y/o modificación están prohibidos o restringidos de modo tal que no es posible llevarlos a cabo. Es decir, este tipo de software le da al usuario derechos limitados sobre su funcionamiento, cuyo alcance establece el autor o quien posea ese derecho. Por ejemplo, ese derecho puede ser el de ejecutar el programa "tal como es" en una determinada computadora.

b. Libre: es el tipo de software que le da al usuario la libertad de usarlo, estudiarlo, modificarlo, mejorarlo, adaptarlo y redistribuirlo, con la única restricción de no agregar ninguna restricción adicional al software modificado, mejorado, adaptado o redistribuido. Vale aclarar que debe permitir el acceso al código fuente, debido a que ello es una condición imprescindible para ejercer las libertades de estudiarlo, modificarlo, mejorarlo y adaptarlo.

2.2.3. Características Generales del Software Libre

El éxito del Software Libre se debe en su mayor parte a Internet. El Internet ha permitido que las personas interesadas en los varios componentes del Software Libre se pongan fácilmente en contacto con otras. Es así que el Internet de esta manera actúa como un catalizador que acelera el desarrollo y sintetiza el conocimiento en áreas muy específicas es decir:

- Todo el mundo tiene derecho de usarlo sin costo alguno.
- Todo el mundo tiene derecho a acceder a su diseño y aprender de él.
- Todo el mundo tiene derecho de modificarlo: si el software tiene limitaciones o no es adecuado para una tarea, es posible adaptarlo a necesidades específicas y redistribuirlo libremente.
- No tiene un costo asociado, es gratuito.
- Es de libre distribución, cualquier persona puede regalarlo, venderlo o prestarlo.

2.3. Tecnologías para el desarrollo de Aplicaciones Web

Una aplicación Web está comúnmente estructurada como una aplicación de tres partes. En su forma más común, el navegador Web es la primera parte, un motor usando alguna tecnología Web dinámica (ejemplo: CGI, PHP, Java Servlets o ASP) es la parte intermedia, y una base de datos como última parte. El navegador Web manda peticiones a la parte del medio, que la entrega valiéndose de consultas y actualizaciones a la base de datos generando una interfaz de usuario.

Existen numerosos lenguajes de programación empleados para el desarrollo de Aplicaciones Web, entre los que destacan:

- **PHP**
 - PHP (acronimo de "PHP: Hypertext Preprocessor") es un lenguaje interpretado de alto nivel embebido en páginas HTML y ejecutado en el servidor.

- **ASP/ASP.NET**
 - Microsoft .NET Framework se integra en el entorno de desarrollo de forma transparente al programador, y lo utiliza para ayudarnos a desarrollar de forma rápida, eficiente y segura, nuestras aplicaciones. Nos permite aumentar el rendimiento y disminuir la curva de tiempo de desarrollo enormemente.
 - La filosofía de ASP resulta muy sencilla, en pocas palabras se puede definir de la siguiente forma: las páginas ASP, también llamadas páginas activas, son páginas que contienen código HTML, script de cliente y un script que se ejecuta en el servidor, dando como resultado código HTML.

- **Perl**
 - Perl es un lenguaje de propósito general originalmente desarrollado para la manipulación de texto y que ahora es utilizado para un amplio rango de tareas incluyendo administración de sistemas, Desarrollo Web, programación en red, desarrollo de GUI y más.
- **Ruby**
 - Ruby es el lenguaje hecho para programar con el menor código posible, seguro, Orientado a Objetos
- **Python**
 - Python es un lenguaje muy expresivo, es decir, los programas Python son muy compactos, un programa Python suele ser bastante más corto que su equivalente en lenguajes como C. Python es muy legible. La sintaxis de Python es muy elegante y permite la escritura de programas cuya lectura resulta más fácil que si utilizaremos otros lenguajes de programación.
- **Mono**
 - Mono es el nombre de un proyecto de código abierto iniciado por Ximian y actualmente impulsado por Novell (tras su adquisición de Ximian) para crear un grupo de herramientas libres, basadas en GNU/Linux y compatibles con .NET según lo especificado por el ECMA.

2.4. Sistemas Operativos

En el inicio de la informática cada fabricante tenía sus propios sistemas operativos que no eran compatibles con los de otros, incluso dentro de un mismo fabricante podían coexistir varios, caso típico de IBM. Estos se conocen como sistemas propietarios.

La tendencia actual es hacia los llamados sistemas abiertos, lo cual indica que estos sistemas operativos trabajan sobre una gran variedad de máquinas con independencia del fabricante del equipo. La gran ventaja es el ahorro a todos los niveles, pues por ejemplo una empresa con ordenadores de distintos fabricantes puede tener totalmente uniformado todo su software.

2.4.1. Tipos de Sistemas Operativos

- **Windows**
 - Microsoft es el más grande informático que produce y comercializa Windows, el sistema operativo que usa el 90% de los ordenadores personales de todo el mundo. Windows es sencillo de usar ya que cubre la gran mayoría de necesidades del usuario medio. Ya sea para escribir documentos, navegar por Internet, escuchar música, ver películas.

- **Unix**
 - Fue diseñado en los laboratorios Bell de la empresa AT&T, para su empleo en ordenadores marca Digital. Dadas sus características pronto se difundió ampliamente en ambientes universitarios, se ha considerado como un sistema operativo orientado hacia ambientes de investigación y no en aplicaciones de gestión.

- **MS-DOS**

- Fue un sistema operativo adaptado por Microsoft para IBM (PC-DOS), y en concreto para el modelo PC, aunque se popularizó rápidamente siendo el más usado a nivel personal. Dado que el entorno es poco amigable se crearon añadidos que proporcionan un ambiente de trabajo más fácil, el que ha tenido más éxito es WINDOWS (no estrictamente por razones de calidad), que ofrece un entorno gráfico de ventanas y sencillez de manejo mediante un ratón. La principal desventaja de MS-DOS es que es monousuario y monotarea., puede trabajar un usuario (no admite terminales) y que a su vez este sólo puede ejecutar un programa al mismo tiempo.

- **Linux**

- Linux es básicamente un sistema operativo compatible con UNIX, que opera principalmente bajo equipos compatibles con el estándar del mercado. Su ventaja principal es que su costo es prácticamente nulo. Fue escrito por Linus Torvalds (1969-), como un sistema operativo abierto y estandar, siendo desarrollado posteriormente por muchos programadores, de forma independiente. El código fuente, gestores de dispositivos y utilidades están disponibles gratuitamente.

- **Mac OS**

- Mac OS es considerado por muchos expertos el sistema operativo más sencillo de utilizar, más innovador y de estética más cuidada.

2.4.2. Clasificación de Sistemas Operativos

Debido a la evolución de los sistemas operativos fue necesario realizar una clasificación; considerando las diferencias existentes entre sus componentes los podemos clasificar en:

- Sistemas operativos por lotes.
- Sistemas operativos multiprogramación.
- Sistemas operativos multiusuario.
- Sistemas operativos de tiempo compartido.
- Sistemas operativos de tiempo real.

Sistemas operativos por lotes

Los sistemas operativos por lotes requieren que la información esté reunida en bloque o "lote" (el programa, los datos, y las instrucciones). Los trabajos son procesados en el orden de admisión, según el modelo de "primero en llegar primero en ser atendido". En estos sistemas la memoria se divide en dos zonas. Una de ellas es ocupada por el sistema operativo, y la otra se usa para cargar programas transitorios para su ejecución. Cuando termina la ejecución de un programa se carga un nuevo programa en la misma zona de memoria.

Sistemas operativos multiprogramación

Los sistemas de multiprogramación son capaces de soportar dos o más procesos concurrentes múltiples, permiten que residan al mismo tiempo en la memoria primaria las instrucciones y los datos procedentes de dos o más procesos. Estos sistemas implican la operación de multiproceso, para el manejo de la información. Se

caracterizan principalmente por un gran número de programas activos simultáneamente que compiten por los recursos del sistema, como el procesador, la memoria, y los "dispositivos de E/S". Estos sistemas monitorean el estado de todos los programas activos y recursos del sistema.

Sistemas operativos multiusuario

Los sistemas operativos multiusuario permiten acceder simultáneamente a un sistema de computadoras a través de dos o más terminales. Este tipo de sistema operativo es fundamental en el manejo de redes de computadoras actualmente.

Sistemas operativos de tiempo compartido

Los sistemas operativos de tiempo compartido tratan de proporcionar un reparto equitativo de los recursos comunes para dar la impresión a los usuarios de que poseen una computadora independiente. El administrador de archivos proporciona protección y control en el acceso de la información, dada la posibilidad de concurrencia y conflictos al tratar de acceder a los archivos.

Sistemas operativos de tiempo real

Estos sistemas tienen como objetivo proporcionar tiempos más rápidos de respuesta, procesar la información sin tiempos muertos. En estos sistemas el administrador de memoria es relativamente menos solicitado debido a que muchos procesos residen permanentemente en memoria. El administrador de archivos se encuentra normalmente en grandes sistemas de tiempo real y su objetivo principal es manejar la velocidad de acceso, más que la utilización eficaz del almacenamiento secundario.

2.5. Base de Datos Mysql

El servidor de bases de datos MySQL soporta distintos tipos de tablas, tales como ISAM, MyISAM, InnoDB, y DBD (Berkeley Database). De éstos, InnoDB es el tipo de tabla más importante (después del tipo predeterminado, MyISAM). Las tablas del tipo InnoDB están estructuradas de forma distinta que MyISAM, ya que se almacenan en un sólo archivo en lugar de tres, y sus principales características son que permite trabajar con transacciones, y definir reglas de integridad referencial.

Mysql en los últimos años ha tenido un crecimiento vertiginoso. Es la base de datos de código abierto más popular del mundo. Código abierto significa que todo el mundo puede acceder al código fuente, es decir, al código de programación de Mysql, esto significa que también todos pueden contribuir con ideas, elementos, mejoras o sugerir optimizaciones. Y así es que Mysql ha pasado de ser una pequeña base de datos a una completa herramienta. Su rápido desarrollo se debe en gran medida a la contribución de mucha gente al proyecto, así como la dedicación del equipo de Mysql.

Lo que en un tiempo se consideró como un sencillo servidor de Base de Datos para uso en sitios Web, se ha convertido en la actualidad en una solución viable para la administración de datos. Mysql es un sistema de administración de bases de datos relacional (RDBMS). Se trata de un programa capaz de almacenar una enorme cantidad de datos de gran variedad y de distribuirlos para cubrir las necesidades de cualquier tipo de organización, desde pequeños establecimientos comerciales a grandes empresas y organismos administrativos. Mysql compite con sistemas RDBMS propietarios como Oracle, Sql Server y Db2.

2.5.1. ¿Que es MySql?

- Es un sistema gestor de bases de datos
- Se trata de una de las bases de datos más rápida actualmente.
- No se trata de una base de datos de escritorio (como Microsoft Access), MySQL es un servidor de bases de datos sobre una red TCP/IP.
- El puerto donde escucha el servidor MySQL es el 3306 (TCP)

2.5.2. Base de Datos Relacionales

- Una base de datos relacional (MySQL) consiste en una serie tablas relacionadas, donde cada tabla consiste en una serie de registros (filas) y cada registro tiene una serie de campos (columnas)
- Cada registro tiene que tener un identificador único (ID), también conocido como clave primaria. Cuando se usa un identificador en una tabla secundaria se llama clave foránea.
- Cada campo (atributo) usa un tipo de datos particular, por ejemplo Char, Varchar, Int, boolean, etc.

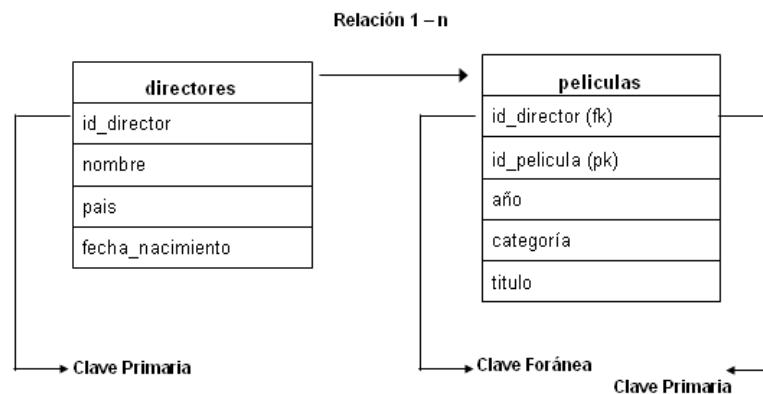


Figura N° II. 8. Tablas Relacionadas

2.5.3. Tipos de Datos en MySql

Existen tres tipos fundamentales de columnas: numéricas, de cadena y de fecha. Por regla general se debe seleccionar el tipo de columna de menor tamaño, ya que de esta forma se ahorra espacio y se logra una mayor velocidad de acceso y actualización. Sin embargo, si se selecciona un tipo de columna demasiado pequeño, puede dar como resultado la pérdida de datos o que se recorten al introducirlos.

- **Tipo de Columna Numérico**

- Las columnas numéricas están diseñadas para almacenar todo tipo de datos numéricos, como precios, edades y cantidades. Hay dos tipos principales de tipos numéricos: tipos enteros y de punto flotante.

Tabla N° II. 1. Tipos de datos numéricos en Mysql

TIPO		
tinyint	mediumint	float
bit	int	double
bool	integer	dec
smallint	bigint	numeric

- **Tipo de Columna Cadena**

- Los tipos de columna de cadena se utilizan para almacenar todo tipo de datos compuestos de caracteres como nombres, direcciones.

Tabla N° II. 2. Tipo de datos Cadena en Mysql

TIPO		
char	blob	longblob
varchar	text	longtext
tinyblob	mediumblob	enum
tinytext	mediumtext	set

- **Tipo de Columna de Fecha y Hora**

- Los tipos de columna de fecha y hora están diseñados para trabajar con las necesidades especiales que exigen los datos de tipo temporal y se puede utilizar para almacenar datos tales como la hora del día o fechas de nacimiento.

Tabla N° II. 3. Tipos de datos fecha y hora en Mysql

TIPO	
datetime	time
date	year
timestamp	

2.5.4. Lenguaje SQL

- Se trata de un lenguaje para manipular bases de datos (SQL = Lenguaje de Consultas Estructurado)
- Se trata de un lenguaje estándar, ampliamente utilizado.
- Se compone de un grupo de órdenes básicas para visualizar registros, borrar, actualizar y añadir registros.
- SQL trabaja con conjuntos de múltiples filas
- Un conjunto de datos puede ser vacío (sin filas) o puede ser una tabla entera.
- SQL es un lenguaje no-procedimental, es decir, solo hay que indicar cual es la operación que se quiere realizar, pero no la forma de hacerlo.

- **Reglas de SQL**

- Los nombres de tablas, columnas, etc., deben:
- Contener letras (A-Z, a-z), números (0-9) o guión bajo “_”

- Empezar por una letra
- Usar siempre minúsculas en nombres de tablas y columnas (campos)
- Si el nombre de un campo es ambiguo, debemos indicar la tabla a la que nos referimos (tabla.campo)

- **Operaciones con SQL**

- Insertar datos (insert)**

- Insert into nombre_tabla (lista_campos) values (lista_valores)

- Seleccionar datos para leerlos (select)**

- Select * from nombre_tabla

- Actualizar datos (Update)**

- Update nombre_tabla set nombre_columna= valor

- Eliminar datos (Delete)**

- Delete from nombre_tabla borra todas las filas

CAPÍTULO III

ANÁLISIS COMPARATIVO DE LAS TECNOLOGÍAS RUBY ON RAILS Y MONO

3.1. Introducción

A continuación se procede a efectuar un análisis comparativo entre las tecnologías de software libre Ruby on Rails y Mono para determinar cual de las dos tecnologías es la más óptima para el desarrollo de aplicaciones Web y así permita brindar una solución factible y fiable en la realización del Sistema de Seguimiento y Monitoreo de Proyectos en el Departamento de Planificación del Honorable Consejo Provincial de Chimborazo.

3.2. Estudio de las Tecnologías Ruby on Rails y Mono

En la actualidad existen muchas tecnologías de software libre para el desarrollo de aplicaciones Web entre ellas PHP, PYTHON, PERL, MONO, RUBY ON RAILS ASP, JAVA, bastante populares, y de gran importancia para los desarrolladores, de las cuales se han seleccionado dos tecnologías como son Ruby on Rails y Mono por razones de que estas dos tecnologías trabajan con framework y tiene su propio servidor Web para sus respectivas pruebas.

3.2.1. Tecnología Ruby on Rails

Ruby on Rails es un entorno de desarrollo Web de código abierto que está optimizado para satisfacción de los programadores y de la productividad. Ruby fue creado por el Japonés Yukihiro Matsumoto, la Primera versión fue en el año 1995.

Lenguaje de scripting verdaderamente orientado a objetos, dinámicamente tipado, muy flexible y con lo mejor de otros lenguajes como: Smalltalk, Perl, Python, ADA, Lisp, Eiffel. Es Open Source como todas las buenas cosas de este mundo, y tiene una comunidad que crece de forma impresionante.

Tiene el soporte de empresas como Sun, que ha aportado con proyectos como JRuby y la maquina para el servidor de control de versiones (Subversion), actualmente bajo Solaris 10.

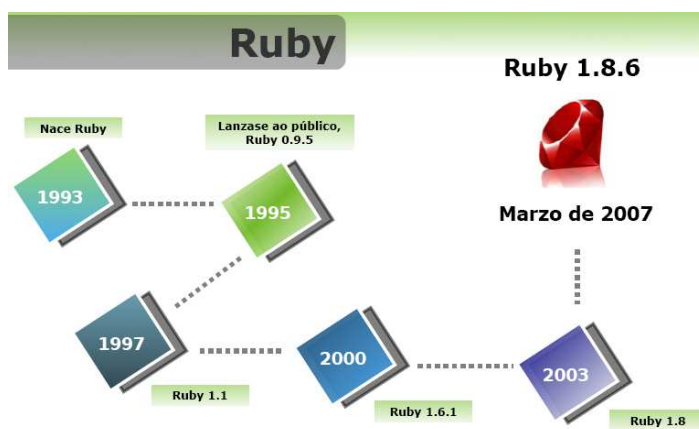


Figura N° III. 1. Evolución de Ruby

Ruby on Rails (RoR) o simplemente Rails fue creado a partir de 2003 por David Heinemeier Hansson (desarrollador Web danés que trabaja en 37 signals) y fue extraído a partir de una aplicación que él desarrolló: BaseCamp. Fue enviado a la comunidad Open a partir de Octubre del 2004.

Rails es un framework para el desarrollo de aplicaciones Web, muy completas. Su objetivo es tener desarrolladores ágiles y hacer que el desarrollo de aplicaciones Web sea rápido y sencillo.

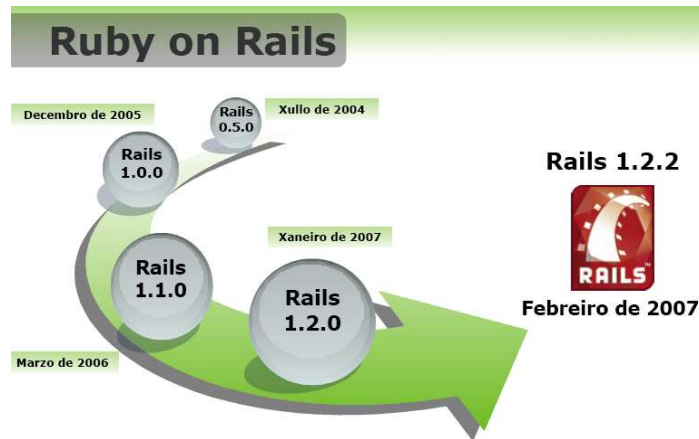


Figura Nº III. 2. Evolución de Ruby on Rails

3.2.1.1. ¿Qué es Ruby on Rails?

Ruby es un lenguaje de programación dinámico y orientado a objetos, es decir es un “lenguaje de guiones (scripts) para una programación orientada a objetos rápida y sencilla”.

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.

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, etc.
- Métodos singleton.
- Mixins por módulos.
- Iteradores y cierres.

Rails es un Framework open source para el desarrollo de aplicaciones Web, escrita en el lenguaje de programación Ruby, basada en patrones de diseño como: MVC (Modelo – Vista – Controlador), Active Record

Ruby on Rails es un entorno de desarrollo Web, de código abierto (open source, software libre), que nos permite construir aplicaciones Web flexibles y robustas rápidamente.

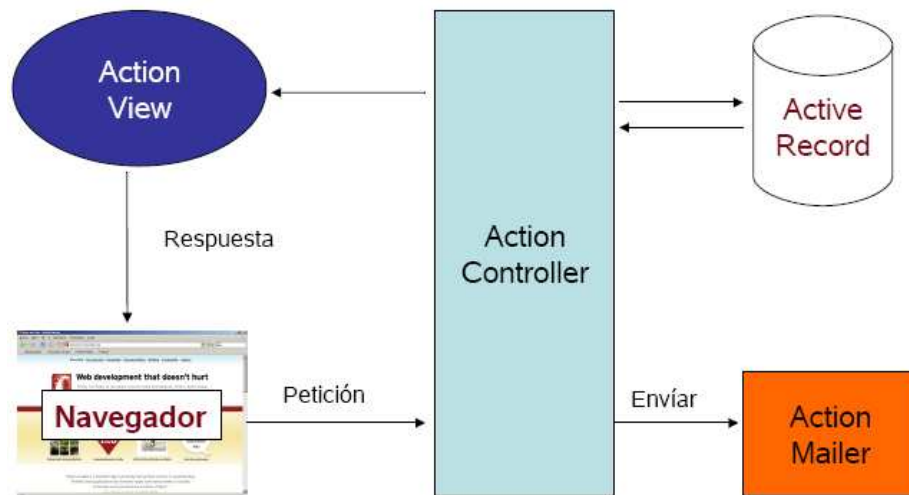


Figura N° III. 3. Modelo – Vista – Controlador

3.2.1.2. Características de Ruby on Rails

Programación Orientada a objetos (POO)

- Todo es un objeto.
- Características de la POO (herencia, clases, métodos).
- Métodos singleton.
- Iteradores y cierres.

Programación dinámica

- Son innecesarias las declaraciones de variables.
- Las variables no tienen tipo.
- La sintaxis es simple y consistente.
- La gestión de memoria se realiza automáticamente.

Programación potente y flexible

- Enteros de precisión múltiple.
- Manejo de Excepciones.
- Tratamiento de expresiones regulares.
- Carga dinámica.
- Hilos.
- Portable (Windows, Linux y Mac).
- Capacidades de introspección, reflexión y metaprogramación.

Programación interpretado

- Posibilidad de llamadas directas al sistema operativo.
- Potentes expresiones sobre cadenas de caracteres y expresiones regulares.
- Facilidades de desarrollo y desarrollo incremental.

Mapeo automático de objetos a modelo relacional

- Al crear una clase de tipo Model, Rails automáticamente infiere las propiedades de la misma, mapeándolas a los campos de la tabla de la base de datos. Rails sabe a qué tabla corresponde cada clase del Model gracias a las convenciones de nombres.
- Lo mejor de esto es que, al agregar un nuevo campo a una tabla, no hace falta tocar nada en el Model; la nueva propiedad estará disponible automáticamente.

Caching, validación y callbacks, transacciones, testing, generadores, seguridad

- Rails se definen las reglas en el Modelo.
- Una de las opciones más útiles y potentes son los validadores. Mediante esta opción podremos verificar la coherencia e integridad de los datos.
- Rails nos proporciona 3 eventos de validación: `Validate`, `Validate_on_create` y `Validate_on_update` donde podremos situar todo el código que deseemos para asegurarnos de la coherencia de los datos recibidos con los esperados (números, fechas, nombres, etc...). Además nos proporciona unos cuantos validadores predefinidos (helpers) que simplificarán las tareas de validación como ejemplo (`validates_length_of` ó `validates_numericality_of`).
- Los callbacks son a Rails como los Triggers a las base de datos. Existen una serie de métodos del tipo:
- `before_validation`, `after_validation`, `before_save`, `after_update`, `before_create`, `after_save`, etc.

3.2.1.3. Arquitectura de Rails

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



Figura N° III. 4. Arquitectura de Ruby on Rails

No te repitas (DRY) 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.

La repetición innecesaria de conocimiento es fuente de errores (incongruencias).
Ejemplo: Definir atributos de una clase.

Convención sobre configuración (COC) significa que el programador sólo necesita definir aquella configuración que no es convencional, por ejemplo, si hay una clase Historia en el modelo, la tabla correspondiente de la base de datos es historias, pero si la tabla no sigue la convención (por ejemplo blogposts) debe ser especificada

manualmente (`set_table_name "blogposts"`), 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). En vez de requerir innumerables archivos de configuración, Rails propone defaults razonables. Que, por supuesto, podemos cambiar (si lo necesitamos).

Ejemplo:

Clase Person ↔ tabla People

Se puede forzar: `set_table_name 'Personas'`

Rails establece una estructura de directorios relativamente rígida

De esta forma, puede encontrar las cosas a través de convenciones

Modelo – Vista – Controlador (MVC)

MVC son las siglas de Modelo, Vista y Controlador. Es el patrón de diseño de software muy común en programas interactivos orientados a objetos.

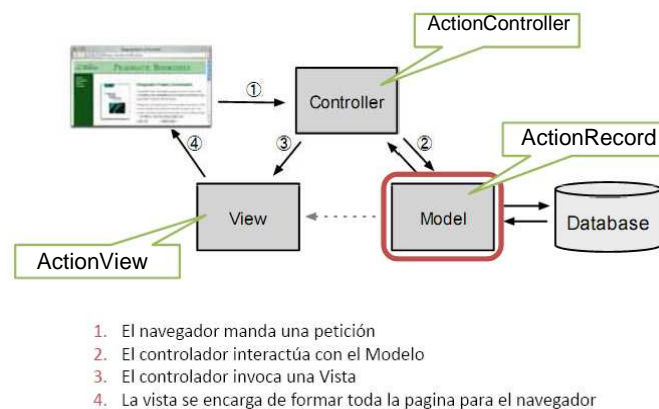


Figura N° III. 5. Patrón Modelo – Vista – Controlador (MVC) Simple

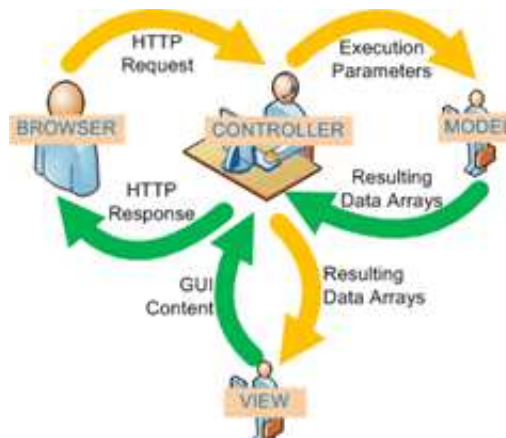


Figura N° III. 6. Petición de una Vista

Bajo el patrón de diseño de MVC, dentro de un programa, cada dominio lógico de edición (por ejemplo datos personales, cuentas bancarias, artículos noticiosos, etcétera) necesita tres partes:

- **Modelo:** ActiveRecord

Que es una representación de objetos del dominio lógico. En el ejemplo de datos personales, objetos como Persona, Dirección y Teléfono; es decir es el responsable de mantener el estado de la aplicación.

Ejemplo:

```
class Alumno < ActiveRecord::Base
  has_many :cursos
  def socio?
    not num_socio.nil?
  end
end
```

- **Controlador Lógica de la aplicación:** ActionController

Que interpreta la información que el usuario provee para hacer cambios en el modelo; es decir organiza la aplicación, recibe eventos del exterior, interactúa con el modelo y actualiza la información de las vistas.

Datos situado entre el ActiveRecord y ActiveView, proporciona mecanismos para la manipulación y organización de los datos y la entrada de estos a las formas Web.

Ejemplo:

```
class CursoController < ApplicationController
  layout 'base'
  def nueva_accion
    @curso = Curso.find(params[:id])
  end
end
```

- **Vista Interfaz (Web) de usuario:** ActionView

Que es una o varias piezas de código que formatean y muestran todo o parte del modelo en el dispositivo interactivo (típicamente la pantalla). La vista típicamente es notificada de cambios en el modelo; es decir es la responsable de presentar la interfaz y la información al usuario.

Ejemplo:

```
<h1>Añadir alumno del curso <%= @curso.titulo %></h1>
<% Alumno.find_all(nil, 'nombre').each do |a| %>
<%= radio_button('alumno', 'elegido', a.id) %>
<% end %>
```

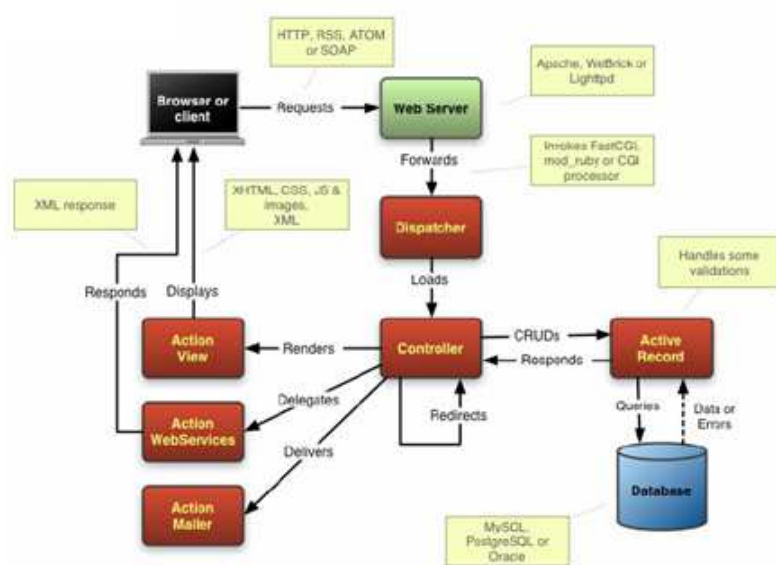


Figura N° III. 7. Patrón Modelo – Vista – Controlador – RoR Aplicación Web

Patrón ActiveRecord

- ActiveRecord es una librería de Rails encargada de mapear los objetos de negocio en las la base de datos.
- Crear un sistema de clases que podemos utilizar en nuestra aplicación
- Implementa el patrón Object Relational Mapping (ORM)
- Se pueden declarar relaciones con otros modelos/tablas
- Asociación automática entre columnas y atributos de la clase
- Las tablas son clases
- Las filas son objetos
- Las columnas son atributos
- Minimizar el número de líneas y maximizar la productividad
- Operadores CRUD
- Métodos Find()
- Relaciones (Uno a Uno – Uno a Muchos – Muchos a Muchos)

- Agregación
- Herencia de Tabla Simple
- Transacciones
- Validaciones
- Callbacks
- Un observer responde eventos sobre un objeto

Sintáxis de ActiveRecord

```
class Persona < ActiveRecord::Base
end
```

CRUD

Crear

```
p = Persona.new
p.nombre = "Wilian"
p.ciudad = "Riobamba"
p = Persona.create(:nombre => "Wilian", :ciudad => "Wilian")
p.save
```

Leer – Buscar

```
persona = Persona.find(3)
personas = Persona.find(:all, :conditions => "ciudad = 'Riobamba'")
persona = Persona.find_by_nombre("Wilian")
```

Búsqueda Múltiple

```
Persona.find_by_nombre_and_ciudad("Wilian", "Riobamba")
select * from personas where (nombre = "Wilian")
```

Actualizar – Modificar

```
persona = Persona.find_by_nombre("Wilian")
persona.ciudad = "Trujillo"
persona.save
```

Eliminar

```
persona = Persona.find_by_nombre("Wilian")
persona.destroy
```

Relaciones

```
class Proyecto < ActiveRecord::Base
  belongs_to :cliente
  has_one :gerente (Uno a Uno)
  has_many :milestones (Uno a Muchos)
  has_and_belongs_to_many :categories (Muchos a Muchos)
end
```

Validaciones

```
class Persona < ActiveRecord::Base
  validates_presence_of :nombre, :apellidos
  validates_length_of :codigo_postal, :is => 5
  validates_uniqueness_of :nif
  validates_format_of :codigo_postal, :with => /^[0-9]{5}$/
end
```

ActionView

- En principio, hay una plantilla por método
- Se encarga de convertir los datos que le pasa el controlador en el HTML que se servirá al navegador

Ejemplo:

```
<html>
<body>
<table><tr><td>Nombre</td><td>Apellidos</td> <td>E-mail</td></tr>
  <% @personas.each do |persona| %>
    <tr>
      <td><%= persona.nombre %></td>
      <td><%= persona.apellidos %></td>
    </tr>
  <% end %>
</table>
</body>
</html>
```

ActionController

- Procesa la URL solicitada
- Dirige la petición al controlador correspondiente
- El controlador realiza la tarea correspondiente, solicitando al Modelo los datos que necesite
- Renderiza la plantilla (la Vista)
- ActionController usa la ruta para decidir:
 - Qué controlador se usará
 - Qué método del mismo se ejecutará
 - Qué parámetros se le pasarán (array params)

Ejemplo

```
class AgendaController < ApplicationController
  def lista
    @personas = Persona.find(:all)
  end
  def mostrar
    @persona = Persona.find(params[:id])
  end
end
```

3.2.1.4. Licencias de la Tecnología Ruby on Rails

El intérprete y las bibliotecas están licenciadas de forma dual (inseparable) bajo las licencias libres y de código abierto GPL y Licencia Ruby.

3.2.1.5. Entorno de Programación de Ruby

Debido a que Ruby es un lenguaje de código abierto, ha sido adecuado para ejecutarse en muchas plataformas y arquitecturas diferentes. Esto quiere decir que si desarrollamos

un programa en Ruby en una máquina, es probable que puedas ejecutarlo sin cambios en una máquina diferente.

Podemos usar Ruby, en una forma u otra, en casi todos los sistemas operativos y plataformas. En Ruby, todos los archivos fuente tienen la extensión `.rb`, existe diferencia entre mayúsculas y minúsculas (Case Sensitive); además los comentarios son con el símbolo `#` hasta el final de la línea en que aparece, es ignorado por el intérprete.

Para facilitar bloques de comentarios, el intérprete ignora también todo el texto que aparezca entre una línea que comience con `=begin` y una que termine con `=end`.

Operadores en Ruby

Un operador es un símbolo formado por uno o más caracteres que permite realizar una determinada operación entre uno o más datos y produce un resultado. En Ruby, los números que no contienen punto decimal se llaman integres (enteros) y los que si contienen punto decimal se llaman números de punto flotante o simplemente floats.

Ejemplo:

```
puts 1 + 2      puts 2 * 3      puts 3 / 2      puts 10 - 11
```

Los números enteros en Ruby son objetos de la clase `Fixnum` o `Bignum`. Los números decimales son objetos de la clase `Float`.

Tabla N° III. 1. Operadores en Ruby

OPERADOR	DESCRIPCIÓN
[]	Array reference
[]=	Array element set
**	Exponentiation
!	Not
~	Complement
+	Unary plus
-	Minus
*	Multiply
/	Divide
%	Modulo
+	Plus
-	Minus
>>	Right shift
<<	Left shift
&	Bitwise And
^	Bitwise exclusive Or (Xor)
	Regular Or
<=	Less than or equal to
<	Less than
>	Greater than
>=	Greater than or equal to
<=>	Less than, equal to, greater than
==	Equal to
===	Tests equality in a when clause of a case statement
!=	Not equal to
=~	Regular expression pattern match
&&	Logical And
	Logical Or
..	Inclusive range
...	Exclusive range
?	Ternary if
:	Else
=	Normal assign
%=	Modulus and assign
/=	Divide and assign
-=	Subtract and assign
+=	Add and assign
*=	Multiply and assign
**=	Exponent and assign
defined?	True if symbol defined
not	Logical negation
and - or	Logical composition
If – unless – while – until	Statement modifiers
begin/end	Block expression

Cadenas en Ruby

Las cadenas son secuencias de caracteres entre comillas sencillas o dobles. Una cadena vacía es sólo un par de comillas sin más caracteres ''.

Ejemplo:

```
# Podemos usar " o ' para las cadenas
```

```
puts "Hola Mundo"
```

```
puts 'Hola Mundo'
```

```
# Las cadenas pueden concatenarse
```

```
puts 'A mi me gusta' + ' Ruby'
```

```
# Podemos reproducir las cadenas un número n de veces
```

```
puts 'Hola' * 3
```

La diferencia entre comillas simples (') y dobles (") es importante. Si el intérprete de Ruby encuentra comillas simples, no intenta hacer sustituciones en la cadena. Por otro lado, si encuentra comillas dobles, va a buscar en la cadena elementos que puede sustituir.

Ejemplo:

```
# Con comillas simples
```

```
puts 'El resultado de sumar 2 + 2 es #{2+2}'
```

```
# Con comillas dobles
```

```
puts "El resultado de sumar 2 + 2 es #{2+2}"
```

Dentro de las cadenas delimitadas con comillas dobles, podemos insertar código Ruby dentro de los caracteres #{ } y va a ser ejecutado por el intérprete.

Tabla N° III. 2. Funciones de Cadenas

CADENA	DESCRIPCIÓN
reverse	Regresa una copia de la cadena original con los caracteres en orden inverso. No modifica la cadena original.
length	Regresa el número de caracteres de la cadena (incluyendo espacios en blanco).
uppercase	Regresa una copia de la cadena cambiando todos los caracteres a letras mayúsculas.
downcase	Regresa una copia de la cadena cambiando todos los caracteres a letras minúsculas.
capitalize	Regresa una copia de la cadena con el primer caracter en mayúscula.
Slice	Regresa una subcadena de la cadena original.
Swapcase	Regresa una copia de la cadena invirtiendo las letras mayúsculas y minúsculas.

Expresiones regulares

Las expresiones regulares son caracteres y combinaciones de caracteres que tienen un significado especial:

Tabla N° III. 3. Caracteres especiales en expresiones regulares

SÍMBOLO	DESCRIPCIÓN
[]	Especificación de rango. (p.e. [a-z] representa una letra en el rango de la a a la z.
\w	Letra o dígito; es lo mismo que [0-9A-Za-z]
\W	Ni letra, ni dígito
\s	Espacio, es lo mismo que [\t\n\r\f]
\S	No espacio
\d	Dígito; es lo mismo que [0-9]
\D	No dígito
\b	Backspace (0x08) (sólo si aparece en una especificación de rango)
\B	Límite de palabra (sólo si no aparece en una especificación de rango)
\b	No límite de palabra
*	Cero o más repeticiones de lo que precede
+	Una o más repeticiones de lo que precede
[m,n]	Al menos m y como máximo n de lo que precede
?	Al menos una repetición de lo que precede; es lo mismo que [0,1]
	Puede coincidir con lo que precede o con lo que sigue
()	Agrupamiento

Por ejemplo, supongamos que queremos comprobar si una cadena se ajusta a esta descripción: "Comienza con una f minúscula, a la que sigue exactamente una letra mayúscula y opcionalmente cualquier cosa detrás de ésta, siempre y cuando no haya más letras minúsculas."; es necesario solicitar que se verifique la cadena contra la siguiente expresión regular `/^f[A-Z][^a-z]*$/`.

Arrays en Ruby

Se pueden crear un array listando elementos entre corchetes (`[]`) y separándolos por comas. Los arrays en Ruby pueden almacenar objetos de diferentes tipos. Los índices negativos en un array indican que se empieza a contar desde el final del array, en vez del principio.

Ejemplo:

```
# Arrays
ary = [1, 2, "3"]
[1, 2, "3"]
# Un array vacío
var1 = []
# Un array cuyos elementos hacen referencia a tres objetos diferentes
var4 = [80.5, sabor, [true, false]]
puts var4[2]
# podemos incluso agregar un array a otro array
nombre[6] = [1, 2, 3]
puts nombre[6]
```

Hashes

Un array asociativo contiene elementos que se pueden acceder, no a través de índices numéricos secuenciales, sino a través de claves que pueden tener cualquier tipo de valor. Estos arrays se conocen a veces como hash o diccionario; en el mundo Ruby se prefiere el término hash. Los hash se pueden crear mediante pares de elementos dentro

de llaves ({ }). Se usa la clave para encontrar algo en un hash de la misma forma que se utiliza el índice para encontrar algo en un array.

Ejemplo:

```
h = { 1 => 2, "2" => "4" }
      {"2"=>"4", 1=>2}
      h[1]
        2
      h["2"]
        "4"
      h[5]
        nil
      h[5] = 10 # añadimos un valor
        10
h={5=>10, "2"=>"4", 1=>2}
  h[1]=nil # borramos un valor
  h={5=>10, "2"=>"4", 1=>nil}
```

Estructuras de Control

Las instrucciones condicionales son instrucciones que permiten ejecutar bloques de instrucciones sólo si se da una determinada condición.

Instrucción if ... else ... end

En Ruby, sólo nil y false son evaluados como falso, todo lo demás (incluyendo 0), es verdadero. En Ruby nil es un objeto.

Síntaxis:

```
if conditional [then] code [elsif conditional [then] code ]... [else code ]
end
```

Ejemplo:

```
if 5 > 6
  puts 'SI'
else
  puts 'NO'
end
```

Algunos operadores condicionales comunes son: ==, != <=, >=, <, >. Ruby tiene también una forma negativa de if. La estructura unless comienza con la palabra unless y termina con end. El cuerpo es el texto que aparece en medio de las dos. A menos que la expresión que continúe a la palabra unless sea verdadera, el cuerpo es ejecutado, de lo contrario el intérprete lo ignora.

Sintaxis:

```
unless conditional [then]
  code
[else code ] end
```

Instrucción case valor when 0 when 1 end

Se utiliza la sentencia case para comprobar una secuencia de condiciones. Superficialmente se parece al switch de C y Java pero es considerablemente más potente en Ruby. Esta estructura funciona de manera muy similar a una serie de expresiones if, te permite enumerar una serie de condiciones y ejecutar una expresión que corresponda al primer valor que sea verdadero.

Sintaxis:

```
case expression
[when expression [, expression ...] [then]
  code ]...
[else code ]
end
```

Ejemplo:

```
year = 2000
leap = case year
  when year % 400 == 0: true
  when year % 100 == 0: false
  else year % 4 == 0
end
puts leap
```

Instrucción while condición ... end

Un while es un if repetido. Se ha utilizado while condición ... end que rodeaba el código a repetir mientras la condición fuera cierta. Pero while e if se pueden aplicar fácilmente a sentencias individuales.

Sintaxis:

```
while conditional [do] code end
```

Ejemplo:

```
a = 8
While a < 10
    a += 1
    puts 'a'
end
```

Existen cuatro formas de interrumpir el progreso de un bucle desde su interior son:

- El primer **break**, como en C, sale completamente del bucle.
- La segunda **next** salta al principio de la siguiente iteración del bucle (se corresponde con la sentencia **continue** del C).
- La tercera **redo** reinicia la iteración en curso.
- La cuarta forma de salir del interior de un bucle es **return**. La evaluación de **return** provoca la salida no sólo del bucle sino también del método que contiene el bucle. Si se le pasa un argumento, lo devolverá como retorno de la llamada al método, si no el retorno será **nil**.

Instrucción for elememto in colección ... end

El for es realmente otra forma de escribir each. La siguientes dos estructuras son equivalentes: `for num in (4..6) print num, "\n" end`

Métodos en Ruby

En Ruby, se llama a un método con la notación punto (como en C++ o Java). El objeto con el que nos comunicamos se nombra a la izquierda del punto.

Ejemplo:

```
"abcdef".length  
6
```

Intuitivamente, a este objeto cadena se le está pidiendo que diga la longitud que tiene.

Técnicamente, se está llamando al método `length` del objeto `"abcdef"`.

Clases

El mundo real está lleno de objetos que podemos clasificar. Por ejemplo, un niño muy pequeño es probable que diga "guau guau" cuando vea un perro, independientemente de su raza; naturalmente vemos el mundo en base a estas categorías.

En terminología Orientado a Objetos (OO), una categoría de objetos, como "perro", se denomina clase y cualquier objeto determinado que pertenece a una clase se conoce como instancia de esa clase. Generalmente, en Ruby y en cualquier otro lenguaje OO, se define primero las características de una clase, luego se crean las instancias. Para mostrar el proceso, definamos primero una clase muy simple **Perro**.

Ejemplo:

```
class Perro  
  def ladra  
    print "guau guau\n"  
  end  
end
```

En Ruby, la definición de una clase es la región de código que se encuentra entre las palabras reservadas **class** y **end**. Dentro de esta área, **def** inicia la definición de un método, corresponde con algún comportamiento específico de los objetos de esa clase.

Ahora que tenemos definida la clase **Perro**, vamos a utilizarla:

```
rufi = Perro.new
```

Hemos creado una instancia nueva de la clase **Perro** y le hemos llamado **rufi**. El método **new** de cualquier clase, crea una nueva instancia. Dado que **rufi** es un **Perro**, según la definición de la clase, tiene las propiedades que se decidió que un **Perro** debía tener. Dado que la idea de Perro es muy simple, sólo hay una cosa que puede hacer **rufi**

Ejemplo:

```
rufi.ladra  
guau guau
```

Herencia en Ruby

La clasificación de los objetos en nuestra vida diaria es evidentemente jerárquica. Sabemos que todos los gatos son mamíferos y que todos los mamíferos son animales. Las clases inferiores heredan características de las clases superiores a las que pertenecen. Si todos los mamíferos respiran, entonces los gatos respiran. Este concepto se puede expresar en Ruby:

```
class Mamifero  
  def respira  
    print "inhalar y exhalar\n"  
  end  
end  
  
class Gato<Mamifero  
  def maulla
```



```
        print "miau \n"  
    end  
end  
  
Gato = Gato.new  
Gato.respira  
inhalar y exhalar
```

Módulos

Los módulos en Ruby son similares a las clases, excepto en:

- Un módulo no puede tener instancias
- Un módulo no puede tener subclases
- Un módulo se define con **module ... end**

Ciertamente la clase `Module` de un módulo es la superclase de la clase `Class` de una clase. Existen dos usos típicos de los módulos. Uno es agrupar métodos y constantes relacionadas en un repositorio central.

El módulo `Math` de Ruby hace esta función:

```
Math.sqrt(2)  
    1.414213562  
  
Math::PI  
    3.141592654
```

Variables en Ruby

Las variables pueden contener datos de todo tipo y pueden ser usadas sin necesidad de declararlas. El nombre mismo de la variable denota su alcance (local, global, de instancia, etc.). Las variables y las constantes no tienen tipo. Aunque las variables sin

tipo tienen sus inconvenientes, presentan más ventajas y se adaptan mejor a la filosofía rápida y sencilla de Ruby.

- El nombre de una **variable local** consiste de una letra minúscula (o un guión bajo) y un número arbitrario de letras minúsculas, mayúsculas, números o guiones bajos. `sunil`, `_z`, `numero_de_veces` son ejemplos de variables locales.
- Las **variables de instancia** (instance variables) comienzan con el símbolo arroba (`@`) seguidas por letras minúsculas, mayúsculas, guiones bajos o números. `@signo`, `@_`, y `@contador` son ejemplos de variables de instancia.
- El nombre de las **variable de clase** (class variables) comienza con dos símbolos arroba (`@@`) seguidos por letras mayúsculas, minúsculas, números o guiones bajos. Por ejemplo: `@@signo`, `@@_`, `@@contador`.
- Las **variables globales** (global variables) comienzan con un símbolo de dolar (`$`) seguidas de letras mayúsculas, minúsculas, números, guiones bajos o un guión (`-`) seguido de cualquier carácter. `$contador`, `$CONTADOR`, `$-x` son ejemplos de variables globales.
- En la mayoría de los lenguajes hay que declarar las variables para especificar su tipo, si se pueden modificar (si son constantes) e indicar su ámbito.

Tabla N° III. 4. Clases de variables

VARIABLE	DESCRIPCIÓN
<code>\$</code>	Variable global
<code>@</code>	Variable instancia
<code>[a-z]</code> ó <code>_</code>	Variable local
<code>[A-Z]</code>	Constante
<code>@@</code>	Variable de clase

Las únicas excepciones a lo expuesto en la tabla son las pseudo-variables de Ruby: **self**, que referencia al objeto que está en ese momento en ejecución y **nil** que es el valor nulo

que toman las variables no inicializadas. Ambos tienen un identificador como de variable local pero **self** es una variable global que la mantiene el interprete y **nil** es una constante. No se debe asignar valores a **self** y **nil** principalmente porque un valor de **self** referencia al objeto de nivel superior

Tabla N° III. 5. Variables de Sistema

VARIABLE	DESCRIPCIÓN
#!	Último mensaje de error
\$@	Posición del error
\$_	Última cadena leída con gets
\$.	Último número de línea leído por el interprete
\$&	Última cadena que ha coincidido con una expresión regular
\$~	Última cadena que ha coincidido con una expresión regular como array de subexpresiones
\$n	La n-ésima subexpresión regular de la última coincidencia (igual que \$~[n])
\$=	flag para tratar igual las mayúsculas y minúsculas
\$/	Separador de registros de entrada
\$\	Separador de registros de salida
\$0	El nombre del fichero del guión Ruby
\$*	El comando de la línea de argumentos
\$\$	El número de identificación del proceso del intérprete Ruby
\$?	Estado de retorno del último proceso hijo ejecutado

Constantes en Ruby

Los nombres de constantes comienzan con una letra mayúscula seguida de un número arbitrario de letras mayúsculas, minúsculas, números o guiones bajos. Los nombres de clases y módulos son constantes y deben seguir las convenciones de los nombres de constantes. Por ejemplo: `module Matematicas`, `PI=3.1416`, `class Animales`. Una constante tiene un nombre que comienza con una letra mayúscula. Se le debe asignar valor sólo una vez. En la implementación actual de Ruby, reasignar un valor a una constante genera un aviso y no un error.

Procesamiento de excepciones

rescue

Para ejecutar una posible vía de escape en caso de error. Un programa en ejecución puede encontrarse con problemas inesperados. Por ejemplo podría no existir un fichero que desea leer, al salvar algunos datos se podría llenar un disco, un usuario podría introducir algún tipo de datos de entrada poco adecuados. Dentro del código de **rescue** se puede utilizar **retry** para intentar de nuevo el código en **begin**. Esto nos permite reescribir el ejemplo anterior de una forma más compacta, con **retry** se debe tener mucho cuidado ya que se puede entrar en bucles infinitos, también se puede utilizar **raise** para levantar una excepción.

ensure

Que nos garantiza que el código de ese bloque se ejecutará pase lo que pase. Por esta razón se añadió otra palabra reservada a esquema **begin ... rescue ... end, ensure**. El bloque de código de **ensure** se ejecuta independientemente del éxito o fracaso del bloque de código en **begin**. Se puede utilizar **ensure** sin **rescue** y viceversa, pero si se utilizan en el mismo bloque **begin ... end, rescue** debe preceder a **ensure**.

Tabla N° III. 6. Control de Excepciones

NOMBRE	DESCRIPCIÓN
raise	Para levantar una excepción.
rescue	Para ejecutar una posible vía de escape en caso de error.
retry	Que puede ejecutarse dentro de un rescue para volver a ejecutar el código que falló.
ensure	Que nos garantiza que el código de ese bloque se ejecutará pase lo que pase.

Estructura de Directorios de una Aplicación Rails

Rails trata por todos los medios de minimizar el número de decisiones que tienes que tomar así como de eliminar el trabajo innecesario por ende Rails crea toda la estructura de directorios. Rails sabe dónde encontrar todo lo que necesita dentro de esta estructura de forma que luego no tengamos que indicarle donde se encuentra.

Tabla N° III. 7. Estructura de Directorios de Rails

DIRECTORIOS	DESCRIPCIÓN
app	Contiene subdir model (clases derivadas de Active Record, view (.rhtml y layouts), controller (clases derivadas de ApplicationController) y helpers.
config	Contiene routers (routes.rb), setup de BD (database.yml – bd_test, bd_integration y bd_production) y environment.
db	Migraciones de BD.
doc	Documentación autogenerada.
lib	Reservado para tasks anexables a Rake, y código compartido.
public	Contiene dispatchers, aceptan y redireccionan los request provenientes de los browsers (dispatch.cgi, dispatch.fcgi, dispatch.rb).
script	Contiene scripts utilitarios (generate, server, console, etc.).
test	Reservado para pruebas: unitarias, de integración y funcionales.
app	Contiene subdir model (clases derivadas de Active Record, view (.rhtml y layouts), controller (clases derivadas de ApplicationController) y helpers.
controllers	Es donde Rails espera encontrar las clases de nuestros controladores. La misión del controlador es manejar una petición Web recibida desde el navegador, es decir, del usuario.
views	Contiene las plantillas de visualización que se rellenarán con datos de nuestra aplicación, serán convertidas a HTML y devueltas al navegador del usuario.
models	Contiene las clases que modelan los datos almacenados en la base de datos de nuestra aplicación.
helpers	Se guardan las clases de ayuda que se usan para asistir a las clases de modelos, vistas y controladores. Dichas clases se mantengan sin crecer de tamaño, dedicadas y ordenadas.

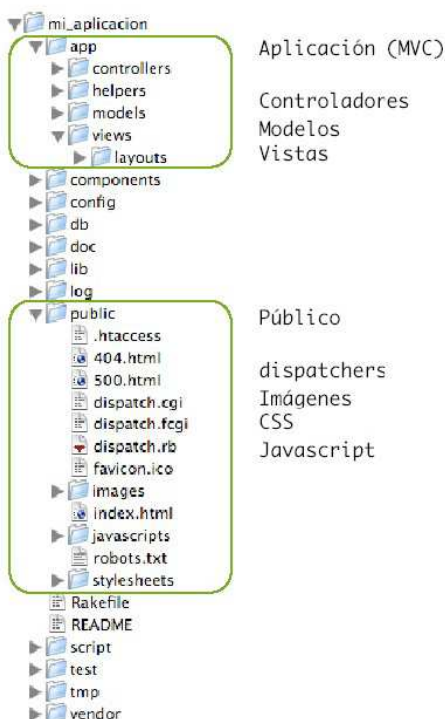


Figura N° III. 8. Estructura de Directorios de Rails

3.2.2. Tecnología Mono

En Diciembre del 2000 Miguel de Icaza (Co-fundador de la empresa Ximian, fundador y presidente de la GNOME Foundation). El proyecto mono tiene como objetivo crear una implementación libre de algunas herramientas y parte de la arquitectura de .NET. Parte de esta tecnología se basa en un estándar propuesto a la ECMA y lo interesante es que tiene ideas muy buenas.

Oficialmente la fecha del lanzamiento del proyecto es el 9 de Julio del 2001, aunque realmente ya llevaba algún tiempo en marcha debido a que el propio Miguel ya había empezado a desarrollar un compilador de C#, curiosamente usando C# para familiarizarse con el lenguaje a la vez, y Ximian también vio interesantes los beneficios

de la plataforma y asignó a su laboratorio de investigación la tarea de colaborar en el proyecto.

Básicamente el proyecto MONO ha comenzado con un compilador de C#. La idea inicial es explorar hasta que punto el lenguaje C# y la arquitectura propuesta para su máquina virtual (VM) puede ser una opción que mejore las condiciones de producción de los desarrolladores de software libre. Mono tiene implementación libre de los estándares ECMA/ISO, Ecma-334 (lenguaje de programación C#), Ecma-335 (Common Language Infrastructure), es patrocinado por Novell Inc, Biblioteca de clases: MIT X11, Máquina Virtual LGPL y Compilador de C#: GPL

Mono es la versión open source de la plataforma .NET de Microsoft, teniendo como herramienta principal el compilador de C# , el soporte para páginas ASP.NET bajo Linux y una serie de herramientas para desarrollar distintos tipos de programas de cómputo tal como aplicaciones gráficas de escritorio, aplicaciones de consola y Formularios Web. Mono tiene implementados un servidor de paginas aspx llamado XSP, el cual se puede integrar con Apache utilizando el módulo mod_mono.

3.2.2.1. ¿Qué es Proyecto Mono?

Mono es un proyecto de implementación del Framework .NET de Microsoft utilizando código libre (open source), gestionado por Ximian y basado en las especificaciones definidas en ECMA, de plataforma libre de desarrollo multiplataforma.

3.2.2.2. Características de Mono

- Un compilador para el lenguaje C#, Visual Basic.Net y JScript
- Un entorno de ejecución virtual: Un compilador JIT (Just-In-Time = justo-a-tiempo, esto es, que compila el código justo antes de ser ejecutado), un compilador AOT (AOT=ahead-of-time, antes-de-tiempo , esto es, que compila a código nativo un archivo y de esta forma no necesita la compilación JIT cada vez que se ejecute el programa), gestión automática de memoria, un interprete (mint), motor multiproceso.
- Una máquina virtual para los bytecodes del Lenguaje Intermedio Común (CLI).
- Una implementación de la librería de clases de .NET: manipulación XML, Entrada/Salida, funciones matemáticas, criptografía, xslt, etc.
- Librería de clases multiplataforma para el acceso a bases de datos: Postgress, MySQL, DB2, TDS, Sybase, Oracle, ODBC y Gnome-GDA.
- Librería de clases UNIX: Mono.Posix
- Librería de clases GNOME: la familia Gtk# .
- MCS (Mono Compiler Suite) es el Compilador Mono de C#, es posible compilar, embeber recursos y enlazar programas Mono usando mcs

3.2.2.3. Arquitectura de Mono

- **Common Language Infrastructure (CLI).**

Se trata del componente más importante dentro de la plataforma. Su labor es proporcionar una plataforma independiente del lenguaje para el desarrollo y ejecución de aplicaciones, incluyendo componentes encargados del manejo de excepciones, la seguridad, la interoperabilidad y la recogida de basura (basura en

cuanto memoria nos referimos). La implementación de Microsoft para su CLI, es el denominado CLR (Common Language Runtime) o Lenguaje Común en Tiempo de Ejecución.

- **Assemblies**

Son la unidad de desarrollo dentro de .NET. El código intermedio es albergado dentro de ellos por lo que se constituyen dentro de plataformas Windows como ejecutables del tipo .exe o .dll. Una de las mayores ventajas que conlleva el uso de ensamblados en sistemas Windows a la hora de desarrollar y ejecutar software, es alejarnos de las dlls.

- **Metadata**

Se refiere a una información que describe al código intermedio (CIL, Common Intermediate Language).

- **Basic Class Library (BCL)**

La librería de clases básica, es por así decirlo la recopilación de todas las clases que albergan a las funciones necesarias para casi todo lo que un programador puede necesitar, incluyendo funciones de alto nivel. Por ejemplo proporciona funcionalidad para lectura/escritura desde flujos, interacción con bases de datos, renderización gráfica, tratamiento de ficheros XML.

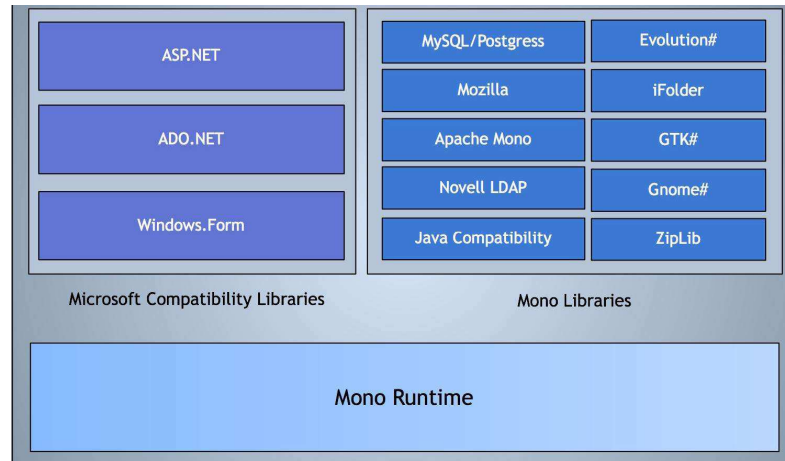


Figura N° III. 9. Estructura de Mono

3.2.2.4. Licencias de la tecnología Mono

Mono usa tres tipos de licencia:

- Las librerías de clases están bajo una licencia X11. Este tipo de licencia permite prácticamente cualquier uso del código, incluido copiarlo y usarlo en una aplicación propia. El único requisito es que se mantenga la información de copyright de los archivos. No es necesario siquiera indicar que se está utilizando software bajo licencia X11.
- Las librerías del runtime, como el JIT, se distribuyen bajo licencia LGPL. De esta forma si se realiza un programa que linke con ellas, se puede mantener bajo código propietario. Sin embargo es necesario permitir que se pueda enlazar con versiones más recientes de las librerías. La forma más fácil de hacer esto sería linkando dinámicamente con ellas. Esto obligaría al usuario a descargar e instalar mono por separado.
- En lo que respecta al cumplimiento de las patentes de software, las partes contempladas en el estándar ECMA no tienen ningún problema ya que se permite a cualquiera implementar esos componentes gratuitamente y para

cualquier propósito. Sin embargo, cuestiones referentes a ADO.NET, ASP.NET y WinForms, son bastante diferentes. La estrategia de Mono ante estas tecnologías es la siguiente:

- Evitar la patente utilizando otros mecanismos para implementar la misma funcionalidad.
- Eliminar las porciones de código bajo patente.
- Encontrar algo más novedoso que deje en sin uso a la patente.

3.2.2.5. Entorno de Programación de la tecnología Mono

Un comentario es texto que se incluye en el código fuente para facilitar su lectura a los programadores y cuyo contenido es, por defecto, completamente ignorado por el compilador. Suelen usarse para incluir información sobre el autor del código, para aclarar el significado o el porqué de determinadas secciones de código, para describir el funcionamiento de los métodos de las clases, etc. Hay dos formas de escribir comentarios. La primera consiste en encerrar todo el texto que se desee comentar entre caracteres `/*` y `*/`; `//`. Estos comentarios pueden abarcar tantas líneas como sea necesario. Mono puede utilizar varias librerías de widgets:

- **GTK#**
 - GTK# es el binding para Mono de las populares GTK+ (Gimp Tool Kit).
 - Íntimamente ligada al proyecto GNOMO.

- **wx.NET**
 - Es un wrapper para la CLI de wxWidgedts.

- wxWidgets es una librería de widgets cuya finalidad es proveer un API sencillo para crear GUIs en múltiples plataformas, utilizando el UI nativo de cada plataforma (Utiliza librerías nativas en cada plataforma ya sea Windows, GTK para GNU/Linux, etc.).
- Se puede ver en los screenshots.
- **Windows Forms**
 - Windows Forms es la librería de Widgets que viene por defecto en Microsoft.NET.
 - Su especificación no está recogida como estándar en el ECMA o ISO.
 - Mono está tratando más de implementar Windows Forms que DotGNU.
- **Cocoa#**
- **QT#**
- **Sharp WT**

Operadores en Mono

Un operador es un símbolo formado por uno o más caracteres que permite realizar una determinada operación entre uno o más datos y produce un resultado.

- **Operaciones aritméticas:** Los operadores aritméticos incluidos en Mono son los típicos de suma (+), resta (-), producto (*), división (/) y módulo (%). También se incluyen operadores de “menos unario” (-) y “más unario” (+).
- **Operaciones lógicas:** Se incluyen operadores que permiten realizar las operaciones lógicas típicas: “and” (&& y &), “or” (|| y |), “not” (!) y “xor” (^). Los operadores && y || se diferencian de & y | en que los primeros realizan evaluación perezosa y los segundos no. La evaluación perezosa consiste en que

si el resultado de evaluar el primer operando permite deducir el resultado de la operación, entonces no se evalúa el segundo y se devuelve dicho resultado directamente, mientras que la evaluación no perezosa consiste en evaluar siempre ambos operandos.

- **Operaciones relacionales:** Se han incluido los tradicionales operadores de igualdad (`==`), desigualdad (`!=`), “mayor que” (`>`), “menor que” (`<`), “mayor o igual que” (`>=`) y “menor o igual que” (`<=`)
- **Operaciones de asignación:** Para realizar asignaciones se usa en C# el operador `=`, operador que además de realizar la asignación que se le solicita devuelve el valor asignado. Por ejemplo, la expresión `a = b` asigna a la variable `a` el valor de la variable `b` y devuelve dicho valor, mientras que la expresión `c = a = b` asigna a las variables `c` y `a` el valor de `b` (el operador `=` es asociativo por la derecha).
- **Operadores de asignación compuestos** que permiten ahorrar tecleo a la hora de realizar asignaciones tan comunes como: `+=`, `-=`, `*=`, `/=`, `%=`, `&=`, `|=`, `^=`, `<<=` y `>>=`, otros dos operadores de asignación incluidos son los de incremento (`++`) y decremento (`--`) Estos operadores permiten, respectivamente, aumentar y disminuir en una unidad el valor de la variable sobre el que se aplican.

Estructuras de Control

Instrucción if

La instrucción `if` permite ejecutar ciertas instrucciones sólo si se da una determinada condición. Su sintaxis de uso es la siguiente:

```
if (<condición>) { <instruccionesIf> } else { <instruccionesElse> }
```

Instrucción switch

La **instrucción switch** permite ejecutar unos u otros bloques de instrucciones según el valor de una cierta expresión. Su estructura es:

```
switch (<expresión>
{ case <valor1>: <bloque1>
  case <valor2>: <bloque2>
default: <bloquedefault> }
```

Instrucción while

La instrucción **while** permite ejecutar un bloque de instrucciones mientras se de una cierta instrucción. Su sintaxis de uso es: **while** (<condición>) { <instrucciones> }

Por otro lado, dentro de las <instrucciones> de un **while** pueden utilizarse las siguientes dos instrucciones especiales:

- **break;**: Indica que se ha de abortar la ejecución del bucle y continuarse ejecutando por la instrucción siguiente al **while**.
- **continue;**: Indica que se ha de abortar la ejecución de las <instrucciones> y reevaluarse la <condición> del bucle, volviéndose a ejecutar las <instrucciones> si es cierta o pasándose a ejecutar la instrucción siguiente al **while** si es falsa.

Instrucción do...while

La instrucción **do...while** es una variante del **while** que se usa así:

```
do { <instrucciones> } while(<condición>);
```

Instrucción for

La instrucción for es una variante de while que permite reducir el código necesario para escribir los tipos de bucles más comúnmente usados en programación. Su sintaxis es:

```
for (<inicialización>; <condición>; <modificación>) { <instrucciones> }
```

Instrucción foreach

La instrucción foreach es una variante del for pensada especialmente para compactar la escritura de códigos donde se realice algún tratamiento a todos los elementos de una colección, que suele un uso muy habitual de **for** en los lenguajes de programación que lo incluyen. La sintaxis que se sigue a la hora de escribir esta instrucción **foreach** es:

```
foreach (<tipoElemento> <elemento> in <colección>) { <instrucciones> }
```

El significado de esta instrucción es muy sencillo: se ejecutan <instrucciones> para cada uno de los elementos de la <colección> indicada. <elemento> es una variable de sólo lectura de tipo <tipoElemento> que almacenará en cada momento el elemento de la colección que se esté procesando y que podrá ser accedida desde <instrucciones>.

Cadenas en Mono

Una cadena no es más que una secuencia de caracteres, representada como objetos de un tipo de dato string. Las cadenas de texto suelen crearse a partir literales de cadena o de otras cadenas previamente creadas. Ejemplos de ambos casos se muestran a continuación: **string** cadena1 = "José Antonio"; **string** cadena2 = cadena1;

Tabla N° III. 8. Funciones de Cadenas

NOMBRE	DESCRIPCIÓN
int IndexOf(string subcadena)	Indica cuál es el índice de la primera aparición de la subcadena indicada dentro de la cadena sobre la que se aplica, pues sólo si no la encuentra devuelve un -1 .
int LastIndexOf(string subcadena)	Funciona de forma similar a IndexOf() sólo que devuelve la posición de la última aparición de la subcadena buscada en lugar de devolver la de la primera.
string Insert(int posición, string subcadena)	Devuelve la cadena resultante de insertar la subcadena indicada en la posición especificada de la cadena sobre la que se aplica.
string Remove(int posición, int número)	Devuelve la cadena resultante de eliminar el número de caracteres indicado que hubiese en la cadena sobre al que se aplica a partir de la posición especificada.
string Replace(string aSustituir, string sustituta)	Devuelve la cadena resultante de sustituir en la cadena sobre la que se aplica toda aparición de la cadena aSustituir indicada por la cadena sustituta especificada como segundo parámetro.
string Substring(int posición, int número)	Devuelve la subcadena de la cadena sobre la que se aplica que comienza en la posición indicada y tiene el número de caracteres especificados. Si no se indica dicho número se devuelve la subcadena que va desde la posición indicada hasta el final de la cadena.
string ToUpper() y string ToLower()	Devuelven, respectivamente, la cadena que resulte de convertir a mayúsculas o minúsculas la cadena sobre la que se aplican.

Arrays en Mono

Un Array o vector es un tipo especial de variable que es capaz de almacenar en su interior y de manera ordenada uno o varios datos de un determinado tipo.

Para declarar vectores se usa la siguiente sintaxis:

```
<tipoDatos>[ ] <nombreTabla>;
```

Por ejemplo, una tabla que pueda almacenar objetos de tipo **int** se declara así:

```
int[ ] tabla = new int[100];
```

Ejemplo:

```
int[] tabla = new int[4];
```



```
tabla[0] = 5;

// Por defecto se inicializó a 0, luego ahora el valor de tabla[1] pasa a ser 1

tabla[1]++;

tabla[2] = tabla[0] - tabla[1];      // tabla[2] pasa a valer 4, pues 5-1 = 4

// El contenido de la tabla será {5,1,4,0}, // pues tabla[3] se inicializó por defecto a 0.
```

Métodos en Mono

Un método es un conjunto de instrucciones a las que se les da un determinado nombre de tal manera que sea posible ejecutarlas en cualquier momento sin tenerlas que reescribir sino usando sólo su nombre. A estas instrucciones se les denomina cuerpo del método, y a su ejecución a través de su nombre se le denomina llamada al método.

Sintaxis:

```
<tipoRetorno> <nombreMétodo>(<parámetros>) { <cuerpo> }
```

Ejemplo:

```
int Saluda(){

Console.WriteLine("Hola Mundo");

return 1;

}
```

En <tipoRetorno> se indica cuál es el tipo de dato del objeto que el método devuelve, y si no devuelve ninguno se ha de escribir **void** en su lugar. Como nombre del método se puede poner en <nombreMétodo> cualquier identificador válido. El <cuerpo> del

método también es opcional, pero si el método retorna algún tipo de objeto entonces ha de incluir al menos una instrucción **return** que indique cuál objeto.

Clases en Mono

Una clase es la definición de las características concretas de un determinado tipo de objetos. Es decir, de cuáles son los datos y los métodos de los que van a disponer todos los objetos de ese tipo. Por esta razón, se suele decir que el tipo de dato de un objeto es la clase que define las características del mismo.

Sintaxis: **class** <nombreClase> { <miembros> }

Ejemplo:

```
class Persona{  
    string Nombre; int Edad; string NIF;  
    void Cumpleaños() { Edad++; }  
}
```

Herencia en Mono

El mecanismo de **herencia** es uno de los pilares fundamentales en los que se basa la programación orientada a objetos. Es un mecanismo que permite definir nuevas clases a partir de otras ya definidas de modo que si en la definición de una clase indicamos que ésta deriva de otra, entonces la primera -a la que se le suele llamar **clase hija**- será tratada por el compilador automáticamente como si su definición incluyese la definición de la segunda -a la que se le suele llamar **clase padre** o **clase base**. Las clases que derivan de otras se definen usando la siguiente sintaxis:

```
class <nombreHija>:<nombrePadre> { ..... }
```

Variables en Mono

Las variables son "nombres" que pueden contener un valor, ya sea de tipo numérico como de cualquier otro tipo. Las variables representan un espacio donde alojar información. Toda variable tiene un tipo que determina qué valor puede ser almacenado en la variable. Las variables locales son variables que son declaradas dentro de métodos, propiedades o indexadores; las variables globales son las que se pueden utilizar en cualquier lugar del proyecto.

```
int a;    int b = 1;
```

Constantes en Mono

Una **constante** es una variable cuyo valor puede determinar el compilador durante la compilación y puede aplicar optimizaciones derivadas de ello. Para que esto sea posible se ha de cumplir que el valor de una constante no pueda cambiar durante la ejecución, por lo que el compilador informará con un error de todo intento de modificar el valor inicial de una constante. Las constantes se definen como variables normales pero precediendo el nombre de su tipo del modificador **const** y dándoles siempre un valor inicial al declararlas.

Sintaxis: **const** <tipoConstante> <nombreConstante> = <valor>;

Ejemplo: **const int** a = 123; **const int** b = a + 125;

Excepciones en Mono

Las **excepciones** son para controlar errores los que se produzcan durante la ejecución de las aplicaciones (divisiones por cero, lectura de archivos no disponibles, etc.) Sin embargo, las excepciones proporcionan las siguientes ventajas frente a dicho sistema.

Tabla N° III. 9. Control de Excepciones en Mono

NOMBRE	DESCRIPCIÓN
ArgumentException	Pasado argumento no válido (base de excepciones de argumentos).
ArgumentNullException	Pasado argumento nulo
ArgumentOutOfRangeException	Pasado argumento fuera de rango
ArrayTypeMismatchException	Asignación a tabla de elemento que no es de su tipo
COMException	Excepción de objeto COM
DivideByZeroException	División por cero
IndexOutOfRangeException	Índice de acceso a elemento de tabla fuera del rango válido (menor que cero o mayor que el tamaño de la tabla)
InvalidCastException	Conversión explícita entre tipos no válida
InvalidOperationException	Operación inválida en estado actual del objeto
InteropException	Base de excepciones producidas en comunicación con código inseguro
NullReferenceException	Acceso a miembro de objeto que vale null
OverflowException	Desbordamiento dentro de contexto donde se ha de comprobar los desbordamientos (expresión constante, instrucción checked , operación checked u opción del compilador /checked)
OutOfMemoryException	Falta de memoria para crear un objeto con new
SEHException	Excepción SHE del API Win32
StackOverflowException	Desbordamiento de la pila, generalmente debido a un excesivo número de llamadas recurrentes.
TypeInitializationException	Ha ocurrido alguna excepción al inicializar los campos estáticos o el constructor estático de un tipo. En InnerException se indica cuál es.

3.3. Determinación de los Parámetros de Comparación

Los parámetros que a continuación se definirán para la realización del estudio comparativo de las Tecnologías Ruby on Rails y Mono están basadas en un artículo publicado por Anibal Rojas de las Jornadas de Mono, Ruby, TurboGears, Ruby on Rails, Catalyst en FLISOL¹ 2006 y por el autor de esta tesis.

¹ <http://flisolcaracas.org.ve/blog/index.php?/blog/show/>

Los criterios que se va a considerar en general en este estudio, para analizar las dos tecnologías para el desarrollo de aplicaciones Web son:

Tabla N° III. 10. Determinación de los Criterios de Comparación

PARÁMETROS	CONCEPTO
Acceso a Base de Datos	El acceso a datos es una parte fundamental dentro de las aplicaciones Web porque permite acceder y manipular información almacenada en bases de datos, a través de la utilización de consultas.
Líneas de Código	Se representa para establecer el menor número de líneas código en las inserciones, actualizaciones, eliminaciones y consultas a la Base de Datos.
Portabilidad	Se refiere a la facilidad de ejecutar el sistema sobre cualquier plataforma, además de la posibilidad de operarlo desde diferentes localizaciones físicas.
Interfaz de Usuario	Se refiere si se puede realizar aplicaciones con interfaz visual, tanto para la Web como para aplicaciones de escritorio.

Los cuatro parámetros generales que hemos tomado en cuenta para desarrollar el estudio comparativo están divididos en varios ítems que detallamos a continuación:

Tabla N° III. 11. Variable del Parámetro de Comparación Acceso a Base de Datos

ACCESO A BASE DE DATOS	
VARIABLES	CONCEPTO
Conexión a la base de datos.	Permite la conexión a base de datos para su respectiva manipulación.
Manipulación de base de datos.	Permite tener un control de las tablas, y las consultas que se van a utilizar para la realización de aplicaciones Web dinámicas.
Soporte de base de datos	El soporte para varios tipos de base de datos, entre ellas las de software libre.
Desempeño con base de datos	Admite revisar la estructura de los datos y de las tablas sin necesidad de acudir a una herramienta para administración de base de datos.

Tabla N° III. 12. Variable del Parámetro de Comparación Líneas de Código

LÍNEAS DE CÓDIGO	
VARIABLES	CONCEPTO
Inserción	Líneas Código utilizado para el ingreso de datos.
Actualización	Líneas Código utilizado para actualizar datos de una tabla.
Eliminación	Líneas Código utilizado para eliminar datos de una tabla.
Consulta	Líneas Código utilizado para realizar consultas a la base de datos.

Tabla N° III. 13. Variable del Parámetro de Comparación Portabilidad

PORTABILIDAD	
VARIABLES	CONCEPTO
Plataformas	Sea de hardware o software, sobre el cual un programa puede ejecutarse.
Versiones	Permite corregir las aplicaciones en distintas versiones y que puedan ser interpretadas sin ninguna alteración
Máquinas Virtuales	Permitir ejecutar varios sistemas operativos simultáneamente sobre el mismo hardware.
Compilación	Es el proceso por el cual se traducen programas en código fuente a programas en código objeto.

Tabla N° III. 14. Variable del Parámetro de Comparación Interfaz de Usuario

INTERFAZ DE USUARIO	
VARIABLES	CONCEPTO
Aplicaciones Web	Interfaz visual para el diseño de aplicaciones Web.
Aplicaciones Escritorio	Interfaz visual para el diseño de aplicaciones de escritorio.
Servidor Web	Soporte para la publicación de aplicaciones en la Web.
Editores de texto	Realizar aplicaciones con o sin autocompletación de código.
Mensajes de Error	Mensajes de errores que sean amigables con el usuario

La forma para evaluar las dos tecnologías en base a los cuatro criterios antes mencionados y así obtener resultados cuantitativos y cualitativos que permitan una selección sustentada de una de las tecnologías analizadas, se indican en la siguiente tabla:

Tabla N° III. 15. Escala de Evaluación para las tecnologías

DESCRIPCIÓN	EVALUACIÓN
EA	0
EB	1
EC	2
ED	3
EE	4

Donde:

- EA = Evaluación Mala
- EB = Evaluación Regular
- EC = Evaluación Bueno
- ED = Evaluación Muy Bueno
- EE = Evaluación Excelente

Tabla N° III. 16. Pesos en el parámetro para las variables

DESCRIPCIÓN	PESO
PV₁	= [EA, EB, EC, ED, EE]
PV₂	= [EA, EB, EC, ED, EE]
...	...
PV_i	= [EA, EB, EC, ED, EE]

Donde:

PV₁, PV₂, ..., PV_i = Peso para las variables en las dos tecnologías en el parámetro.

El valor de cada variable en su respectivo parámetro de interpretación para la tecnología incluye la siguiente fórmula:

$$V_1 \geq 0 \quad y \quad V_1 \leq PV_1$$

$$V_2 \geq 0 \quad y \quad V_2 \leq PV_2$$

$$V_i \geq 0 \quad y \quad V_i \leq PV_i$$

Donde:

V_i = Valor de cada variable en el parámetro.

La calificación definitiva de la herramienta en base a cada parámetro de comparación se obtiene sumando los puntajes obtenidos del análisis, utilizando las siguientes fórmulas:

$$\mathbf{Pror} = \Sigma (V_i) \quad \mathbf{Pm} = \Sigma (V_j) \quad \mathbf{Pc} = \Sigma (PV_i)$$

Donde:

Pror = Puntaje acumulado por Ruby on Rails en el parámetro.

Pm = Puntaje acumulado por Mono en el parámetro.

Pc = Puntaje sobre el que se califica el parámetro.

Tabla N° III. 17. Interpretación de Resultados en cada tecnología

DESCRIPCIÓN	RESULTADO
Cror	$= (\text{Pror} / \text{Pc}) * 100\%$
Cm	$= (\text{Pm} / \text{Pc}) * 100\%$

Donde:

Cror = Calificación que obtuvo la Tecnología Ruby on Rails en el parámetro

Cm = Calificación que obtuvo la Tecnología Mono en el parámetro

Tabla N° III. 18. Calificación de Tecnologías

DESCRIPCIÓN	CALIFICACIÓN
Malo	< 60%
Regular	$\geq 60\%$ y $\leq 70\%$
Bueno	$> 70\%$ y $\leq 80\%$
Muy bueno	$> 80\%$ y $\leq 90\%$
Excelente	$> 90\%$

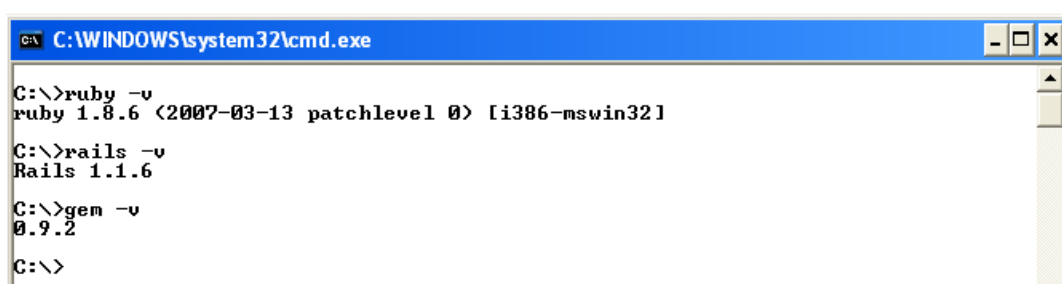
3.4. Desarrollo de módulos de prueba para el Análisis de Comparación

Para el desarrollo de módulos de prueba se realiza con las siguientes especificaciones tanto en Sistema Operativo Windows como en Linux respectivamente:

Windows: XP SP2, MySql 4.1.2, Mono 1.2.5.2, Ruby 1.8.6, Gem 0.9.2, Rails 1.1.6

Linux: Centos Server 4.3, MySql 4.1.12, Mono 1.9, Ruby 1.8.6, Gem 0.9.2, Rails 1.1.6

Windows XP SP2 – Tecnología Ruby on Rails



```
C:\WINDOWS\system32\cmd.exe

C:\>ruby -v
ruby 1.8.6 (2007-03-13 patchlevel 0) [i386-mswin32]

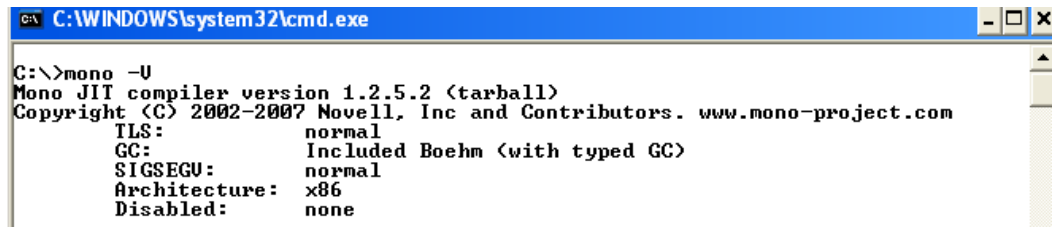
C:\>rails -v
Rails 1.1.6

C:\>gem -v
0.9.2

C:\>
```

Figura N° III. 10. Versiones de Ruby – Rails – Gem

Windows XP SP2 – Tecnología Mono

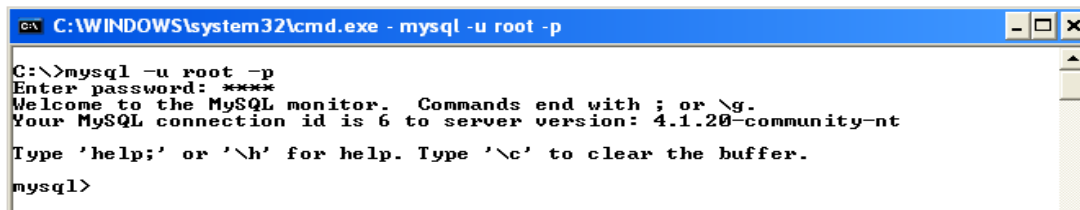


```
C:\WINDOWS\system32\cmd.exe

C:\>mono -U
Mono JIT compiler version 1.2.5.2 (tarball)
Copyright (C) 2002-2007 Novell, Inc and Contributors. www.mono-project.com
  TLS:             normal
  GC:              Included Boehm (with typed GC)
  SIGSEGV:        normal
  Architecture:   x86
  Disabled:       none
```

Figura N° III. 11. Versiones Mono

Windows XP SP2 – Motor de Base de Datos MySql



```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p

C:\>mysql -u root -p
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6 to server version: 4.1.20-community-nt
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql>
```

Figura N° III. 12. Versiones de Motor de Base de Datos MySql

Linux (Centos 4.3.)

Linux – Tecnología Ruby on Rails

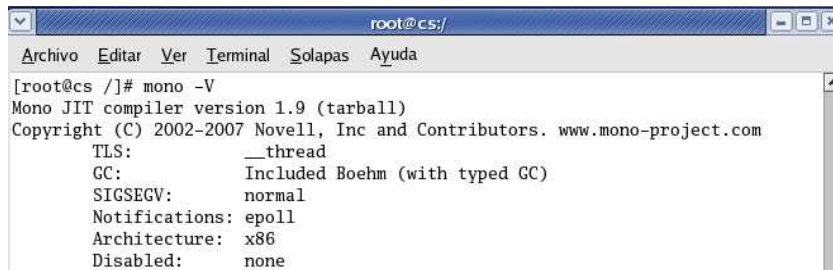


```
root@cs:/

[root@cs /]# ruby -v
ruby 1.8.6 (2007-03-13 patchlevel 0) [i686-linux]
[root@cs /]# rails -v
gemRails 1.1.6
[root@cs /]# gem -v
0.9.2
```

Figura N° III. 13. Versiones de Ruby – Rails – Gem

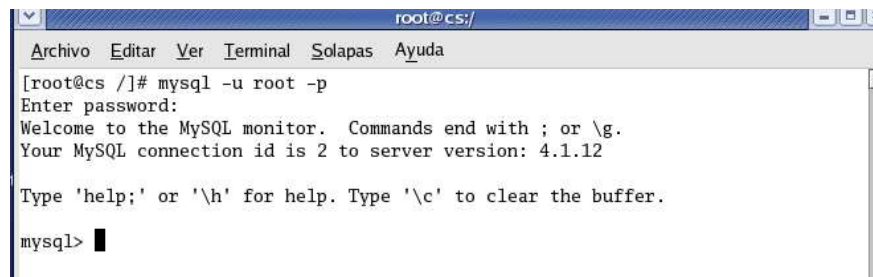
Linux – Tecnología Mono



```
root@cs:/  
Archivo Editar Ver Terminal Solapas Ayuda  
[root@cs /]# mono -V  
Mono JIT compiler version 1.9 (tarball)  
Copyright (C) 2002-2007 Novell, Inc and Contributors. www.mono-project.com  
TLS:          __thread  
GC:           Included Boehm (with typed GC)  
SIGSEGV:     normal  
Notifications: epoll  
Architecture: x86  
Disabled:    none
```

Figura N° III. 14. Versiones Mono

Linux – Motor de Base de Datos MySql



```
root@cs:/  
Archivo Editar Ver Terminal Solapas Ayuda  
[root@cs /]# mysql -u root -p  
Enter password:  
Welcome to the MySQL monitor.  Commands end with ; or \g.  
Your MySQL connection id is 2 to server version: 4.1.12  
  
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.  
mysql> █
```

Figura N° III. 15. Versiones de Motor de Base de Datos MySql

3.4.1. Módulo en Windows

Para el análisis de comparación crearemos una aplicación pequeña tanto en Ruby on Rails y Mono con Base de Datos MySql.

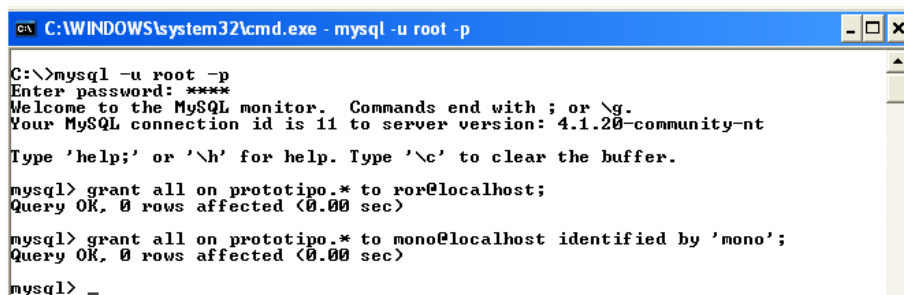
Creamos una base de datos en MySql con el nombre prototipo

```
CREATE DATABASE `prototipo`
```

Creamos una tabla con el nombre datos

```
CREATE TABLE `datos` (  
  `id` int(11) NOT NULL auto_increment,  
  `nombre` varchar(20) default NULL,  
  `apellido` varchar(20) default NULL,  
  `direccion` varchar(50) default NULL,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Creamos usuarios para la base de datos prototipo



```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p
C:\>mysql -u root -p
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11 to server version: 4.1.20-community-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> grant all on prototipo.* to ror@localhost;
Query OK, 0 rows affected (0.00 sec)

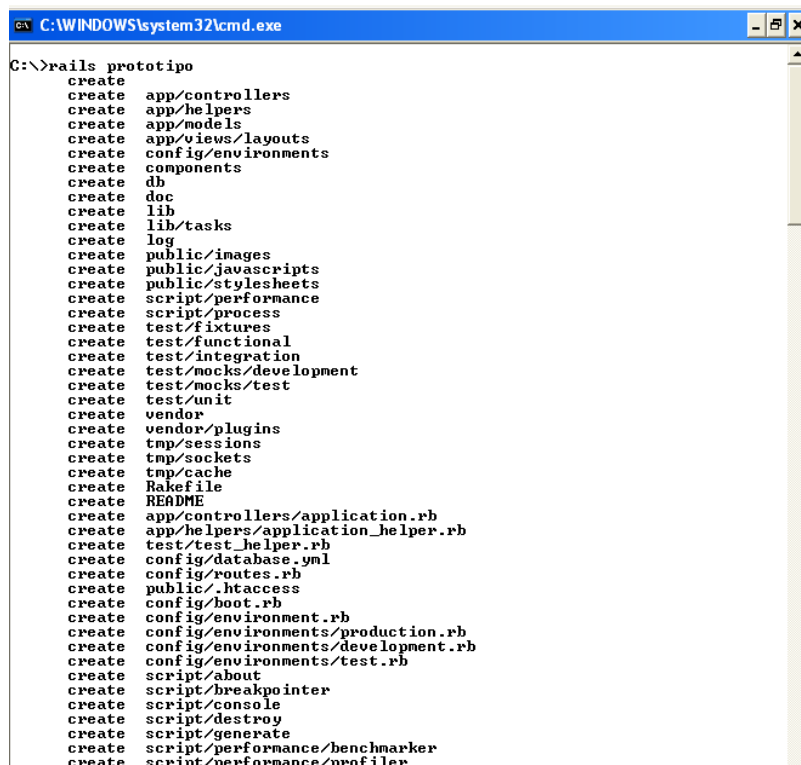
mysql> grant all on prototipo.* to mono@localhost identified by 'mono';
Query OK, 0 rows affected (0.00 sec)

mysql> _
```

Figura N° III. 16. Usuarios para la Base de Datos prototipo

Creamos la Aplicación (TECNOLOGÍA RUBY ON RAILS)

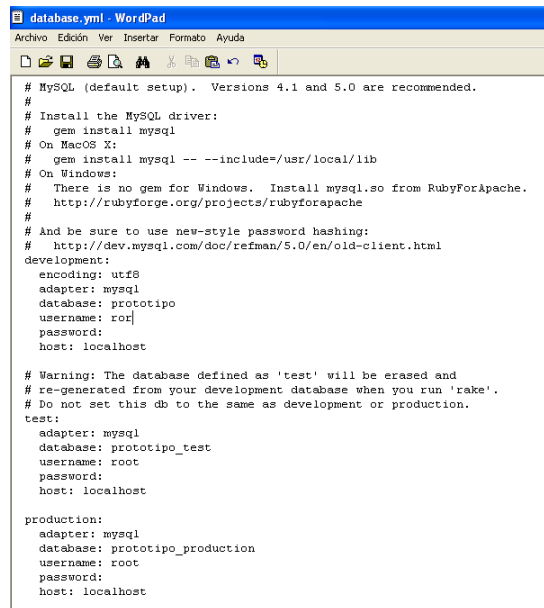
Desde la consola de Windows escribimos el comando Rails nombreaplicacion, con este comando nos crea un directorio con todos los archivos que van hacer utilizados por Ruby on Rails.



```
C:\WINDOWS\system32\cmd.exe
C:\>rails prototipo
create
create  app/controllers
create  app/helpers
create  app/models
create  app/views/layouts
create  config/environments
create  components
create  db
create  doc
create  lib
create  lib/tasks
create  log
create  public/images
create  public/javascripts
create  public/stylesheets
create  script/performance
create  script/process
create  test/fixtures
create  test/functional
create  test/integration
create  test/mocks/development
create  test/mocks/test
create  test/unit
create  vendor
create  vendor/plugins
create  tmp/sessions
create  tmp/sockets
create  tmp/cache
create  Rakefile
create  README
create  app/controllers/application.rb
create  app/helpers/application_helper.rb
create  test/test_helper.rb
create  config/database.yml
create  config/routes.rb
create  public/htaccess
create  config/boot.rb
create  config/environment.rb
create  config/environments/production.rb
create  config/environments/development.rb
create  config/environments/test.rb
create  script/about
create  script/breakpointer
create  script/console
create  script/destroy
create  script/generate
create  script/performance/benchmark
create  script/performance/benchmark
```

Figura N° III. 17. Creación de la aplicación (Ruby on Rails)

Nos ubicamos en (directorio creado anteriormente) C:/prototipo/config/ y editamos el archivo database.yml el cual sirve para la conexión a la base de datos.



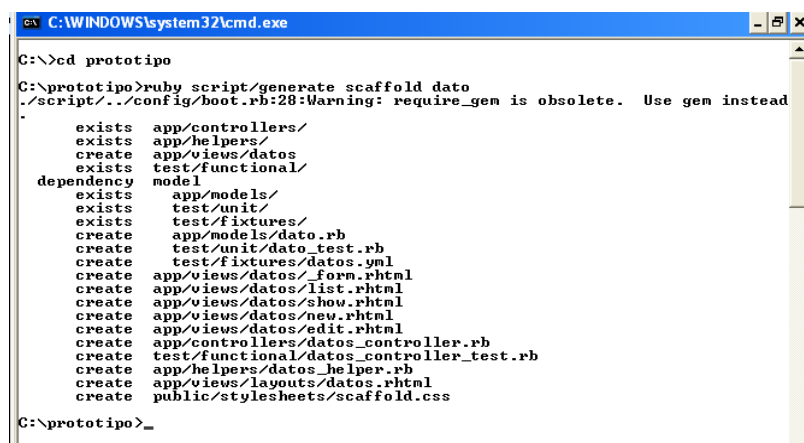
```
# MySQL (default setup). Versions 4.1 and 5.0 are recommended.
#
# Install the MySQL driver:
# gem install mysql
# On MacOS X:
# gem install mysql -- --include=/usr/local/lib
# On Windows:
# There is no gem for Windows. Install mysql.so from RubyForApache.
# http://rubyforge.org/projects/rubyforapache
#
# And be sure to use new-style password hashing:
# http://dev.mysql.com/doc/refman/5.0/en/old-client.html
development:
  encoding: utf8
  adapter: mysql
  database: prototipo
  username: root
  password:
  host: localhost

# Warning: The database defined as 'test' will be erased and
# re-generated from your development database when you run 'rake'.
# Do not set this db to the same as development or production.
test:
  adapter: mysql
  database: prototipo_test
  username: root
  password:
  host: localhost

production:
  adapter: mysql
  database: prototipo_production
  username: root
  password:
  host: localhost
```

Figura N° III. 18. Conexión a la base de datos

Nos vamos a la consola y nos ubicamos en el directorio creado anteriormente (prototipo) y creamos nuestro scaffold el cual nos crea archivos views, controladores, para la manipulación con la base de datos como insertar, eliminar, actualizar, listar, visualizar.



```
C:\WINDOWS\system32\cmd.exe
C:\>cd prototipo
C:\prototipo>ruby script/generate scaffold dato
./script/../config/boot.rb:28:Warning: require_gem is obsolete. Use gem instead
-
  exists    app/controllers/
  exists    app/helpers/
  create    app/views/datos
  exists    test/functional/
  dependency model
  exists    app/models/
  exists    test/unit/
  exists    test/fixtures/
  create    app/models/dato.rb
  create    test/unit/dato_test.rb
  create    test/fixtures/datos.yml
  create    app/views/datos/_form.rhtml
  create    app/views/datos/list.rhtml
  create    app/views/datos/show.rhtml
  create    app/views/datos/new.rhtml
  create    app/views/datos/edit.rhtml
  create    app/controllers/datos_controller.rb
  create    test/functional/datos_controller_test.rb
  create    app/helpers/datos_helper.rb
  create    app/views/layouts/datos.rhtml
  create    public/stylesheets/scaffold.css
C:\prototipo>
```

Figura N° III. 19. Scaffold – Creación de archivos rhtml

En la línea de comandos hacemos correr nuestro servidor de prueba WEBrick.



```
C:\WINDOWS\system32\cmd.exe - ruby script/server
C:\prototipo>ruby script/server
./script/../config/boot.rb:28:Warning: require_gem is obsolete. Use gem instead
=> Booting WEBrick...
=> Rails application started on http://0.0.0.0:3000
=> Ctrl-C to shutdown server; call with --help for options
[2008-04-01 09:57:02] INFO WEBrick 1.3.1
[2008-04-01 09:57:02] INFO ruby 1.8.6 (2007-03-13) [i386-mswin32]
[2008-04-01 09:57:02] INFO WEBrick::HTTPServer#start: pid=4440 port=3000
```

Figura N° III. 20. Servidor de Prueba WEBrick

Abrimos nuestro explorador de Windows y en el browser escribimos <http://localhost:3000/datos> donde debe aparecer los datos que se encuentran en la base de datos (tabla datos).



Figura N° III. 21. Datos de Base de Datos (Tabla datos)

Podemos dar un diseño a nuestra pequeña aplicación, en la carpeta layout en el archivo `datos.rhtml`



Figura N° III. 22. Prototipo de la tecnología RoR

Código de inserción, actualización, eliminación en Ruby on Rails

def list

```
@dato_pages, @datos = paginate :datos, :per_page => 100  
end
```

def show

```
@dato = Dato.find(params[:id])  
end
```

def new

```
@dato = Dato.new  
end
```

def create

```
@dato = Dato.new(params[:dato])  
if @dato.save  
  flash[:notice] = 'Datos Guardados'  
  redirect_to :action => 'list'  
else  
  render :action => 'new'  
end  
end
```

def edit

```
@dato = Dato.find(params[:id])  
end
```

def update

```
@dato = Dato.find(params[:id])  
if @dato.update_attributes(params[:dato])  
  flash[:notice] = 'Datos Actualizados'  
  redirect_to :action => 'show', :id => @dato  
else  
  render :action => 'edit'  
end  
end
```

def destroy

```
Dato.find(params[:id]).destroy  
flash[:notice] = 'Datos Eliminados'  
redirect_to :action => 'list'  
end
```

Creamos Nuestra Aplicación (TECNOLOGÍA MONO)

Esta solución la podemos hacer creando desde alguna herramienta visual o con editor de textos, en nuestro caso optamos por realizarla desde Visual Estudio .net 2003. Cabe mencionar que para la conexión en mono con MySql debemos tener un mysql-connector-net-1.0.10.1 el cual contiene un archivo MySql.Data.dll la misma que sirve para la conexión a MySql. Para conexión a la base de datos se la realiza a la misma base de datos creada anteriormente.

Conexión:

```
using MySql.Data.MySqlClient;
string connectionString = "Server=localhost;Database=prototipo;User
ID=root;Password=root;Pooling=false";
IDbConnection dbcon;
dbcon = new MySqlConnection(connectionString);
dbcon.Open();
dbClose();
```

Insertar:

```
private void btnInsertar_Click(object sender, System.EventArgs e) {
string connectionString = "Server=localhost;Database=prototipo;User
ID=root;Password=root;Pooling=false";
IDbConnection dbcon;
dbcon = new MySqlConnection(connectionString);
dbcon.Open();
IDbCommand dbcmd = dbcon.CreateCommand();
string sql ="insert into datos (nombre,apellido,direccion) values(" + txtNombre.Text +
"," + txtApellido.Text + "," + txtDireccion.Text + ")";
dbcmd.CommandText = sql;
dbcmd.ExecuteNonQuery();
dbcon.Close();
}
```

Actualizar:

```
private void btnActualizar_Click(object sender, System.EventArgs e) {
string connectionString = "Server=localhost;Database=prototipo;User
ID=root;Password=root;Pooling=false";
IDbConnection dbcon;
dbcon = new MySqlConnection(connectionString);
dbcon.Open();
IDbCommand dbcmd = dbcon.CreateCommand();
```

```
string sql ="update datos set nombre=" + txtNombre.Text + "," + "apellido=" +
txtApellido.Text + "," + "direccion=" + txtDireccion.Text + " where id=" + txtId.Text
+ """;
dbcmd.CommandText = sql;
dbcmd.ExecuteNonQuery();
dbcon.Close();
}
```

Eliminar:

```
private void dgListar_DeleteCommand(object source,
System.Web.UI.WebControls.DataGridCommandEventArgs e){
string connectionString = "Server=localhost;Database=prototipo;User
ID=root;Password=root;Pooling=false";
IDbConnection dbcon;
dbcon = new MySqlConnection(connectionString);
dbcon.Open();
IDbCommand dbcmd = dbcon.CreateCommand();
string sql ="delete from datos where id=" + e.Item.Cells[2].Text + """;
dbcmd.CommandText = sql;
dbcmd.ExecuteNonQuery();
dbcon.Close();
}
```

Seleccionar:

```
private void dgListar_EditCommand(object source,
System.Web.UI.WebControls.DataGridCommandEventArgs e){
string connectionString = "Server=localhost;Database=prototipo;User
ID=root;Password=root;Pooling=false";
IDbConnection dbcon;
dbcon = new MySqlConnection(connectionString);
dbcon.Open();
IDbCommand dbcmd = dbcon.CreateCommand();
string sql ="select id, nombre, apellido, direccion from datos where id=" +
e.Item.Cells[2].Text + """;
dbcmd.CommandText = sql;
IDataReader reader = dbcmd.ExecuteReader();
while(reader.Read())
{
int t = (int) reader["id"];
txtId.Text = Convert.ToString(t);
txtNombre.Text = (string) reader["nombre"];
txtApellido.Text = (string) reader["apellido"];
txtDireccion.Text = (string) reader["direccion"];
}
reader.Close();
reader = null;
dbcmd.Dispose();
dbcmd = null; dbcon.Close(); dbcon = null;
}
```


Para probar nuestra aplicación en mono

Copiamos la carpeta creada por Visual Estudio .Net 2003 en cualquier lugar de nuestra PC. Nos ubicamos en el directorio donde fue copiada a carpeta y hacemos correr nuestro servidor de prueba para mono. Cabe mencionar que para ejecutar el servidor de prueba debemos teclear xsp (enter) en el caso de que no exista en nuestra PC instalado ningún otro servidor, caso contrario debemos teclear xsp –port numeropuerto.

```
C:\WINDOWS\system32\cmd.exe - xsp --port 8182
C:\prototipo>cd ..
C:\>cd Mono_Programas
C:\Mono_Programas>cd Prototipo
C:\Mono_Programas\Prototipo>xsp --port 8182
xsp
Listening on port: 8182 (non-secure)
Listening on address: 0.0.0.0
Root directory: C:\Mono_Programas\Prototipo
Hit Return to stop the server.
```

Figura N° III. 23. Servidor de prueba XSP

Abrimos el explorador y en el browser escribimos http://localhost:8182/Prototipo.aspx, donde nos debe cargar nuestra aplicación prototipo.

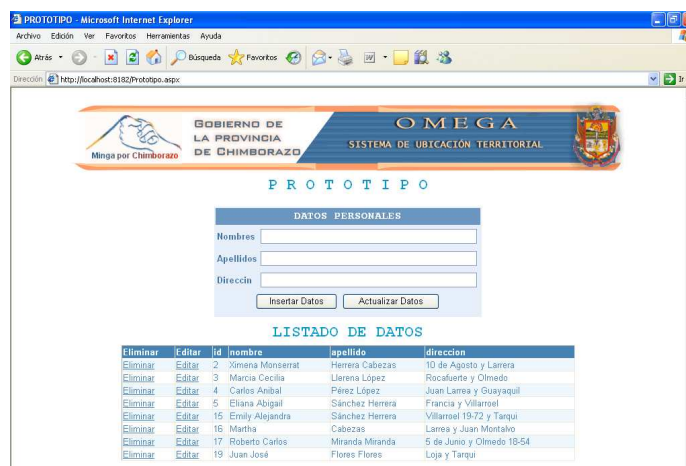


Figura N° III. 24. Prototipo de la tecnología Mono

3.4.2. Módulo en Linux

En este Sistema Operativo debemos tener instalado las dos tecnologías Ruby On Rails y Mono para su respectivo desarrollo de aplicaciones Web. Creamos la misma base de datos y los usuarios como se explicó anteriormente. Posteriormente copiamos los prototipos realizados en Ruby on Rails y Mono del Sistema Operativo Windows y lo pasamos al Sistema Operativo Linux en el lugar que se quiera, cabe recalcar que el Sistema Operativo Linux esta corriendo como una máquina virtual. Una vez hecho todo esto nos dirigimos donde esta cada prototipo y ejecutamos los servidores propios de cada tecnología como se explico anteriormente.

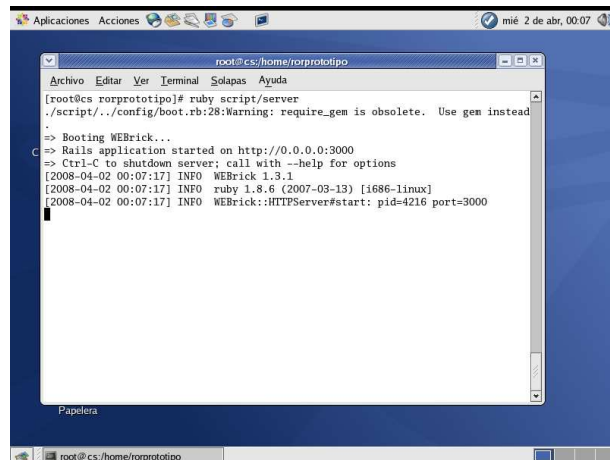


Figura N° III. 25. Servidor WEBrick en Linux para la tecnología Ruby on Rails

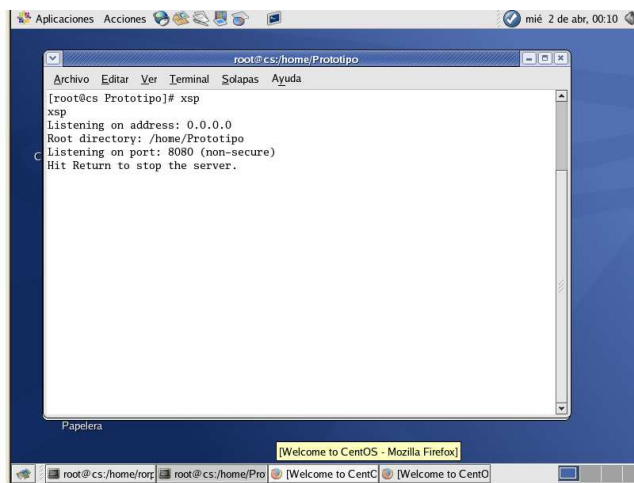


Figura N° III. 26. Servidor XSP en Linux para la Tecnología Mono

Ejecutamos el Mozilla para verificar las aplicaciones están corriendo normalmente y en el browser escribimos <http://localhost:3000/datos> y <http://localhost:8080/Prototipo.aspx>, respectivamente para cada tecnología.



Figura N° III. 27. Ejecución del Prototipo en Linux – Tecnología Ruby on Rails

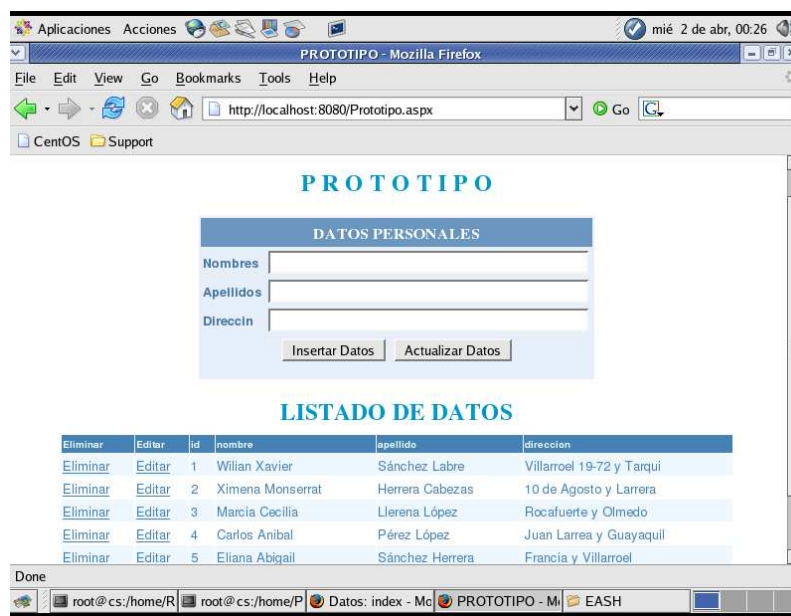


Figura N° III. 28. Ejecución del Prototipo en Linux – Tecnología Mono

3.5. Análisis Comparativo de las Tecnologías Ruby on Rails y Mono

3.5.1. Acceso a Base de Datos

Tabla N° III. 19. Pesos de las variables para las dos tecnologías

DESCRIPCIÓN	PESO	VALOR
PV ₁	EE	4
PV ₂	EE	4
PV ₃	EE	4
PV ₄	EC	2

Tabla N° III. 20. Parámetro Acceso a Base de Datos Tecnología Ruby on Rails

N°	VARIABLES	RUBY ON RAILS	V _i
1	Conexión a la base de datos.	Bastante Simple mediante la configuración del archivo database.yml	4
2	Manipulación de base de datos.	Permite creación de consultas	4
3	Soporte de base de datos.	MySQL – PostgreSQL – Oracle – MS SQL Server - DB2 - SQLite	4
4	Desempeño con base de datos.	Mediante las migraciones.	2

Tabla N° III. 21. Parámetro Acceso a Base de Datos Tecnología Mono

N°	VARIABLES	MONO	V _i
1	Conexión a la base de datos.	Relativamente Simple.	3
2	Manipulación de base de datos.	Permite creación de consultas	4
3	Soporte de base de datos.	MySQL – PostgreSQL – Oracle – MS SQL Server – DB2 – SQL Lite – Firebird - Sybase	4
4	Desempeño con base de datos.	Mediante código	1

Fuente:

<http://www.icsharpcode.net/OpenSource/SD/>

<http://www.ecma-international.org/>

<http://www.maestrosdelweb.com/editorial/desarrollo-web/ruby-on-rails/>

<http://www.riojasoft.com/articles/ruby-on-railss-bases-de-datos-parte-ii>

a. Interpretación de Resultados

$$\mathbf{Pror} = \Sigma (V_i)$$

$$\text{Pror} = \Sigma(4 + 4 + 4 + 2)$$

$$\text{Pror} = 14$$

$$\mathbf{Pm} = \Sigma (V_j)$$

$$\text{Pm} = \Sigma(3 + 4 + 4 + 1)$$

$$\text{Pm} = 12$$

$$\mathbf{Pc} = \Sigma (PV_i)$$

$$\text{Pc} = \Sigma (4 + 4 + 4 + 2)$$

$$\text{Pc} = 14$$

$$\mathbf{Cror} = (\text{Pror} / \text{Pc}) * 100\%$$

$$\text{Cror} = (14 / 14) * 100\%$$

$$\text{Cror} = 100\%$$

$$\mathbf{Cm} = (\text{Pm} / \text{Pc}) * 100\%$$

$$\text{Cm} = (12 / 14) * 100\%$$

$$\text{Cm} = 85.71\%$$

Tabla N° III. 22. Resultados del Parámetro Acceso a Base de Datos

VARIABLES	TECNOLOGÍAS					
	RUBY ON RAILS			MONO		
	PV _i	V _i	Cror	PV _i	V _i	Cm
V ₁	4	4	100%	4	3	85.71%
V ₂	4	4		4	4	
V ₃	4	4		4	4	
V ₄	2	2		2	1	
TOTAL	14			14	12	

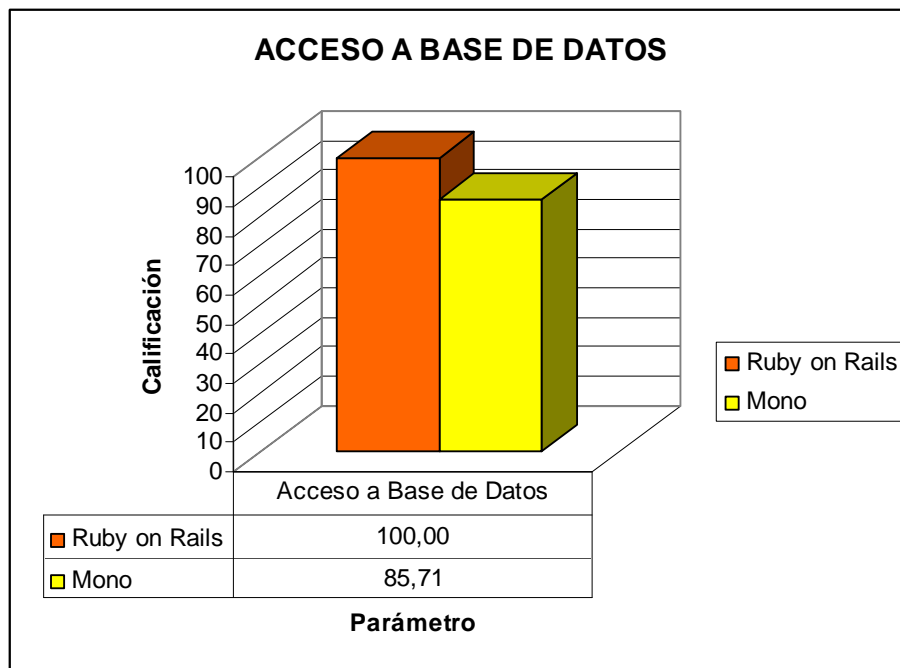


Figura N° III. 29. Parámetro Acceso a Datos

b. Descripción de resultados

- Ruby on Rails es una tecnología casi nueva que en los últimos años a tenido gran acogida en todo el mundo por tal razón tiene un desarrollo de aplicaciones Web muy rápido con conexiones a base de datos sin ningún problema y en pocos pasos en cambio la tecnología mono también tiene su respectiva acogida en todo el mundo por lo cual su conexión hacia la base de datos es casi simple ya que siempre tiene que conectarse y desconectarse cuando hace consultas a la base de datos.
- La manipulación de la base de datos en ambas tecnologías son excelentes debido a que dichas tecnologías tiene las sentencias Sql las mismas que son conocidas por los programadores por la cual su manipulación la hace simple para su respectivo desarrollo, cabe recalcar que en ror las sentencias de Sql se las puede

realizar con las propias sentencias de ror; con la diferencia de realizar la misma consulta en menos líneas de código.

- Las dos tecnologías tiene soporte a casi las mismas base de datos con la diferencia que mono tiene más soporte a base de datos.
- El desempeño con base de datos en ambas tecnologías es excelente debido a en Ruby on Rails se puede manipular las tablas de la base de datos mediante migraciones y en mono mediante el mismo Visual Studio .Net.

3.5.2. Líneas de Código

Tabla N° III. 23. Pesos de las variables para las dos tecnologías

DESCRIPCIÓN	PESO	VALOR
PV ₁	EE	4
PV ₂	EE	4
PV ₃	EE	4
PV ₄	EE	4

Tabla N° III. 24. Parámetro Líneas de Código Tecnología Ruby on Rails

N°	VARIABLES	RUBY ON RAILS	V _i
1	Inserción	Bastante simple	4
2	Actualización	Bastante simple	4
3	Eliminación	Bastante simple	4
4	Consultas	Bastante simple	4

Tabla N° III. 25. Parámetro Líneas de Código Tecnología Mono

N°	VARIABLES	MONO	V _i
1	Inserción	Relativamente Simple.	3
2	Actualización	Relativamente Simple.	3
3	Eliminación	Relativamente Simple.	3
4	Consultas	Relativamente Simple.	3

Fuente:

<http://www.go-mono.com>, <http://dotgnu.org/>
<http://www.humansharp.com/index.php?var=code>
<http://ruben.peruonrails.com/implementando-una-aplicacionror>
 Libro: Agile Web Development With Ruby On Rails.pdf

a. Interpretación de Resultados

$$\mathbf{Pror} = \Sigma (V_i)$$

$$\text{Pror} = \Sigma(4 + 4 + 4 + 4)$$

$$\text{Pror} = 16$$

$$\mathbf{Pm} = \Sigma (V_j)$$

$$\text{Pm} = \Sigma(3 + 3 + 3 + 3)$$

$$\text{Pm} = 12$$

$$\mathbf{Pc} = \Sigma (PV_i)$$

$$\text{Pc} = \Sigma (4 + 4 + 4 + 4)$$

$$\text{Pc} = 16$$

$$\mathbf{Cror} = (\text{Pror} / \text{Pc}) * 100\%$$

$$\text{Cror} = (16 / 16) * 100\%$$

$$\text{Cror} = 100\%$$

$$\mathbf{Cm} = (\text{Pm} / \text{Pc}) * 100\%$$

$$\text{Cm} = (12 / 16) * 100\%$$

$$\text{Cm} = 75\%$$

Tabla N° III. 26. Resultado Parámetro Líneas de Código

VARIABLES	TECNOLOGÍAS					
	RUBY ON RAILS			MONO		
	PV _i	V _i	Cror	PV _i	V _i	Cm
V ₁	4	4	100%	4	3	75%
V ₂	4	4		4	3	
V ₃	4	4		4	3	
V ₄	4	4		4	3	
TOTAL	16	16		16	12	

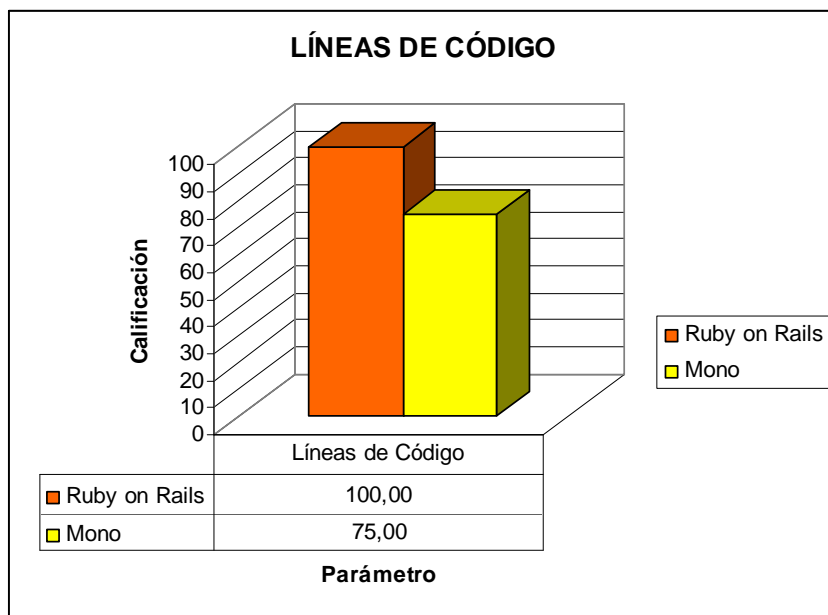


Figura N° III. 30. Parámetro Líneas de Código

b. Descripción de resultados

- En Ruby on Rails la inserción, actualización, eliminación es muy sencilla, ya que mediante el scaffold (andamiaje) este nos genera automáticamente CRUD, en cambio en la tecnología mono debemos realizar su respectiva conexión inserción en cada una de los CRUD.

3.5.3. Portabilidad

Tabla N° III. 27. Pesos de las variables para las dos tecnologías

DESCRIPCIÓN	PESO	VALOR
PV ₁	EE	4
PV ₂	ED	3
PV ₃	EC	2
PV ₄	EB	1

Tabla N° III. 28. Parámetro Potabilidad Tecnología Ruby on Rails

N°	VARIABLES	RUBY ON RAILS	V _I
1	Plataformas	Linux (RedHat, Debian, Fedora, Ubuntu, SuSE), Windows (XP/2000), MacOS X.	4
2	Versiones	Es poco probable que las aplicaciones corran en distintas versiones.	1
3	Máquinas Virtuales	Muy Simple, trabaja como una máquina física.	2
4	Compilación	No necesita compilación para su respectiva ejecución, por lo cual las aplicaciones se ejecutan normalmente.	1

Tabla N° III. 29. Parámetro Portabilidad Tecnología Mono

N°	VARIABLES	MONO	V _i
1	Plataformas	GNU/Linux, Windows, Solaris, Mac OS X, BSD	4
2	Versiones	Casi probable para que las aplicaciones corran en distintas versiones.	3
3	Máquinas Virtuales	Muy Simple, trabaja como una máquina física.	2
4	Compilación	Necesariamente debe compilarse para su respectiva ejecución.	0.5

Fuente:

<http://www.go-mono.com>, <http://dotgnu.org/>
<http://www.monohispano.es/index.php/desarrolladoresMono:Prefacio>
http://sobrerails.com/pages/en_marcha_con_rails_2
 Libro: Agile Web Development With Ruby On Rails.pdf

a. Interpretación de Resultados

$$\text{Pror} = \Sigma (V_i)$$

$$\text{Pror} = \Sigma(4 + 1 + 2 + 1)$$

$$\text{Pror} = 8$$

$$\text{Pm} = \Sigma (V_j)$$

$$\text{Pm} = \Sigma(4 + 3 + 2 + 0.5)$$

$$\text{Pm} = 9.5$$

$$\text{Pc} = \Sigma (PV_i)$$

$$\text{Pc} = \Sigma (4 + 3 + 2 + 1)$$

$$\text{Pc} = 10$$

$$\text{Cror} = (\text{Pror} / \text{Pc}) * 100\%$$

$$\text{Cror} = (8 / 10) * 100\%$$

$$\text{Cror} = 80\%$$

$$\text{Cm} = (\text{Pm} / \text{Pc}) * 100\%$$

$$\text{Cm} = (9.5 / 10) * 100\%$$

$$\text{Cm} = 95\%$$

Tabla N° III. 30. Resultado Parámetro Portabilidad

VARIABLES	TECNOLOGÍAS					
	RUBY ON RAILS			MONO		
	PV _i	V _i	Cror	PV _i	V _i	Cm
V ₁	4	4	80%	4	4	95%
V ₂	3	1		3	3	
V ₃	2	2		2	2	
V ₄	1	1		1	0.5	
TOTAL	10	8		10	9.5	

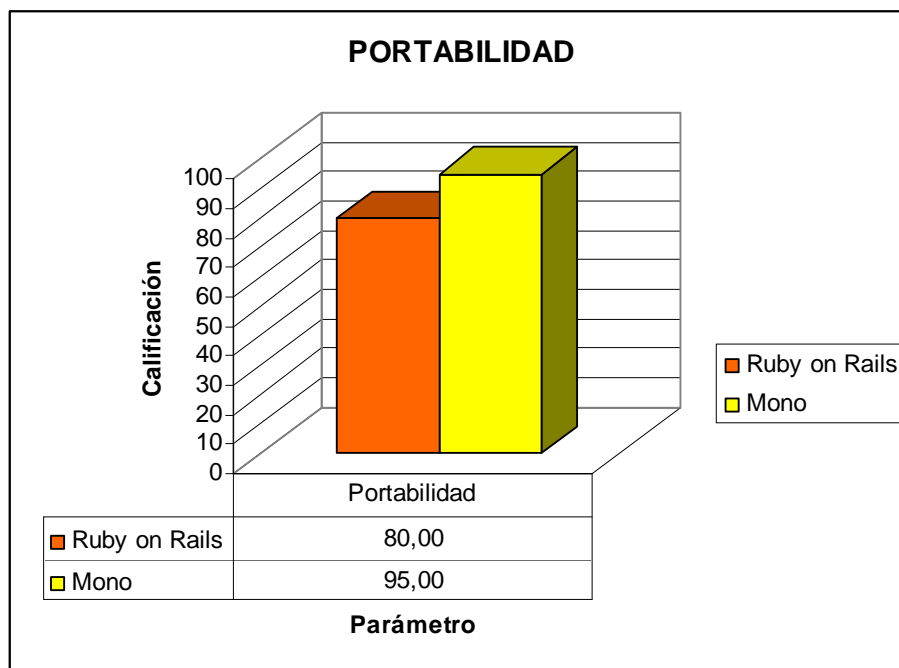


Figura N° III. 31. Parámetro Portabilidad

b. Descripción de resultados

- Ruby on Rails es casi portable para todos los SO, donde mono tiene mucho mas plataformas en la cual se puede ejecutar por ende se concluye que en esta variable mono es superior.
- Mono es probable que se ejecute en sus distintas versiones sin complicaciones; en cambio RoR no es compatible en sus distintas versiones, por ende la tecnología mono en esta variable es muy superior.
- Ambas tecnologías pueden ejecutarse sin ningún problema en máquinas virtuales, por tal situación ambas tecnologías son iguales para esta variable.
- La tecnología RoR no necesita configuraciones para su respectiva ejecución, en cambio mono siempre necesita compilarse para su ejecución, en tal virtud RoR es superior en esta variable de comparación.

3.5.4. Interfaz de Usuario

Tabla N° III. 31. Pesos de las variables para el parámetro Interfaz

DESCRIPCIÓN	PESO	VALOR
PV ₁	EE	4
PV ₂	EC	1
PV ₃	EE	4
PV ₄	EE	4
PV ₅	EE	4

Tabla N° III. 32. Parámetro Interfaz Tecnología Ruby on Rails

N°	VARIABLES	RUBY ON RAILS	V _i
1	Aplicaciones Web	Macromedia, 3ra Rail	3
2	Aplicaciones Escritorio	Ninguna.	0
3	Servidor Web	Apache, Mongrel, WEBrick, FastCGI, Lighttpd.	4
4	Editores de Texto	Varios (RadRails, TexMate, JEdit, RubyWeaver, RidMe, InType)	4
5	Mensajes de error	Para la observación visual hacia el usuario donde existe el error	4

Tabla N° III. 33. Parámetro Interfaz Tecnología Mono

N°	VARIABLES	MONO	V _i
1	Aplicaciones Web	MonoDevelop, SharpDevelop, VS. Net.	4
2	Aplicaciones Escritorio	GTK#, GSharp, QtK, Visual Studio.Net, Cocoa#, wxNet, 3D(Tao), Glade	1
3	Servidor Web	XSP, Apache, Lighttpd, Mongrel.	4
4	Editores de Texto	Varios (RadRails, TexMate, JEdit, RidMew, InType, DreamWeaver)	4
5	Mensajes de error	Observación visual del usuario donde existe el error	4

Fuente:

<http://monoteca.glisc.org>, <http://go-mono.org/monologue/>
http://sobrerailes.com/pages/en_marcha_con_rails_2
 Libro: Agile Web Development With Ruby On Rails.pdf

a. Interpretación de Resultados

$$\begin{aligned}
 \text{Pror} &= \Sigma (V_i) \\
 \text{Pror} &= \Sigma(3 + 0 + 4 + 4 + 4) \\
 \text{Pror} &= 15 \\
 \\
 \text{Pm} &= \Sigma (V_j) \\
 \text{Pm} &= \Sigma(4 + 1 + 4 + 4 + 4) \\
 \text{Pm} &= 17 \\
 \\
 \text{Pc} &= \Sigma (PV_i) \\
 \text{Pc} &= \Sigma(4 + 1 + 4 + 4 + 4) \\
 \text{Pc} &= 17 \\
 \\
 \text{Cror} &= (\text{Pror} / \text{Pc}) * 100\% \\
 \text{Cror} &= (15 / 17) * 100\% \\
 \text{Cror} &= 88.24\% \\
 \\
 \text{Cm} &= (\text{Pm} / \text{Pc}) * 100\% \\
 \text{Cm} &= (17 / 17) * 100\% \\
 \text{Cm} &= 100\%
 \end{aligned}$$

Tabla N° III. 34. Resultado Parámetro Interfaz

VARIABLES	TECNOLOGÍAS					
	RUBY ON RAILS			MONO		
	PV _i	V _i	Cror	PV _i	V _i	Cm
V ₁	4	3	88.24%	4	4	100%
V ₂	1	0		1	1	
V ₃	4	4		4	4	
V ₄	4	4		4	4	
V ₅	4	4		4	4	
TOTAL	14	15		17	17	

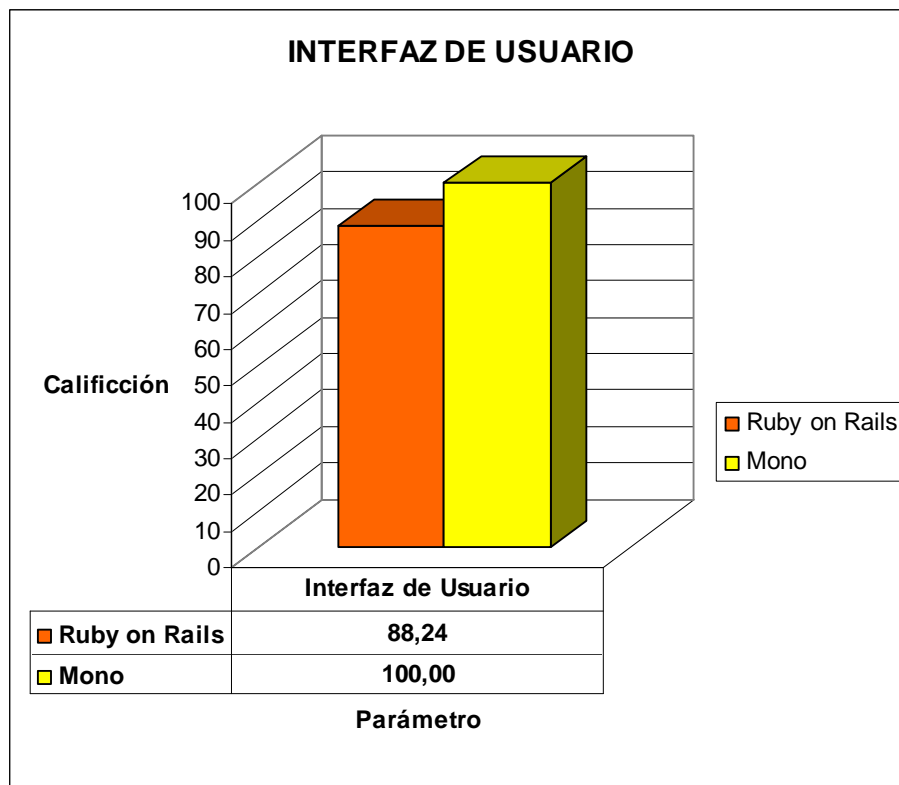


Figura N° III. 32. Parámetro Interfaz

b. Descripción de resultados

- La tecnología RoR tiene muy pocas interfaz de usuario para su desarrollo y diseño de aplicaciones, por otra parte mono tiene un sin numero de interfaz de usuario para su desarrollo.

- La tecnología RoR no tiene aplicaciones de escritorio, en cambio mono tiene aplicaciones de escritorio tanto para la Web como para aplicaciones Windows de escritorio.
- Ambas tecnologías tiene servidor para su respectiva publicación de las aplicaciones en la Web, además ambas tecnologías tiene su servidor de prueba propio como es WEBrick y XSP para RoR y mono respectivamente.
- De igual las dos tecnologías tienen varios editores de texto para la creación de aplicaciones Web.
- Para los mensajes de errores que puedan visualizar los usuarios se dan en las dos tecnologías.

3.5.5. Puntajes Alcanzados

El puntaje final y el porcentaje que ha obtenido cada tecnología se obtiene de la siguiente manera:

Puntaje Total del Análisis

$$PT = \Sigma (Pc)$$

Puntaje Total de RoR

$$PT_{ror} = ((\Sigma(P_{ror})) / PT) * 100\%$$

Puntaje Total de Mono

$$PT_m = ((\Sigma(P_m)) / PT) * 100\%$$

Tabla N° III. 35. Tabla General de Resultados

PARÁMETRO	VARIABLES	TECNOLOGÍAS		PESO
		RoR	MONO	Pc
Acceso a Base de Datos	Conexión a la base de datos.	4	3	4
	Manipulación de base de datos.	4	4	4
	Soporte de base de datos.	4	4	4
	Desempeño con base de datos.	2	1	2
Líneas de Código	Inserción	4	3	4
	Actualización	4	3	4
	Eliminación	4	3	4
	Consulta	4	3	4
Portabilidad	Plataformas	4	4	4
	Versiones	1	3	3
	Máquinas Virtuales	2	2	2
	Compilación	1	0.5	1
Interfaz de Usuario	Aplicaciones Web	3	4	4
	Aplicaciones Escritorio	0	1	1
	Servidor Web	4	4	4
	Editores de texto	4	4	4
	Mensajes de Error	4	4	4
TOTAL		53.00	50.50	57

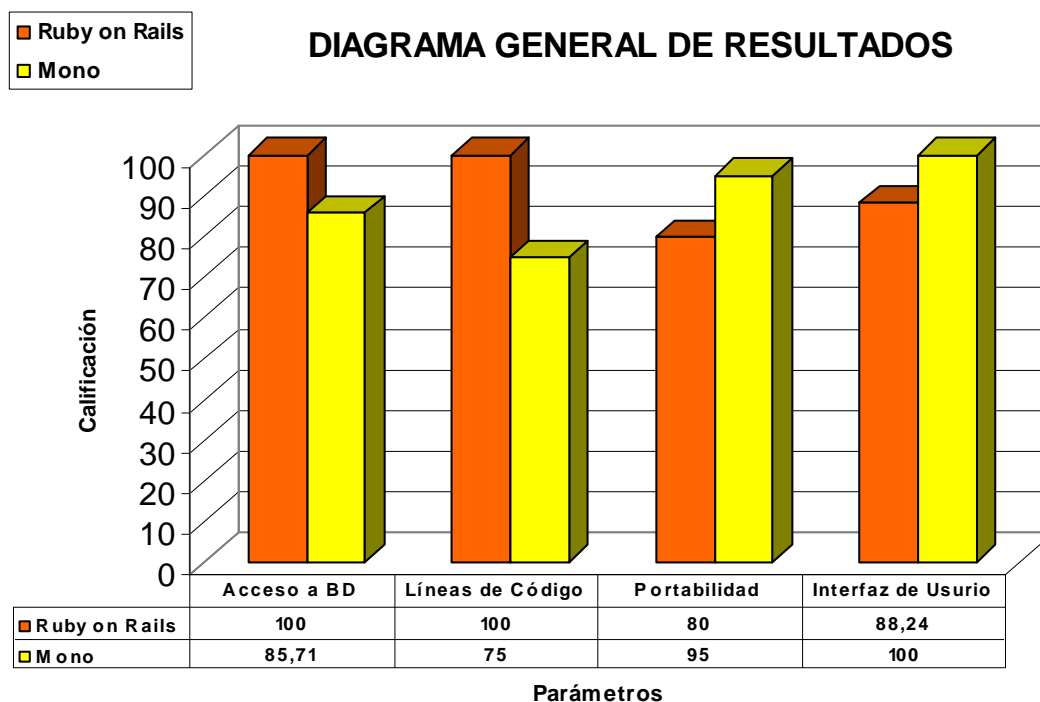


Figura N° III. 33. Diagrama General de Resultados

Puntaje Total del Análisis

$$PT = \Sigma (Pc)$$

$$PT = 57$$

Puntaje Total de RoR

$$PTror = ((\Sigma(Pror)) / PT) * 100\%$$

$$PTror = (53 / 57) * 100\%$$

$$PTror = 92.98\%$$

Puntaje Total de Mono

$$PTm = ((\Sigma(Pm)) / PT) * 100\%$$

$$PTm = (50.5 / 57) * 100\%$$

$$PTm = 88.60\%$$

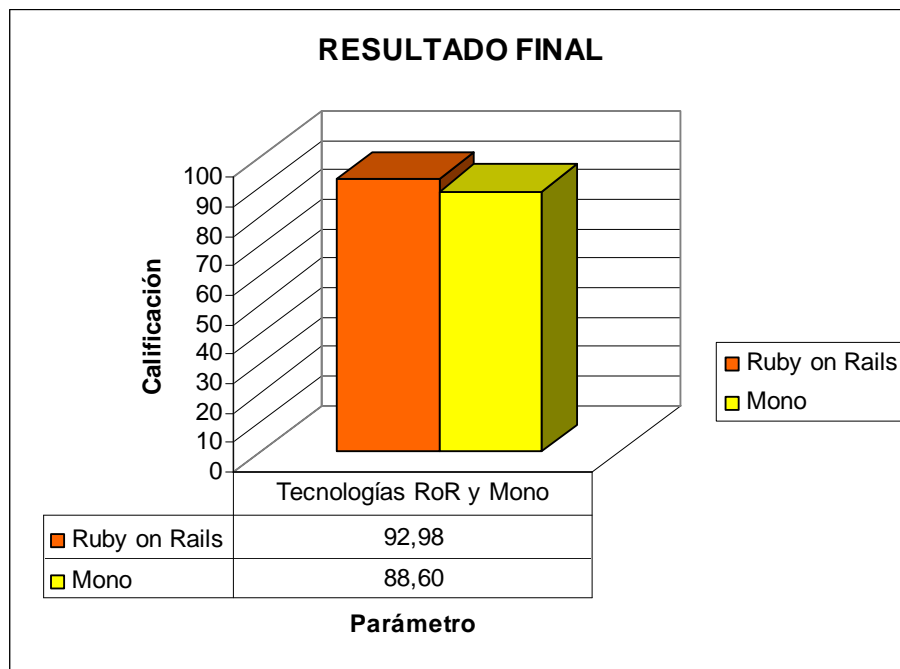


Figura N° III. 34. Resultado Final

3.6. Resultado del Análisis Comparativo

- La facilidad para el desarrollo de aplicaciones Web mediante las diferentes tecnologías que existen hoy en día, entre ellas Ruby on Rails y Mono es una ventaja fundamental, por ende se pueda obtener el mayor beneficio de cualquier tecnología para el desarrollo de software en una empresa pública y/o privada.
- Con el avance tecnológico por lo tanto es una obligación realizar un estudio comparativo entre tecnologías para elegir la más óptima en el momento de realizar un sistema y así cumplir las necesidades y requerimiento actuales de una empresa; donde dicho sistema desarrollado en una tecnología determinada con el transcurrir del tiempo no queden obsoleta.
- En base a todos los parámetros y variables generales se ha demostrado que la mejor tecnología para el desarrollo de aplicaciones Web en el Honorable Consejo Provincial de Chimborazo para el sistema de Seguimiento y Monitoreo de Proyectos y por todo lo expuesto anteriormente de acuerdo a los puntajes alcanzados para cada uno de los parámetros de evaluación con sus respectivas variables se puede demostrar y concluir que la tecnología Ruby on Rails es la que brinda las mejores prestaciones para el desarrollo de aplicaciones Web alcanzado un porcentaje de 92.98% contra la tecnología Mono de 88.60%, por tal motivo la tecnología Ruby on Rails y Mono se las califica como Excelente y Muy Bueno respectivamente.
- También se concluye que la diferencia entre las dos tecnologías comparadas es de un 4.38%, por ende hay un empate técnico, por tal motivo para el desarrollo del Sistema de Seguimiento y Evaluación de Proyectos se opta por realizarla en la tecnología Ruby on Rails.

Tabla N° III. 36. Calificación final de las Tecnologías Ruby on Rails y Mono

DESCRIPCIÓN	CALIFICACIÓN
Ruby on Rails	Excelente
Mono	Muy Bueno

3.6.1. Ventajas y Desventajas de Ruby on Rails

Mediante el análisis comparativo realizado anteriormente y con sus respectivas características podemos dar las siguientes ventajas y desventajas.

- **Ventajas**

- **DRY (Don't Repeat Yourself):** “no te repitas a ti mismo”, con esto podemos tener un formulario, y llamarlo las veces que queramos y desde donde queramos, simplemente con una línea código, o tal vez tener una tabla en nuestra base de datos, y manipular a los registros como un objeto y a sus campos como un atributo, sin necesidad de que declaremos nada.
- **Convención sobre configuración:**
 - Con esa declaración de una clase, mapeamos a una tabla en nuestra base de datos, dicho de otra manera Rails buscara una tabla llamada 'autos', en nuestra base de datos, y ¿porque en plural?, esto es así porque Rails cree conveniente que debe llamarse así (principio de pluralización), aunque este comportamiento se puede desactivar de una manera muy sencilla, y ¿si no la encuentra?, pues nos dará un error.

Ejemplo: class Auto < ActiveRecord::Base end

¿Y si la tabla con la que quiero trabajar no tiene ese nombre exacto?, no hay problema, con una línea más lo podemos

```
solucionar: class Auto < ActiveRecord::Base
              set_table_name 'carros'
            end
```

Con esto el framework comprenderá que en vez de usar 'autos', debería usar 'carros'.

- **Uso de patrones de diseño:** Modelo Vista Controlador (MVC)
Generación de código (helpers): permiten la generación de código xhtml, xml y javascript a partir de código Ruby.
- **Menos código, menos errores.**
- **Test integrados (unitarios y funcionales).**
- **Soporta Scaffolding**, esto es que sin escribir una línea de código podemos generar un CRUD (Crear – Leer/Buscar – Actualizar/Modificar – Eliminar) a partir de una clase del Modelo.
- **Plataformas**
 - Linux (RedHat, Debian, Fedora, Ubuntu, SuSE, etc.)
 - Windows (XP/2000)
 - MacOS X
- **Web Servers**
 - Apache
 - Mongrel, WEBrick
 - Fast CGI, lighttpd
 - IIS, etc.

- **BD**
 - MySQL
 - Oracle
 - PostgreSQL
 - MS SQL Server
 - DB2
 - SQLite
- **Servidor Web propio WEBrick**
 - Tiene propio servidor Web WEBrick o se acopla con Apache (mod_ruby).
- **Desventajas**
 - Para el desarrollo de aplicaciones muy grandes.
 - Algunas incompatibilidades entre versiones.

3.6.2. Ventajas y Desventajas de la Tecnología Mono

Mediante el análisis comparativo realizado anteriormente y con sus respectivas características podemos dar las siguientes ventajas y desventajas.

- **Ventajas**
 - **Independencia de lenguaje:** puedes usar clases escritas en cualquier lenguaje soportado por Mono (por ahora, C#, Mono Basic, Java, Nemerle, MonoLOGO, Boo, IronPython)
 - **Independencia de plataforma:** las aplicaciones son muy portables, y la mayoría compatibles en binario entre plataformas

- **Gran soporte para bases de datos:** MS SQL, MySQL, Postgres, OLE DB, etc.
 - **Velocidad:** el lenguaje intermedio se compila en cada plataforma con unos compiladores muy rápidos (JIT) lo que lo hace mucho más rápido que lenguajes interpretados como PHP ó Python y más rápido en la compilación (JIT) que Java. Únicamente un poco más lento que C.
 - **Gestión automática de memoria:** es una fuente inagotable de errores y se pierde una gran cantidad de tiempo programando esto.
 - **Seguridad**
 - **Aplicaciones Web:** Cualquier lenguaje soportado por Mono se puede usar para Aplicaciones Web. No hay necesidad de lenguajes especiales como PHP.
 - **Servicios Web:** soporte para SOAP
 - **Soporte para XML:** Mono tiene muchas clases para trabajar con xml
 - Extensa librería de clases: Criptografía, HTTP, Bases de datos, GUI, etc.
 - **Aplicaciones GUI multiplataformas:** se pueden escribir aplicaciones con interfaz gráfica que se ejecutan invariablemente en multitud de plataformas. Por ejemplo, Gtk# es muy potente en cualquier plataforma.
 - **Servidor Web propio:** XSP
- **Desventajas**
 - Siempre necesita compilarse para su respectiva ejecución.
 - El Rendimiento en su ejecución.

CAPÍTULO IV

DESARROLLO DEL SISTEMA DE SEGUIMIENTO Y MONITOREO DE PROYECTOS EN EL CONSEJO PROVINCIAL DE CHIMBORAZO

4.1. Introducción

Con el desarrollo del Sistema de Seguimiento y Evaluación de Proyectos en el Departamento de Planificación del Honorable Consejo Provincial de Chimborazo se permitirá automatizar el proceso de seguimiento de proyectos los mismos que lo realizan de forma manual con el fin de mejorar la calidad de este proceso en beneficio de los técnicos del Departamento de Planificación.

Para la realización de la aplicación, se va a seguir la metodología de desarrollo orientada a objetos, que define una serie de actividades; y que además adopta un enfoque eminentemente práctico, aportando soluciones a las principales dudas y problemas con los que se enfrenta el desarrollador. La notación que se usa para la metodología, es proporcionada por UML estándar para la notación orientado a objetos.

4.2. Secuencia de Etapas o Fases

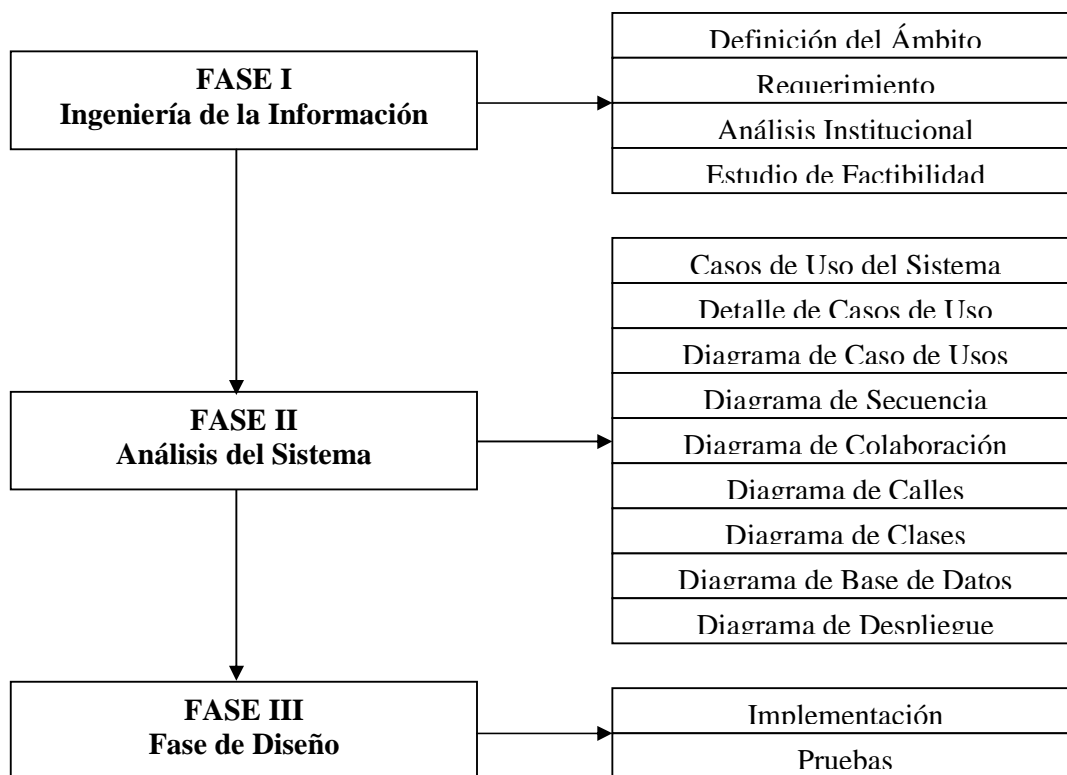


Figura N° IV. 1. Secuencia de Etapas o Fases

4.3. FASE I – Ingeniería de la Información

4.3.1. Definición del Ámbito

Actualmente el Honorable Consejo Provincial de Chimborazo (HCPCH) en el Departamento de Planificación lleva el control de los proyectos que se realizan en la Provincia de Chimborazo mediante archivos de Microsoft Excel, Word, de igual manera poseen un Programa llamado Omega, SEPOA, loca cuales realizan tareas específicas en un determinado departamento, por tal razón no tienen un proyecto que pueda automatizar el control de costos, actividades, de un proyecto, cabe recalcar que en el uso de actividades de un proyecto no realizan la actividad con una fecha de inicio

y una fecha de finalización lo cual el control de un proyecto lo realizan mediante graficas de barras.

4.3.2. Requerimientos

Para solucionar los inconvenientes citados en el literal anterior se puede desarrollar un sistema, el mismo que permita: Insertar, actualizar, consultar, la información sobre el estado, costo, actividades (sin uso de fecha inicio y fecha de finalización) de un proyecto respectivamente, de igual manera los ejes y programas en que esta creado el proyecto. Generar reportes de los requerimientos que se realizan a los proyectos entre ellos gráficas de barras, POA, Estado de los proyectos, Evaluación.

4.3.3. Análisis Institucional

Nombre del Proyecto: Seguimiento y Evaluación de Proyecto en Chimborazo

Empresa: Honorable Consejo Provincial de Chimborazo

Ubicación Geográfica: **Provincia:** Chimborazo **Cantón:** Riobamba

Dirección: 1ra Constituyente y Carabobo

Teléfono: 2960290 – 2960988 **Fax:** 2947397

Antecedentes

El Consejo Provincial de Chimborazo, se crea el primero de enero de 1946, bajo un análisis somero y efectivo de los gobernantes de aquella época, como medio de solución a las necesidades y requerimientos de los pueblos más alejados por el Poder Central.

Su historia ha sido dilatada desde el inicio de su gestión, por carencia de un cuerpo legal que norme sus responsabilidades, derechos y obligaciones. Su funcionamiento físico-

administrativo se desarrolló en lo que hoy, es la sala de recepciones de la Gobernación de Chimborazo, contando con un equipo de servidores de un Secretario, un Auxiliar de Secretaría, un Jefe de Obras, un Tesorero, un Conserje y dos Choferes.

Misión

Liderar un proceso de participación y responsabilidad social con todos los sectores. Cambiando el sistema tradicional del clientelismo y la dependencia hacia una institución dinámica, democrática, innovadora y cambiante. Crear las condiciones necesarias para mejorar la situación de salud de la población vulnerable.

Fortalecer las organizaciones populares para que se conviertan en líderes de la propuesta Minga por Chimborazo. Gestión Institución para lograr recursos orientados a los proyectos de asistencia social. Impulsar la formación de talentos humanos que se incorporen a las propuestas de servicio social, desde el contexto de la educación en todos los niveles. Formar equipos de trabajo que actúen de manera coordinada para cumplir los objetivos definidos por el Patronato.

Disminuir los índices de Desnutrición Crónica y Analfabetismo de mujeres en el área rural y urbano marginal de la Provincia de Chimborazo, con el desarrollo de Proyectos de Salud y Educación. Ejecutar y Promover Brigadas Médicas en el sector Rural y urbano marginal. Desarrollar planes de Mejoramiento para el Centro de Cuidado y Desarrollo Infantil CCDI y trabajar en forma coordinada con el Programa de Rescate Infantil (ORÍ).

Visión

El Patronato del Gobierno de la Provincia de Chimborazo, para después de 4 años habrá logrado desarrollar en la sociedad provincial el sentido de corresponsabilidad social, fomentando los principios de participación, equidad, justicia social, interculturalidad, identidad, para lograr que la minga por Chimborazo promueva el desarrollo que, reconociendo las diferencias culturales, construya la sociedad unitaria y solidaria basada en las potencialidades de sus talentos locales.

Estrategias

- Fomentar procesos que faciliten la atención en educación y salud de toda la provincia, particularmente en la zona rural.
- Concretar acciones a favor de atención preferencial a los sectores vulnerables de la sociedad chimboracense.
- Implementar programas de trabajo social, partiendo de las capacidades locales, con metodología participativa, de equidad e interculturalidad que aporten al desarrollo de la sociedad chimboracense.
- Coordinar acciones y propuestas con organismos e instituciones que permitan ampliar coberturas y optimizar recursos.
- Formación y Capacitación de grupos vulnerables
Campañas de atención médica preventiva y curativa para grupos vulnerables.
Proyecto de Comunicación y Promoción Social de los Derechos Colectivos.

Organigrama Estructural de la Organización

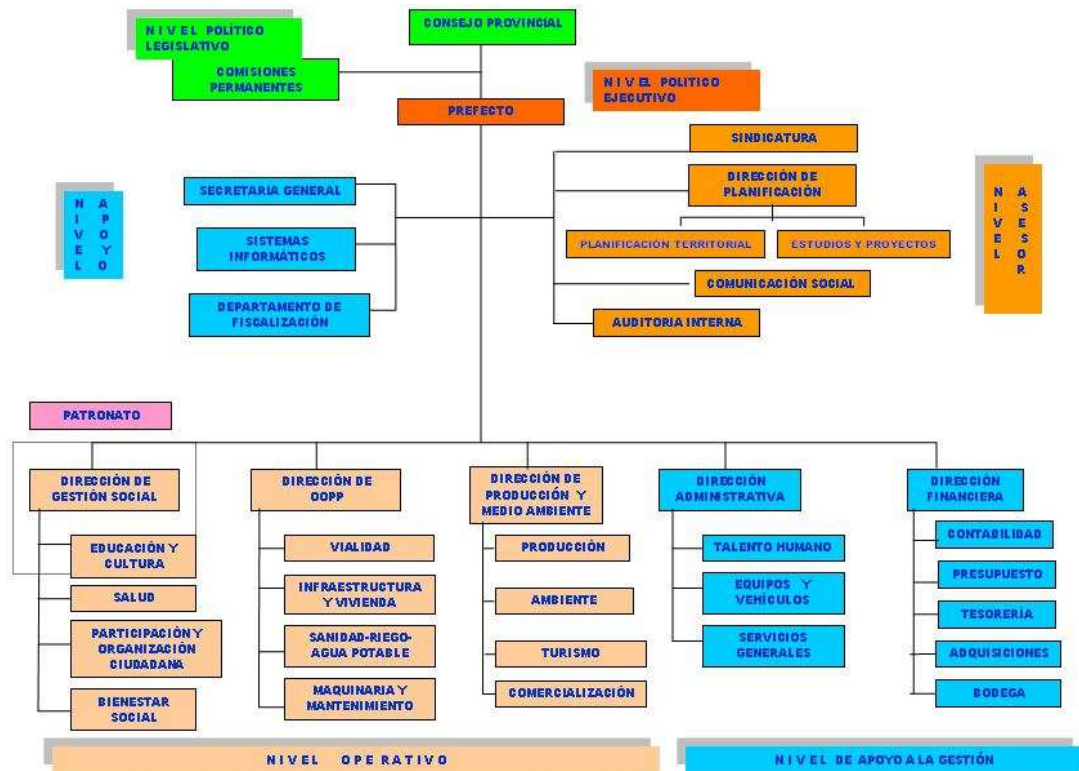


Figura N° IV. 2. Organigrama Estructural de la Organización

4.3.4. Estudio de Factibilidad

Factibilidad Económica

EL HCPCCH cuenta con la infraestructura informática necesaria para la implementación del sistema, es decir que poseen todos los equipos hardware y software por lo que la empresa no tendrá que hacer inversión en la adquisición de recursos.

Factibilidad Económica

Ninguno valor será considerado debido a que el proyecto es realizado como parte practica de la tesis del desarrollador.

Factibilidad Técnica

a. Recurso Humano

El recurso humano con el que se cuenta para el desarrollo del sistema es:

- Ing. Ivonne Rodríguez **Directora**
- Ing. Wladimir Castro **Miembro**
- Ing. Juan Arellano **Técnico de Proyectos (HCPCH)**
- Ing. Guillermo Terán **Director de Planificación (HCPCH)**
- Ing. Jaime Zárate **Jefe del Departamento de Sistemas (HCPCH)**
- Wilian Sánchez **Desarrollador**

b. Recurso Hardware

- Intel Celaron 3.2GHz D
- DVD RW - Faslh Memory Kingston
- Teclado, Mouse, Monitor standard.

c. Recursos Software

- **Sistema Operativo:** Microsoft Windows XP SP2 / Linux (Centoos 4.3)
- **Servidor de Base de Datos:** MySql 4.1.20
- **Servidor Web:** Apache 2.0.48, WEBrick, XSP
- **Tecnologías de Desarrollo:** Ruby 1.8.5, Gem 0.92, Rails 1.1.6, Mono 1.2.5.2.

d. Planificación de Desarrollo

Para poder planificar adecuadamente optimizando tiempo y recursos se ha seguido una planificación ordenada y tratando de ajustarse a fechas. Para lo cual se ha utilizado una herramienta de administración de proyectos flexible y eficaz, la misma que permite controlar los proyectos, manteniendo informados de fechas y realizando un seguimiento de las actividades para supervisar su progreso. El paquete utilizado es Microsoft Project,

donde el Cronograma de Actividades y Diagramas GANTT para el desarrollo de esta investigación.

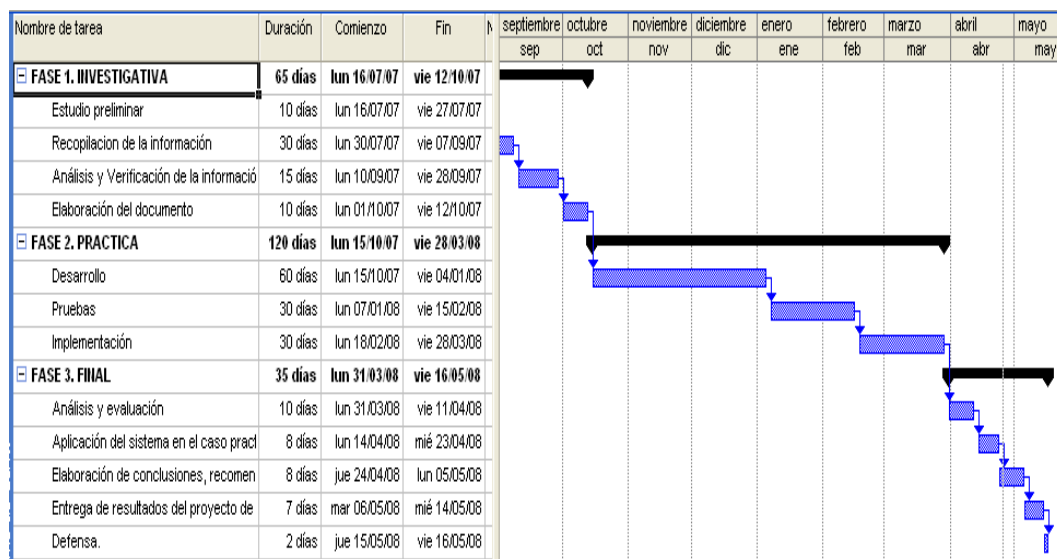


Figura N° IV. 3. Cronograma de Desarrollo

Factibilidad Operativa

Para el desarrollo del sistema se cuenta con el apoyo del Técnico de Proyectos del Departamento de Planificación de HCPCH y el Jefe de Sistemas del HCPCH, al igual que con todo el recurso humano mencionado anteriormente.

Factibilidad Legal

El sistema a ser desarrollado será utilizado únicamente por parte del personal Departamento de Planificación del Consejo Provincial de Chimborazo, además cuenta con la autorización de las autoridades respectivas por lo que no existe ningún tipo de impedimento legal para el desarrollo del sistema.

4.4. FASE II – Análisis del Sistema

4.4.1 Casos de Uso del Sistema

Para la comprensión de los requerimientos se realizar mediante los casos de uso, es decir, descripciones narrativas de los procesos tanto de los usuarios (actores) y como actúa el sistema, mediante un abreve descripción de la utilización del sistema. Se han identificado los siguientes casos de uso para el desarrollo del Sistema de Seguimiento y Evaluación de Proyectos en el Departamento de Planificación del H. Consejo Provincial de Chimborazo: Autenticación de usuarios, Ingreso de información, Reportes, Cuentas de Usuario

4.4.2. Detalle de los Casos de Uso Identificados

Funcionalidad de los Casos de Uso

Los casos de usos identificados anteriormente se detallan a continuación:

Tabla N° IV. 1. Caso de Uso Autenticación de usuarios

Caso de uso:	Autenticación de usuarios	
Actores:	Usuario	
Tipo:	Esencial primario	
Propósito:	Verificar los usuarios activos en el sistema	
Descripción	El usuario debe ingresar y estar activa su cuenta	
Curso de eventos		
ACTOR	SISTEMA	
1. El usuario ingresa al sistema	2. Solicita autenticación (cuenta de usuario y contraseña).	
3. El usuario ingresa su cuenta y contraseña	4. Verifica los datos ingresados para el acceso al sistema.	
5. El usuario ingresa al sistema, y dispone de todas las opciones que se le presentan.		
Cursos alternativos		
6. Si el usuario tiene cuenta limitada no aparecerá la opción de administrador para realizar ingresos de nuevas cuentas de usuarios		

Tabla N° IV. 2. Caso de Uso Ingreso de Información

Caso de uso:	Ingreso de Información	
Actores:	Usuario	
Tipo:	Esencial primario	
Propósito:	Manipulación del sistema (ingreso, actualización, eliminación, reportes)	
Descripción	El usuario realizar consultas hacia el sistema para su propósito.	
Referencias:	Autenticación de Usuarios	
Curso de eventos		
ACTOR	SISTEMA	
1. El usuario escoge opción	2. Presenta submenú	
3. Selecciona un submenú	4. Presenta una para listar, ingresar, editar, eliminar, actualizar los datos del submenú seleccionado.	
5. Escoge consulta	6. Verifica datos para a consulta.	
	7. Visualiza plantilla de consulta	
8. Manipulación de la consulta	9. Visualiza mensajes de la consulta.	
Cursos alternativos		
9. Si los datos no son correctos se muestran mensajes de error y no se realiza la operación.		

Tabla N° IV. 3. Caso de Uso Reportes

Caso de uso:	Reportes	
Actores:	Usuario	
Tipo:	Esencial primario	
Propósito:	Mostrar información de las consultas de usuarios	
Descripción	Muestra reportes de los proyectos, ejes, programas en el sistema.	
Referencias:	Autenticación de Usuarios	
Curso de eventos		
ACTOR	SISTEMA	
1. El usuario escoge opción reportes.	2. Presenta submenú de reportes.	
3. Selecciona un submenú de reportes.	4. Datos a ser consultados	
4. Ingresa datos para la consulta.	5. Verifica datos para a consulta.	
5. Observa los datos de la consulta.	6. Visualiza resultados de la consulta.	
Cursos alternativos		
7. Muestra mensajes hacia el usuario sin no encuentra consulta seleccionada.		

Tabla N° IV. 4. Caso de Uso Cuentas de Usuarios

Caso de uso:	Cuentas de Usuarios
Actores:	Administrador
Tipo:	Esencial primario
Propósito:	Mostrar opción de administrador par al creación de cuentas de usuarios.
Descripción	Ingresar nuevas cuentas de usuarios al sistema, activar cuentas, desactivar cuentas.
Referencias:	Autenticación de Usuarios
Curso de eventos	
ACTOR	SISTEMA
1. El usuario ingresa al sistema como administrador y escoge opción.	2. Presenta las opciones del sistema
3. Escoge opción administrar	4. Presenta lista de usuarios activos y pasivos en el sistema, con opciones de ingreso, actualización, eliminación y listado de usuarios.
3. Selecciona un opción del submenú de administrador.	4. Presenta plantilla de datos a ser consultados.
4. Ingresar datos para la consulta.	5. Verifica datos para a consulta.
6. Observa los datos de la consulta.	7. Visualiza resultados de la consulta.
Cursos alternativos	
7. Muestra mensajes hacia el administrador si no encuentra consulta se ingresaron realizó la consulta seleccionada.	

4.4.3. Diagrama de los Casos de Uso

Los casos de uso se plasman de la siguiente manera.

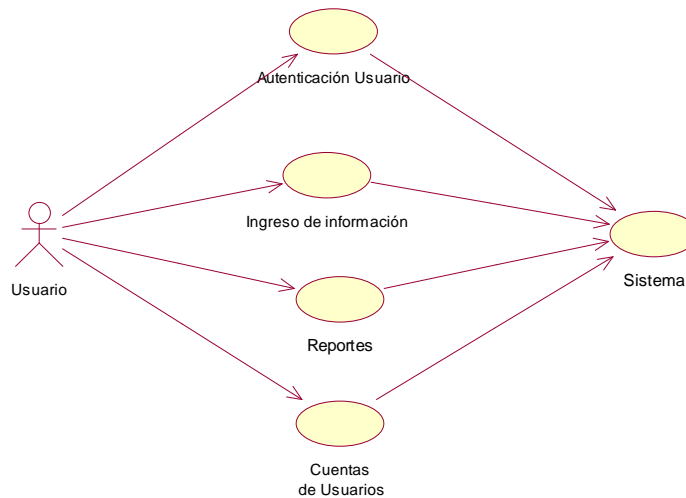


Figura N° IV. 4. Diagrama de Casos de Uso General

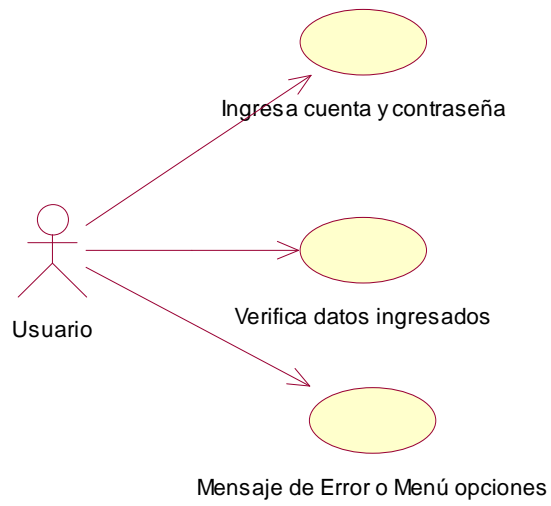


Figura N° IV. 5. Diagrama de Caso de Uso de Autenticación de usuarios

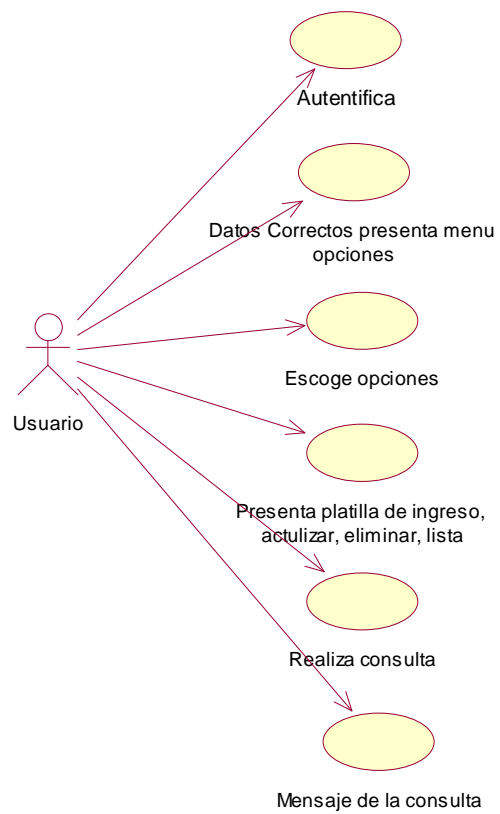


Figura N° IV. 6. Diagrama de Caso de Uso de Ingreso de información

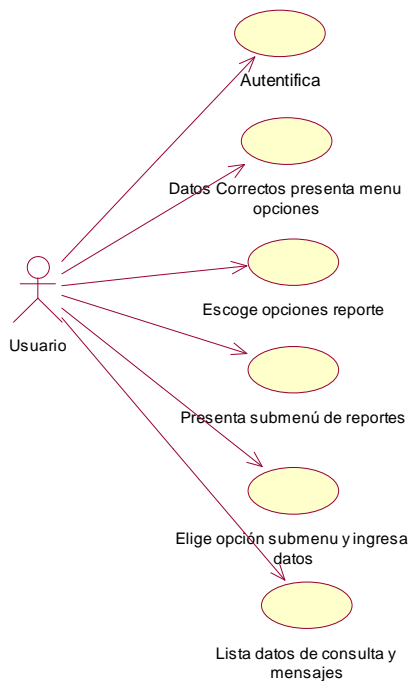


Figura N° IV. 7. Diagrama de Caso de Uso de Reportes

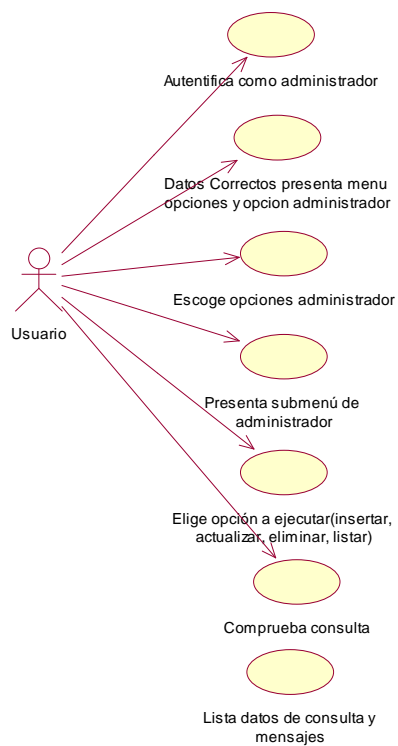


Figura N° IV. 8. Diagrama de Caso de Uso Cuentas de Usuarios

4.4.4. Diagramas de Secuencia

El diagrama de secuencia de un sistema, muestra la representación de un determinado escenario de un caso de uso, los eventos generados por actores externos, su orden y los eventos internos del sistema. A todos los sistemas se les trata como una caja negra; los diagramas se centran en los eventos que trascienden las fronteras del sistema y que fluyen de los actores a los sistemas.

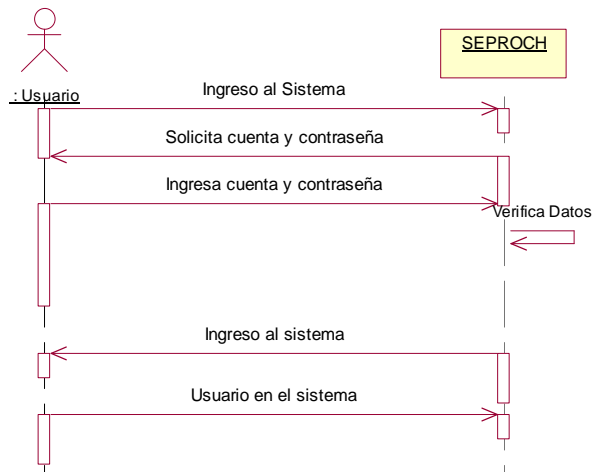


Figura N° IV. 9. Diagrama de secuencia Autenticación de usuario

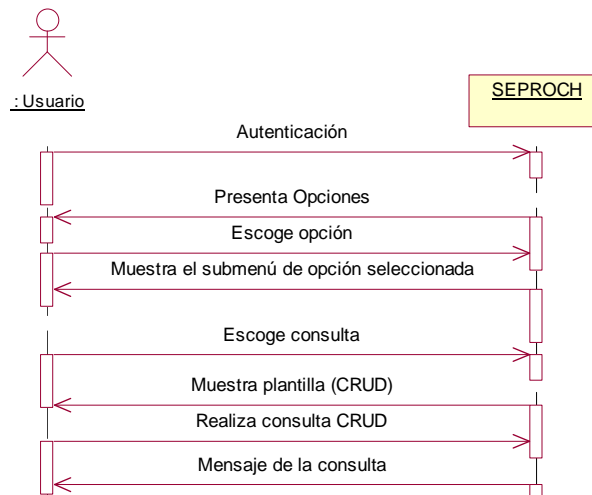


Figura N° IV. 10. Diagrama de secuencia Ingreso de información

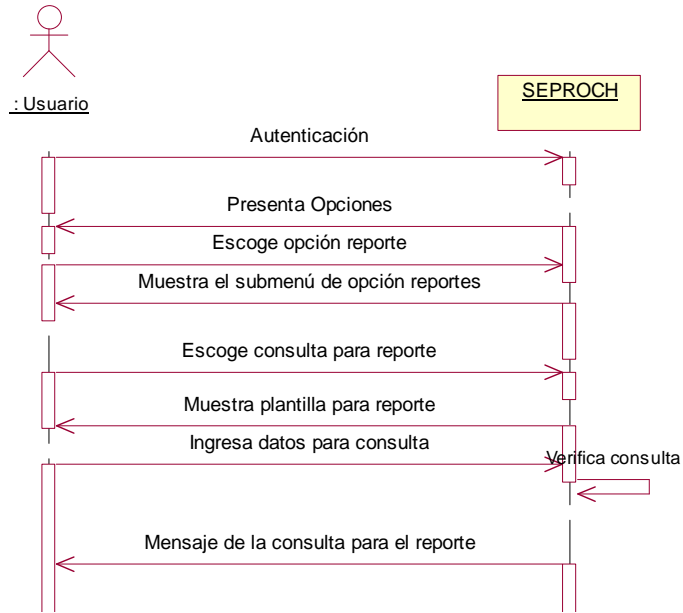


Figura N° IV. 11. Diagrama de secuencia Reportes

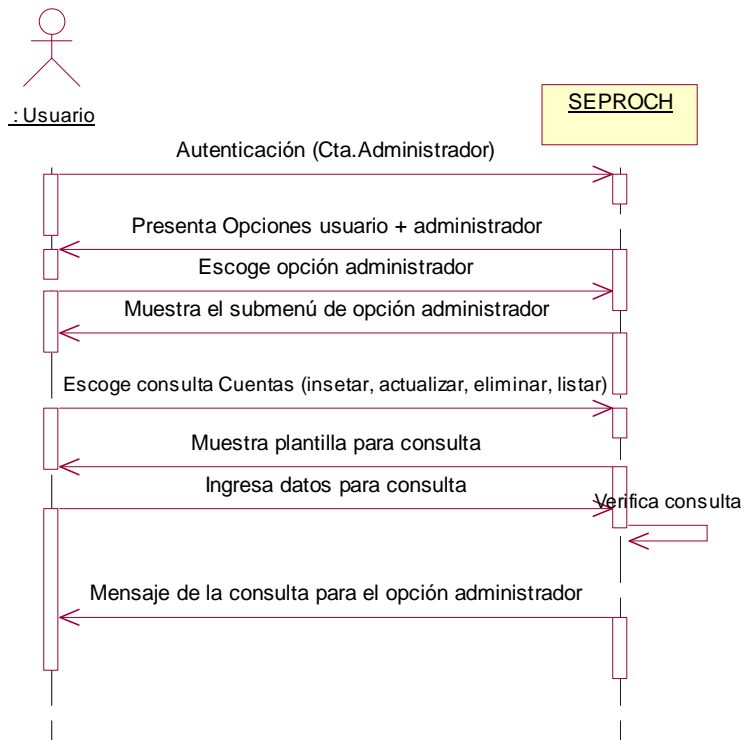


Figura N° IV. 12. Diagrama de secuencia Cuentas de Usuarios

4.4.5. Diagramas de Colaboración

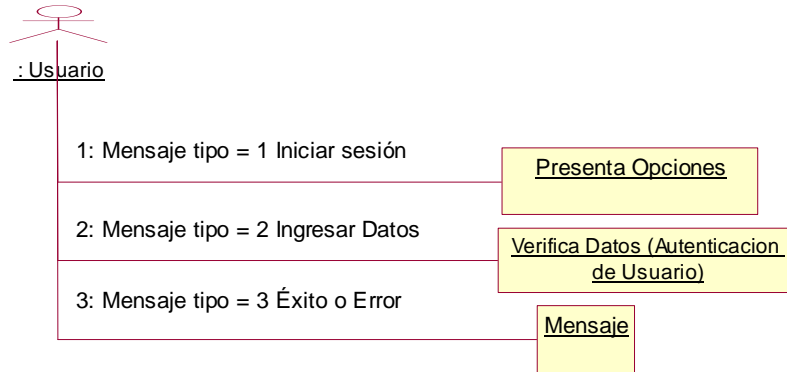


Figura N° IV. 13. Diagrama de Colaboración Autenticación de usuarios

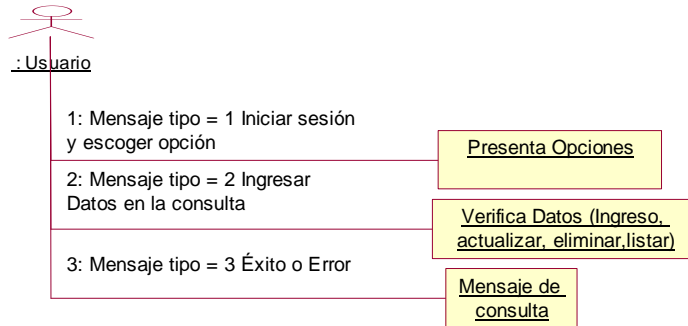


Figura N° IV. 14. Diagrama de Colaboración Ingreso de información

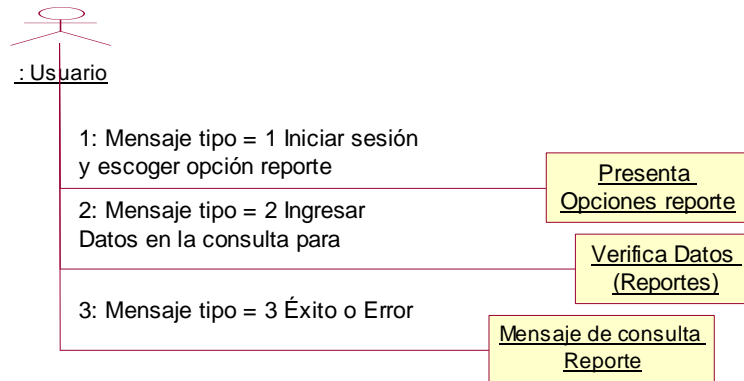


Figura N° IV. 15. Diagrama de Colaboración Reportes

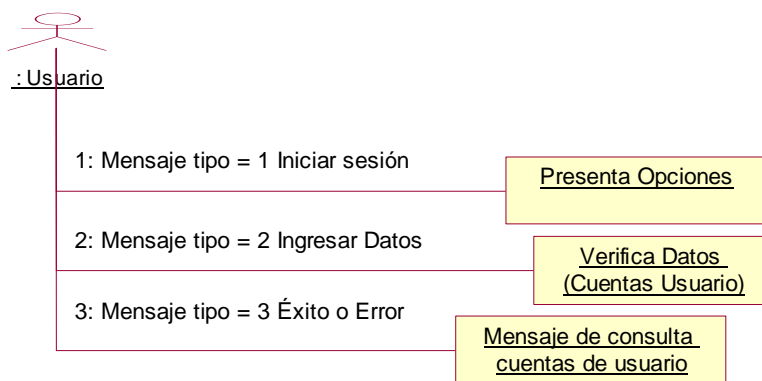


Figura N° IV. 16. Diagrama de Colaboración Cuentas de Usuarios

4.4.6. Diagramas de Calles

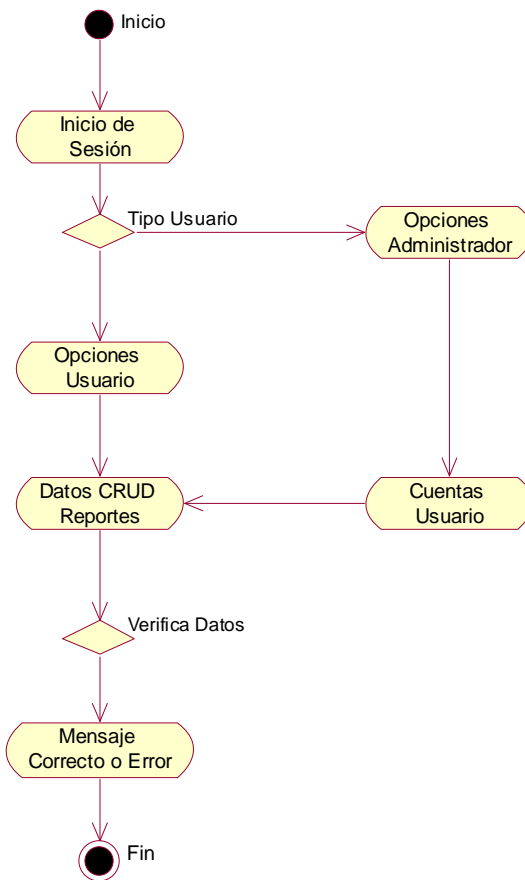


Figura N° IV. 17. Diagrama de Calles

4.4.7. Diagramas de Clases

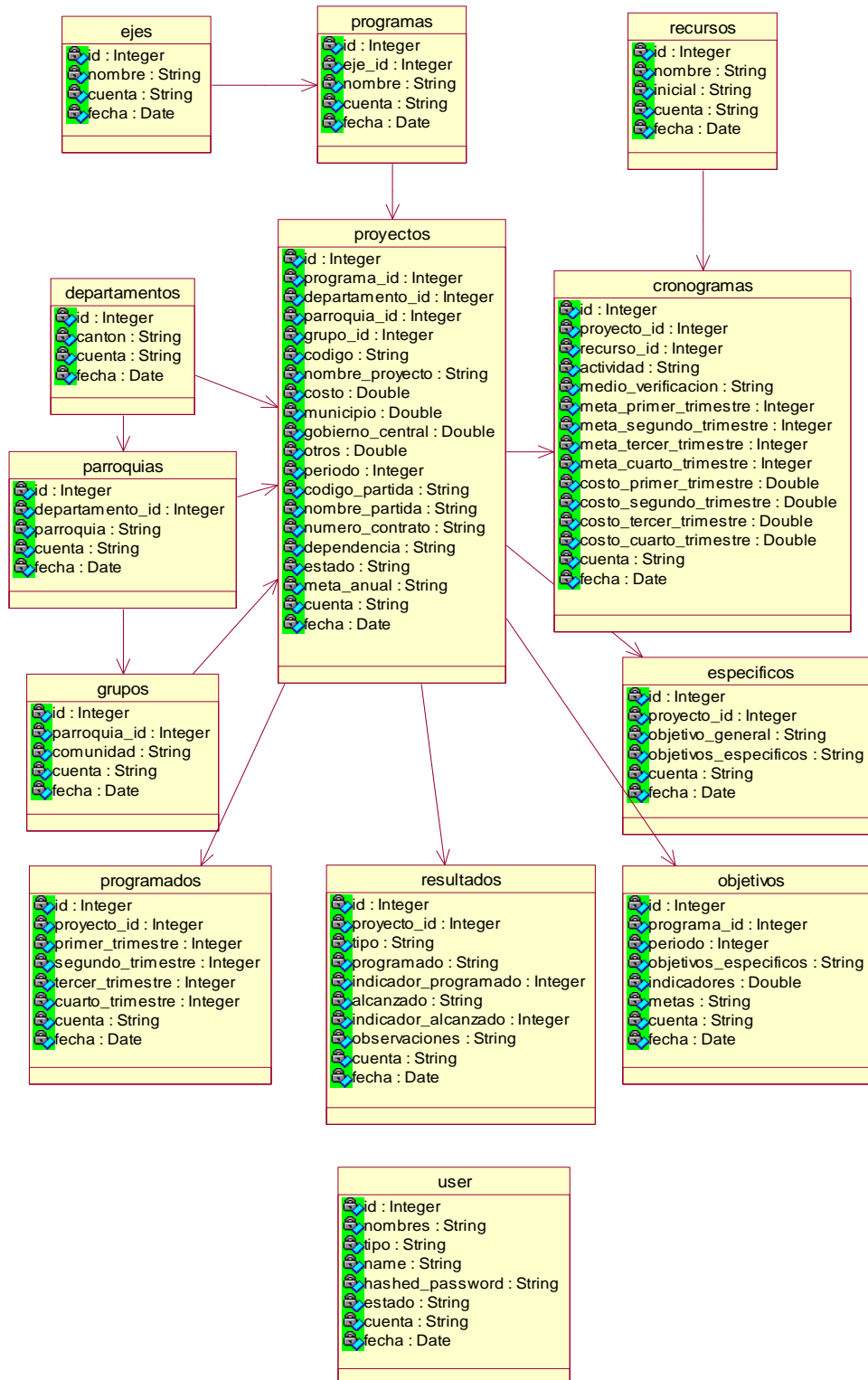


Figura N° IV. 18. Diagrama de Clases

4.4.8. Diagramas de Base de Datos

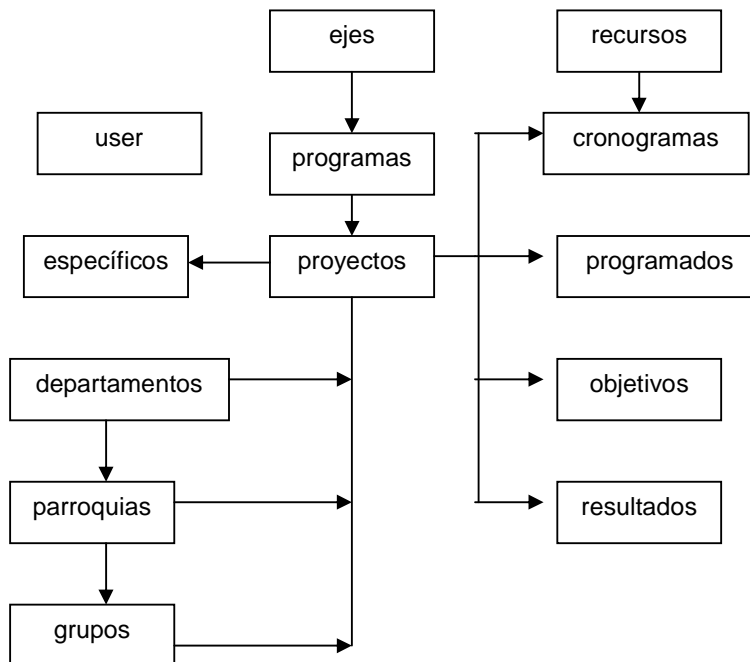


Figura N° IV. 19. Diagrama de Base de Datos

4.4.9. Diagramas de Despliegue

Diagrama de Componentes

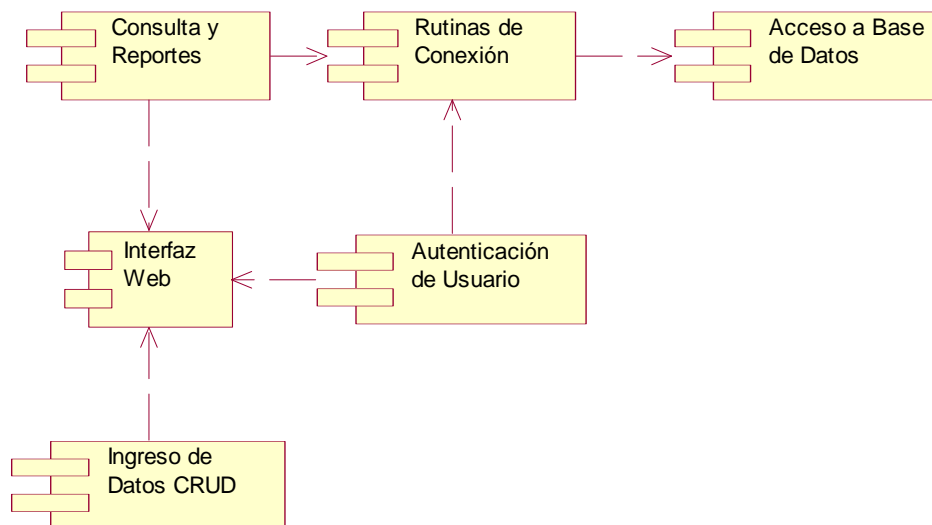


Figura N° IV. 20. Diagrama de Despliegue

Diagrama de Nodos

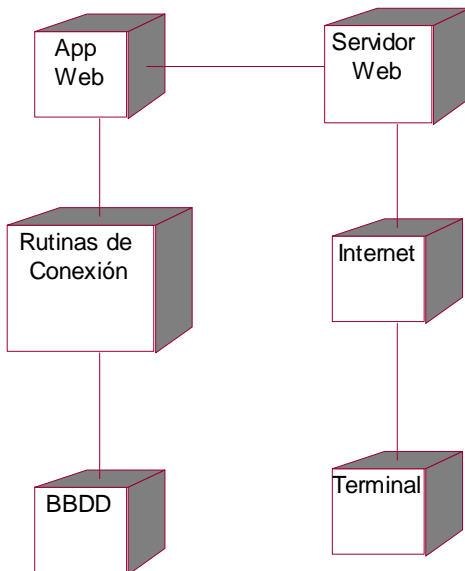


Figura N° IV. 21. Diagrama de nodos

4.5. FASE III – Diseño

Ventana General

En esta ventana es donde se encuentra toda la información sobre los diferentes Cantones de Chimborazo, el inicio de sesión para ingresar al sistema para su respectiva manipulación.

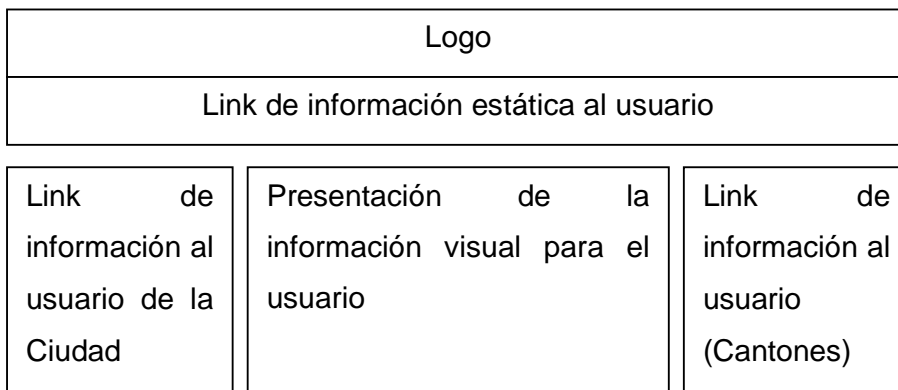


Figura N° IV. 22. Ventana General

Menú de opciones

Para los diferentes ambientes de ingresos de datos al sistema:

- **Administrador**

- Menú para ingreso, actualización, eliminación, listado de cuentas de usuario

Logo
Menú de Opciones para el usuario logueado
Datos de las cuentas de usuario (Ingresar, Eliminar, Actualizar, Listar)

Figura N° IV. 23. Menú Administrador

- **Referencias**

- Menú para ingreso, actualización, eliminación, listado de ejes, programas, metas respectivamente.

Logo	
Menú de Opciones para el usuario logueado	
Submenú Eje Programas Metas	Datos de la selección del submenú para (Ingresar, Eliminar, Actualizar, Listar).

Figura N° IV. 24. Menú Referencias

- **Ubicación**

- Menú para ingreso, actualización, eliminación, listado de cantones, parroquias, comunidades respectivamente.

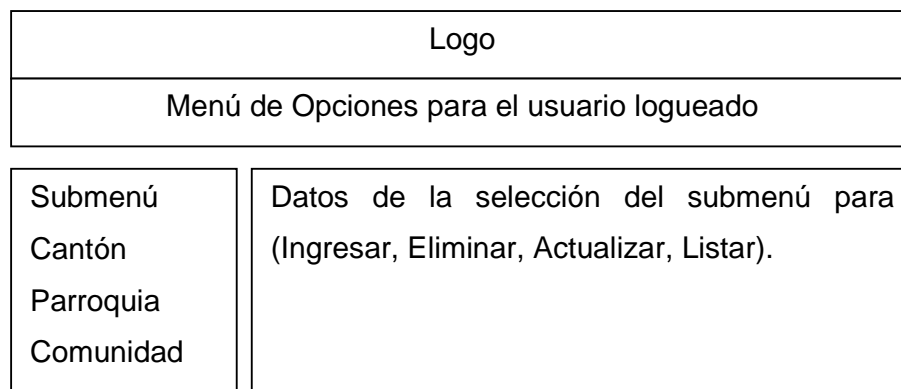


Figura N° IV. 25. Menú Ubicación

- **Proyectos**

- Menú para ingreso, actualización, eliminación, listado de proyecto, actividad, cronograma, recurso, objetivos, resultados respectivamente.

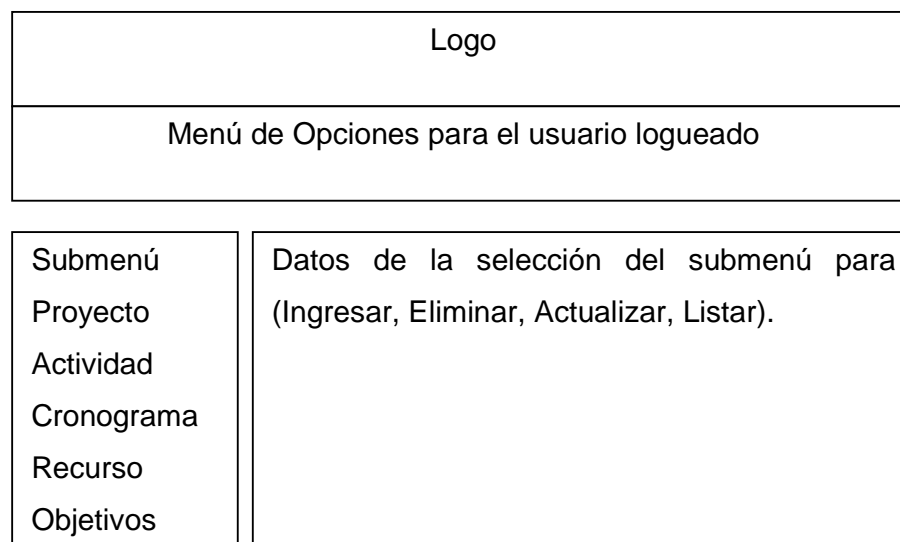


Figura N° IV. 26. Menú Proyectos

- **Reportes**

- Menú para reportes financiero, ejecutivo, estadísticas respectivamente.

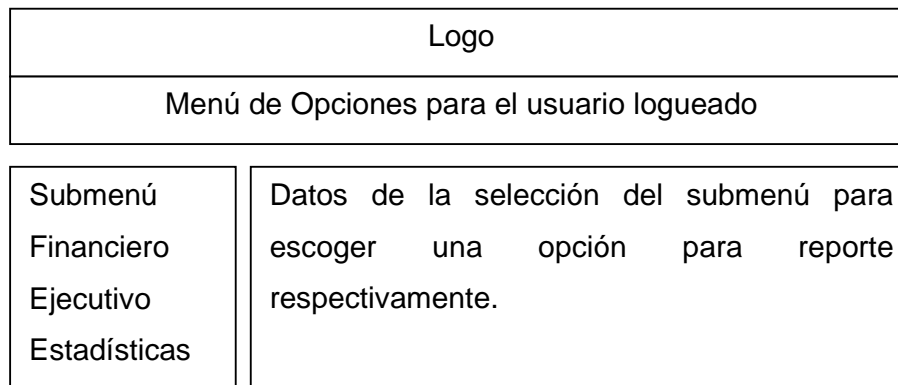


Figura N° IV. 27. Menú Reportes

- **Cerrar sesión.**

- Salir del sistema e ir a la página general.

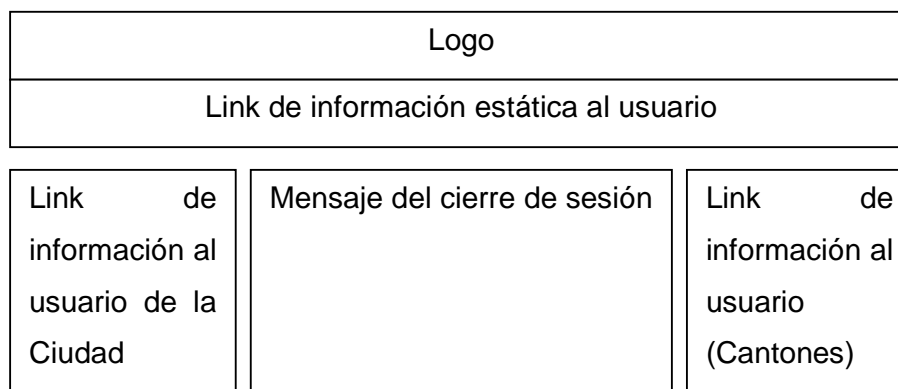


Figura N° IV. 28. Cerrar Sesión

4.5.1. Implementación

La fase de implementación se realizará dependiendo el grado de exactitud del análisis y el diseño siempre y cuando el diseño se lo haya realizado en forma correcta, de acuerdo con los requerimientos del usuario.

Para la implementación de la base de datos, se ha determinado como motor de base de datos MySQL Server 4.1.2, tecnología Ruby on Rails la misma que fue seleccionada mediante el estudio comparativo, Sistema Operativo Windows XP SP2 para las pruebas correspondientes y posteriormente se lo pondrá en el Servidor de Linux (Centos Server 4.3). Se entregará el correspondiente Manual de Usuario (Anexo 1) que permitirá resolver cualquier duda presentada.

4.5.2 Pruebas

Las pruebas para la aplicación es un elemento crítico para la garantía de calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación. Se establece una serie de reglas que sirven acertadamente como objetivos de prueba:

- Es un proceso de ejecución de un programa con la intención de descubrir errores.
- Los casos de prueba son aquellos que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.
- Una prueba tiene éxito si descubre un error no destacado hasta entonces.
- El propósito es diseñar pruebas que sistemáticamente se logre ver las diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo. Si la prueba se lleva a cabo con éxito, se descubrirá errores en el software.

Las pruebas con datos recogidos a medida que se lleva a cabo la prueba proporcionan una buena indicación de la fiabilidad del software y de alguna manera indican la calidad del software como un todo.

El objetivo de la prueba del software es descubrir errores. Para conseguir este objetivo, se planifica y se ejecuta una serie de pasos, y básicamente se realiza una revisión de:

- Código Fuente
- Integridad de Interfaz
- Interacción con el usuario
- Respuestas del sistema
- Manipulación y análisis de datos
- Gestión de base de datos

CONCLUSIONES

1. Mediante el Análisis Comparativo permitió determinar la tecnología Ruby on Rails la más apropiada para el desarrollo del sistema SEPROCH (Seguimiento y Evaluación de Proyectos en Chimborazo).
2. Mediante el desarrollo del Sistema SEPROCH (Seguimiento y Evaluación de Proyectos en Chimborazo) con la tecnología Ruby on Rails, nos permite un desarrollo rápido de la aplicación Web, ayudándonos a realizar su implementación en menos tiempo y con menos esfuerzo para los programadores en esta tarea.
3. Una de las ventajas entre las dos tecnologías nos permite tener un servidor de prueba propio para su respectiva ejecución, ayudándonos a no tener que configurarla en otro servidor Web para la revisión de errores y pruebas, además tiene similitud en el acceso a casi todos las bases de datos analizadas en la comparación y de igual manera se las puede desarrollar en distintos Sistemas Operativos.
4. En el análisis comparativo realizado se determina el porcentaje final para las tecnologías Ruby on Rails y Mono con una calificación de 92,98% (Excelente) y 88,60% (Muy Bueno) respectivamente, en la cual Ruby on Rails se muestra superior a la otra tecnología como es Mono.
5. Mediante el desarrollo e implementación del sistema de Seguimiento y Monitoreo de Proyectos en Chimborazo se a logrado reducir tiempos y errores en la entrega de la documentación con toda la información actualizada y ordenada.

RECOMENDACIONES

1. Se recomienda que hay que investigar la infraestructura y los sistemas informáticos existentes antes de disponerse a la realización de una aplicación en una determina tecnología.
2. Investigar sobre las nuevas versiones de la tecnología Ruby on Rails para poder ayudar a los nuevos programadores en el desarrollo de aplicaciones Web y así expandir la herramienta en nuestro país dando mejores aplicaciones para el futuro.
3. Efectuar un estudio de las facilidades que pueda proporcionar las tecnologías de software libre para la generación de reportes en aplicaciones Web desarrolladas en Ruby on Rails o con otras tecnologías que puedan ser compatibles, ya que en la actualidad hay muy pocas tecnologías que permitan esta opción siendo este un aspecto negativo en los sistemas Web de Software libre.
4. Seguir capacitando a todo el personal del departamento de Planificación y en sí al personal del Honorable Consejo Provincial de Chimborazo que vayan a interactuar directamente con el sistema SEPROCH y así evitar posibles errores de la información por la mala manipulación del Sistema.
5. Brindar una mayor apertura y confianza a los estudiantes Universitarios por parte de las empresas públicas para así poder plasmar ideas nuevas en tecnologías actuales y lograr dar un beneficio directo tanto a la empresa, al estudiante y a la comunidad en sí.

RESUMEN

Se desarrolló un estudio comparativo entre las Tecnologías de Código Libre Ruby on Rails y Mono para el Desarrollo de aplicaciones Web, aplicado al Sistema de Seguimiento y Monitoreo de Proyectos en el Departamento de Planificación del Honorable Consejo Provincial de Chimborazo.

De acuerdo al estudio comparativo y mediante los parámetros de comparación Acceso a Base de Datos, Líneas de Código, Portabilidad, Interfaz de Usuario, se determinó un **92.11%** para la Tecnología Ruby on Rails y **88.60%** para la Tecnología Mono respectivamente; y por ende se concluyó que la Tecnología Ruby on Rails es la mejor opción ya que es multiplataforma y se puede utilizar para el desarrollo de aplicaciones Web con código libre.

Para la implementación y pruebas de módulos se utilizó Ruby 1.8.2, RubyGem 0.9.2, Rails 1.1.6, MySql 4.1.2, Sistema Operativo Centos Server 4.3, XP SP2, donde cada uno de estos software con sus distintas versiones fueron utilizados tanto en Windows como en Linux para su respectivo análisis de comparación.

Se recomienda utilizar este Sistema de Seguimiento y Monitoreo de Proyectos ya que gracias a este sistema se logrará reducir tiempos y errores en la entrega de la documentación con toda la información actualizada y ordenada.

SUMMARY

It developed a comparative study between Technologies Code Free Ruby on Rails and Mono for the Development of Web applications, applied to Tracking System and Monitoring Project in the Department of Planning Honourable Provincial Council of Chimborazo.

According to the study and through the comparative benchmarks Access Database, lines of code, portability, User Interface, was determined by **92.98%** for Technology Ruby on Rails and **88.60%** respectively for Technology Mono, and hence Technology concluded that Ruby on Rails is the best choice since it is multiplatform and can be used for developing Web applications with open source.

For the implementation and testing of modules used Ruby 1.8.2, RubyGem 0.9.2, Rails 1.1.6, 4.1.2 MySQL, Centos Server 4.3 operating system, XP SP2, where each of these software with its various versions were used both Windows and Linux for their respective comparative analysis.

It is recommended to use this Tracking System and Monitoring Project as a result of this system will be achieved times and reduce errors in the delivery of documentation with any updated information and orderly.

GLOSARIO

Applet: Es una aplicación especial que se ejecuta dentro de un navegador o browser.

Browser: Navegador de Internet.

Copyright: Copyright es un concepto general, para proteger actualmente un programa. No permiten a los redistribuidores agregar ninguna restricción adicional cuando éstos redistribuyen o modifican el software.

Copyright: Software protegido por leyes de un país, software que posee dueño

HTTP: (Hiper Text Transfer Protocol). Protocolo de transferencia de Hipertexto. Es un protocolo de aplicación con la sencillez y velocidad necesaria para sistemas de información distribuidos, colaborativos y de diferentes medios.

Intranet: Una intranet es básicamente una Internet diseñada para ser utilizada internamente en una compañía. La principal diferencia entre Internet e Intranet es que la primera es pública y la segunda es privada.

Linux: Sistema operativo Open Source, de utilización libre y sin restricciones en su manejo, se lo denomina software libre.

Mono: Mono es el nombre de un proyecto de código abierto iniciado por Ximian y actualmente impulsado por Novell, el cual permite desarrollar aplicaciones Web.

MySQL: Es un sistema gestor de bases de datos, se trata de una de las bases de datos más rápida actualmente.

PHP: Lenguaje de Programación de uso libre, es decir un lenguaje interpretado de alto nivel embebido en páginas HTML y ejecutado en el servidor.

Perl: Lenguaje de Programación de uso libre para el desarrollo de aplicaciones Web, por ende es un lenguaje de propósito general

Python: Lenguaje de programación orientado a objetos de uso libre para el desarrollo de aplicaciones Web.

Ruby: Es el lenguaje hecho para programar con el menor código posible, seguro, Orientado a Objetos

Rails: es un framework para el desarrollo de aplicaciones Web.

Windows: Sistema Operativo donde Microsoft es el más grande informático que produce y comercializa Windows, el sistema operativo que usa el 90% de los ordenadores personales de todo el mundo.

ANEXOS

ANEXO 1

MANUAL DE USUARIO DEL SISTEMA DE SEGUIMIENTO Y MONITOREO DE PROYECTOS EN EL DEPARTAMENTO DE PLANIFICACIÓN DEL HONORABLE CONSEJO PROVINCIAL DE CHIMBORAZO

INTRODUCCIÓN

El Sistema de Seguimiento y Evaluación de Proyectos en Chimborazo (SEPROCH), es un sistema que provee las facilidades para realizar el seguimiento de los proyectos que se encuentran ejecutándose en la Provincia de Chimborazo y que es administrado por el Departamento de Planificación del H. Consejo Provincial de Chimborazo.

En este documento se hace referencia a la manera eficaz de cómo utilizar al Sistema SEPROCH, el sistema es conformado por dos submenús íntimamente relacionados, la una parte lo conforma el submenú es la del administrador que se ejecutara en una computadora local conectada a la intranet de la empresa y el otro menú funcionará en beneficio de los usuarios que manipularán el ingreso de datos.

La utilización de este sistema debe ser realizado con conocimiento previo, del flujo de información y del área del Departamento de Planificación. Este documento no hace referencia a la enseñanza del proceso y políticas del tratamiento de los proyectos, sino se basa en la forma de hacer fácil la utilización del sistema SEPROCH.

AUDIENCIA

Orientado ha Administradores del Departamento de Sistemas y a Técnicos de Proyectos del Departamento de Planificación para. Es necesario para el administrador tener conocimientos básicos de Administración de Bases de Datos y el flujo del proceso de información para las proyectos que se encuentran en ejecución en el H. Consejo provincial de Chimborazo.

RESUMEN

Este manual es una guía para el administrador y/o usuario del sistema SEPROCH. Tiene por objetivo indicar el proceso para poner a producción el sistema y la forma de cómo utilizarlo.

SOPORTE Y COMENTARIOS

Se agradece todo comentario sobre el presente documento. Para soporte y comentarios comunicarse vía e-mail: wiliansanchez@latinmail.com.

REQUERIMIENTOS MÍNIMOS DE INSTALACIÓN

Los requerimientos mínimos son importantes ya que de ellos depende la buena funcionalidad del sistema. Para lo cual en este capítulo se hará énfasis desde el tipo plataforma, librerías que deben estar instaladas previamente, conexión a la base de datos, etc. Tanto en la parte del Servidor, como en el tipo de computador cliente.

Se debe tener en cuenta que este es el primer paso y deben ser cumplido a cabalidad todos los pasos.

REQUERIMIENTO DE INFRAESTRUCTURA

Con respecto a la infraestructura y con referencia al Hardware es necesario cumplir con los siguientes requerimientos mínimos:

DISPOSITIVO CLIENTE	REQUERIMIENTO MÍNIMO
Procesador	PII o superior
Memoria	256 MB o superior
CD ROM	40X o superior
Espacio en Disco	20GB o superior
Tarjeta de Red	10/100
Puertos	USB 2.0
DISPOSITIVO SERVIDOR	REQUERIMIENTO MÍNIMO
Procesador	PIII o superior
Memoria	256 MB o superior
CD ROM	40X o superior
Espacio en Disco	20GB o superior
Tarjeta de Red	10/100
Puertos	USB 2.0

Nota: A lo largo del documento se utilizara la palabra Local para hacer referencia a una computadora de escritorio y la palabra Servidor para hacer referencia a una Computadora donde se encuentra alojada nuestro Sistema.

REQUERIMIENTO DE PLATAFORMA

SEPROCH V1.0 es un sistema que fue desarrollado en base a la plataforma Linux Centos Server 4.3. Por lo que se considera necesario el tener instalado como mínimo:

CLIENTE
Windows 98, XP, Vista, Linux
Internet Explorer 5 o superior

SERVIDOR		
Centos Server 4.3	PhpMyAdmin 2.11	Apache 2.x.x.
Ruby 1.8.6.	Rails 1.1.6	Mono 1.2.5
RubyGem 0.9.2.	Ruby-MySql	Ruby-fcgi_0.86
MySql 4.1.2	Fcgid_2.4.0	Mod_fcgid_2.2.2.

Cabe señalar que las políticas de seguridad con que se manejen este tipo de aplicación dependerán del administrador de aplicaciones del Departamento de Sistemas.

Nota: Es necesario que el equipo cliente se encuentre conectado a la intranet interna de la empresa con acceso a la base de datos MySQL. Caso que no se hará énfasis en este manual.

REQUERIMIENTO DE BASE DE DATOS

SEPROCH V1.0 trabaja fundamentalmente con la base de Datos MySQL. Por lo cual será necesario que el Administrador brinde el acceso necesario al mismo.

Cabe señalar que las políticas de seguridad y contingencia con que se manejen este tipo Sistema de Manejo de Base de Datos dependerá del administrador del sistema del H. Consejo Provincial de Chimborazo.

INSTALACIÓN Y CONFIGURACIÓN DE COMPONENTES BÁSICOS PARA SEPROCH

Antes que de la instalación del sistema SEPROCH V.1.0 es necesario tener algunos componentes básicos ya instalados y/o configurados. Esto radica en la necesidad de tener en óptimas condiciones las conexiones a la base de datos o los datos base con que el sistema basa su proceso. A continuación se detallará los componentes que deben ser configurados y/o instalados. Este capítulo va orientado fundamentalmente a personas de estudios de tercer nivel en el área de sistemas informáticos, en este capítulo únicamente se expondrá configuraciones esenciales para el sistema SEPROCH.

Install FastCGI

1. `wget http://www.fastcgi.com/dist/fcgi-2.4.0.tar.gz`
2. `tar -xzf fcgi-2.4.0.tar.gz`
3. `cd fcgi-2.4.0`
4. `./configure`
5. `make`
6. `make install`

Install mod_fcgid

1. `wget http://fastcgi.coremail.cn/mod_fcgid.1.09.tar.gz`
2. `tar -xzf mod_fcgid.1.09.tar.gz`
3. `cd mod_fcgid.1.09`
4. `vi Makefile` Change `top_dir` to: `top_dir = /usr/lib/httpd`
5. Uncomment `#INCLUDES` and change to `INCLUDES=-I /usr/include/httpd -I /usr/include/apr-0`
6. `make`
7. `make install`

Install Ruby

1. `cd /etc/yum.repos.d/`
2. `wget http://dev.centos.org/centos/4/CentOS-Testing.repo`
3. `yum --enablerepo=c4-testing install ruby ruby-docs ri ruby-libs ruby-mode`

Install Ruby Gems

1. `wget http://rubyforge.org/frs/download.php/11289/rubygems-0.9.0.tgz`
2. `tar -xzf rubygems-0.9.0.tgz`
3. `cd rubygems-0.9.0`
4. `ruby setup.rb`
5. `gem update`

Install Rails, FastCGI, and Mysql

1. `gem install rails --include-dependencies`
2. `gem install fcgi`
3. `gem install mysql -- --with-mysql-config=/usr/bin/mysql_config`
4. `vi /etc/ld.so.conf` Add line: `/usr/local/lib`
5. `/sbin/ldconfig`

Creamos una Aplicación de prueba.

Estos pasos se pueden omitir ya que la aplicación solo tiene que se copiada en la ubicación que se desee que este dicho sistema.

1. `mkdir /var/www/rails`
2. `cd /var/www/rails/`
3. `rails cookbook`

Nota: `cookbook` es el nombre de la carpeta donde se encuentra el sistema, puede dar el nombre que desea a dicha carpeta

Damos los respectivos permisos

1. `chgrp -R apache cookbook/`
2. `chmod -R g+r cookbook/`
3. `chmod -R g+w cookbook/log/`
4. `chmod -R g+w cookbook/tmp/`
5. `find /var/www/rails/cookbook/ -type d -exec chmod g+x { } \;`

Configure Apache

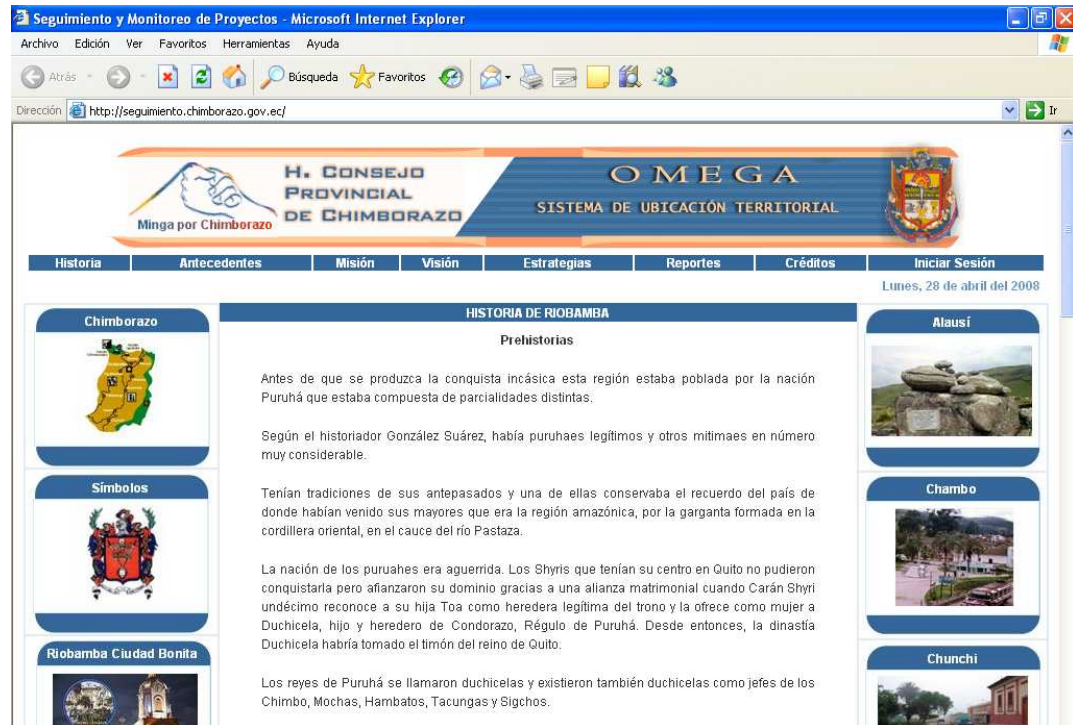
1. `vi /etc/httpd/conf.d/fcgid.conf`
Insertar lo siguiente
`LoadModule fcgid_module /usr/lib/httpd/modules/mod_fcgid.so`
`<IfModule mod_fcgid.c>`
`SocketPath /tmp/fcgid_sock/`
`AddHandler fcgid-script .fcgi`
`</IfModule>`
2. `vi /etc/httpd/conf.d/hosts.conf`
`<VirtualHost *:80>`
`SetEnv RAILS_ENV development`
`ServerName dnsnameoripaddress`
`DocumentRoot /var/www/rails/cookbook/public/`
`ErrorLog /var/www/rails/cookbook/log/apache.log`
`<Directory /var/www/rails/cookbook/public/>`
`Options ExecCGI FollowSymLinks`
`AddHandler fcgid-script .fcgi`
`AllowOverride all`
`Order allow,deny`
`Allow from all`
`</Directory>`
`</VirtualHost>`
3. `vi /var/www/rails/cookbook/public/.htaccess`
Change line: `RewriteRule ^(.*)$ dispatch.cgi [QSA,L]`
To `RewriteRule ^(.*)$ dispatch.fcgi [QSA,L]`
Y Cambiamos:`AddHandler fastcgi-script .fcgi`
To `AddHandler fcgid-script .fcgi`

OPERACIONES DEL SISTEMA SEPROCH

En este capítulo se detalla la forma de operación del sistema SEPROCH en su versión y la forma de administración en la parte de usuarios, la misma que es muy frecuente y bien básica.

INICIO DEL SISTEMA

Para poder ver el sistema en el explorador debemos escribir la dirección <http://seguimiento.chimborazo.gov.ec>, en el cliente que se encuentre en la intranet, caso contrario si no esta el cliente en la intranet el sistema no se podrá visualizar. La aplicación en primera instancia nos mostrará la siguiente pantalla.



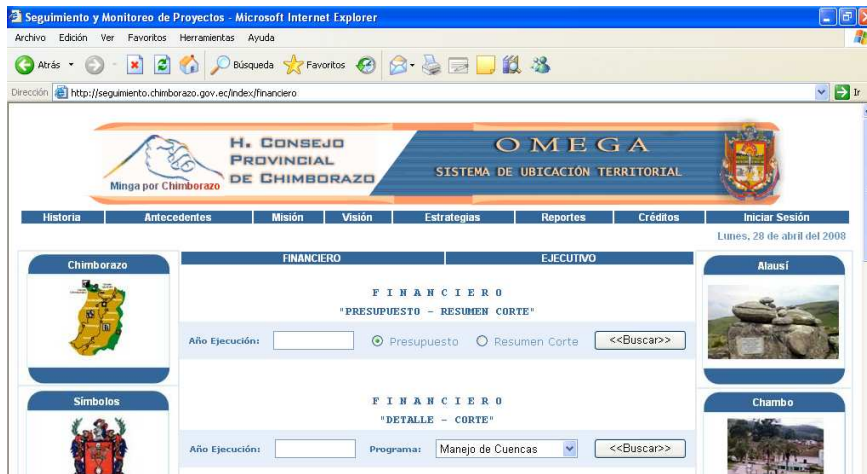
Pantalla de inicio

Cada uno de los link que están en esta pantalla indican nos dan información sobre el H. Consejo Provincial de Chimborazo.

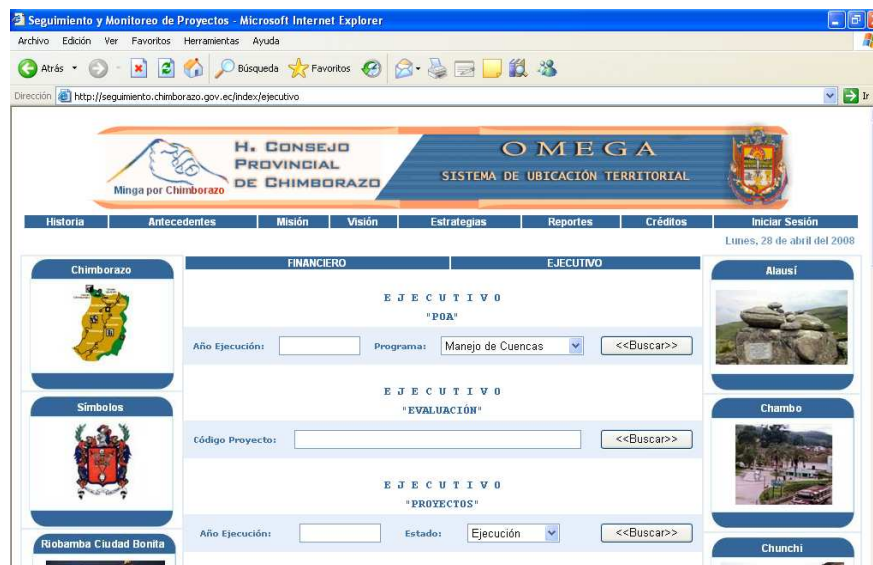
MENÚ REPORTES

El link de Reportes nos indica que podemos ver los reportes de los distintos proyectos que se encuentran en el HCPCH, al cual tienen acceso por parte de la comunidad, dándonos dos opciones de reportes los ejecutivos y los financieros, donde debemos

ingresar los datos requeridos para poder ver el reporte solicitado en cada uno e sus opciones como se muestra a continuación.



Menú reportes financiero



Menú reportes Ejecutivo

Por ejemplo damos clic en el link financiero y procedemos a escribir el año de ejecución de los proyectos y seleccionamos la opción resumen de corte y damos clic en buscar, el cual nos dará el reporte para ese año, si no existiera en ese año el reporte nos dará un mensaje que no existe dicho año, estos mensajes de información al usuario están en

cada uno de sus reportes dependiendo el que se seleccionó, como se puede observa a continuación.

No.	Programas	T.P.	En Ejecución			En Trámite			Pendientes			No Ejecutables			Terminados		
			No.	Costo	%	No.	Costo	%	No.	Costo	%	No.	Costo	%	No.	Costo	%
1	* Salud	22	19500.00	0	0.0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0
2	* Agua Potable	21	45379.52	0	0.0	0.00	9	172816.80	37.85	3	75000.00	16.47	0	0.00	0	0.00	0
3	Letramas	1	44620.47	0	0.0	0.00	0	0.00	1	44620.47	100.00	0	0.00	0	0.00	0	0.00
4	Ciudades Productivas	22	421468.51	0	0.0	0.00	11	244846.12	58.09	0	0.00	0.00	4	80105.22	19.01	0	0.00
5	Educación	7	888221.40	0	0.0	0.00	1	52000.00	0.22	1	20825.20	2.34	0	0.00	0	0.00	0
6	Manejo de Cuentas	33	444609.96	0	0.0	0.00	12	244620.48	54.84	1	5463.12	1.23	1	10000.00	2.22	2	14842.45
7	Riego	45	1108969.66	0	0.0	0.00	4	181501.70	16.37	5	193508.27	17.45	0	0.00	0	0.00	0
8	Sedones Vulnerables	39	530669.57	0	0.0	0.00	4	81000.00	15.26	5	95874.10	18.36	11	59500.95	10.69	0	0.00
9	Turismo	5	192999.89	0	0.0	0.00	2	46917.20	25.28	0	0.00	0.00	1	20000.00	1.04	0	0.00
10	Variedad	12	800000.02	0	0.0	0.00	0	359172.65	36.62	1	143400.47	14.34	0	0.00	0	0.00	0

Reporte Resumen Corte

FINANCIERO
PRESUPUESTO - RESUMEN CORTE

Año Ejecución: Presupuesto Resumen Corte

FINANCIERO
DETALLE - CORTE

Año Ejecución: Programas:

No existe Presupuesto para el Año '2005'

Reporte que no existe con su respectivo mensaje de información

MENÚ CRÉDITOS

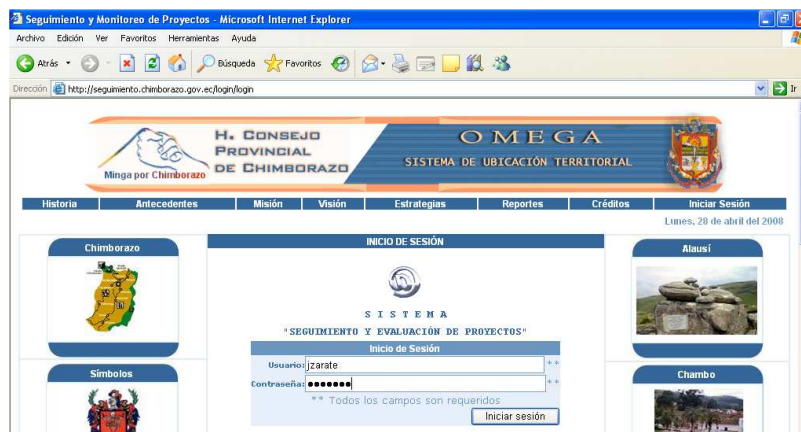
En este link nos da información acerca de los desarrolladores del sistema SEPROCH y los colaboradores en la realización de la misma, como se puede observa a continuación:



Menú Créditos

MENÚ INICIO DE SESIÓN

En este link nos aparecerá una pantalla donde nos pide que ingresemos nuestra cuenta de usuario con su respectiva contraseña, para poder ingresar al sistema e ingresar información en la misma, cabe recalcar que si un usuario tiene cuenta limitada no podrá ver el botón administrador donde este botón solo podrán tener acceso los usuarios con cuentas de administrador, además si el usuario no se encuentra ingresado nos dará mensaje de error informándonos que el usuario no esta registrado o su cuenta está inactiva, por ende para su ingreso se debe pedir una cuenta al administrador del sistema y así poder ingresar al mismo.



Inicio de sesión

MENÚ DE USUARIO REGISTRADO

SIGNIFICADO DE LOS BOTONES DE ACCIÓN USUARIO LIMITADO



En cada uno de los menús se podrá observar esta imagen la cual indica solo el significado de cada imagen (Nuevo, Actualiza, Eliminar, Visualizar, Listar) por lo cual no tiene resultado dar clic en esta imagen, para su respectiva manipulación de ingreso de datos en el sistema.

Estos botones estarán en la parte derecha de cada fila el listado de de datos de cada submenú, para lo cual si queremos hacer una actualización, eliminación, visualización, debemos dar clic en la imagen deseada.

Eje	Actualizado	Cto. Usuario	Acción
Ambiental	2008-02-18	administrador	[New] [Update] [Delete] [View]
Producción y Empleo	2008-02-18	administrador	[New] [Update] [Delete] [View]
Social	2008-04-15	administrador	[New] [Update] [Delete] [View]

Listado de datos

SIGNIFICADO DE LOS BOTONES DE ACCIÓN USUARIO ADMINISTRADOR

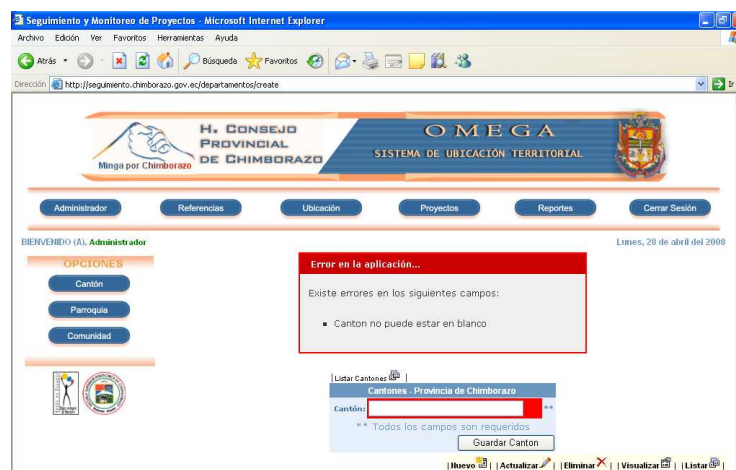


En cada uno de los menús se podrá observar esta imagen la cual indica el significado de cada imagen (Nuevo, Actualiza, Eliminar, Visualizar, Listar), para su respectiva manipulación de cuentas de usuarios. Esta imagen es similar a la anterior con la diferencia que sus imágenes son distintas.



Listado de Usuarios

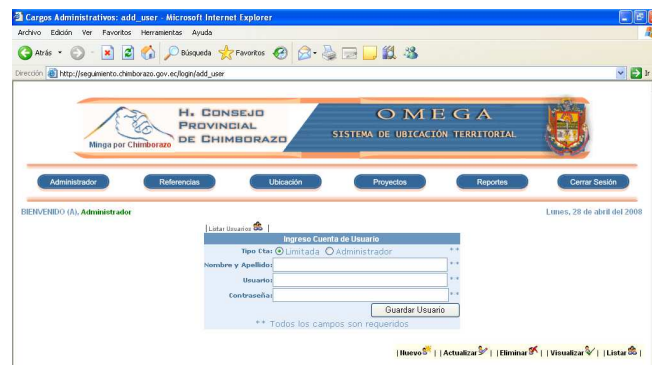
NOTA: En cada submenú del sistema podremos ingresar datos, los cuales están debidamente validados, y en caso de error nos saldrá indicándonos los errores que tenemos en el momento del ingreso de datos.



Mensaje de errores

MENÚ ADMINISTRADOR

El sistema SEPROCH no necesita de una administración especializada una vez configurada correctamente. A continuación se describe las configuraciones básicas que se debe realizar para su trabajo.

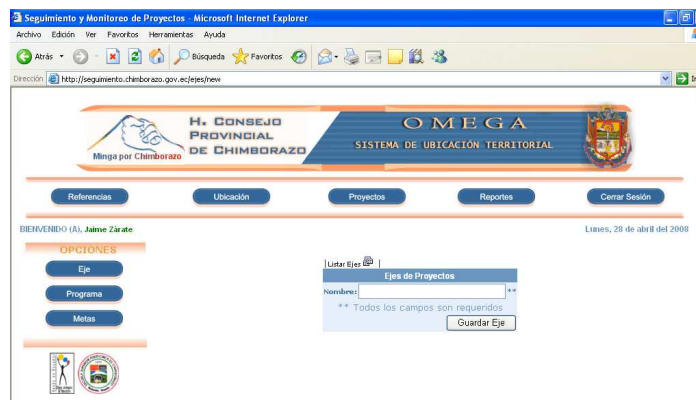


Menú Administrador

En este menú podemos ingresar nuevas cuentas de usuarios, dar de baja, actualizar, visualizar cuentas de usuarios, los cuales estarán a disposición del administrador o administradores del sistema.

MENÚ REFERENCIAS

SUBMENÚ: Eje, Programa, Metas

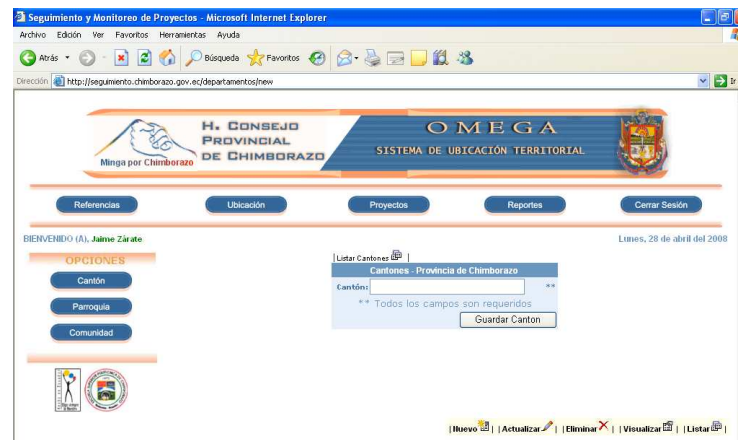


Menú Referencias

- La opción Ejes, y Programas, estos dos van íntimamente ligados ya que un programa pertenece a un eje, es decir si ingresamos un nuevo eje, posteriormente debemos ingresar un nuevo programa para dicho eje, el cual nos servirá para el ingreso de proyectos ya que este necesita saber a cual programa pertenece.
- La opción Metas se deberá ingresar solo una vez ya que esta meta es para cada programa existente en el HCPCH, caso contrario el sistema le informará que ya existe dicho dato ingresado.

UBICACIÓN

SUBMENÚ: Cantón, Parroquia, Comunidad

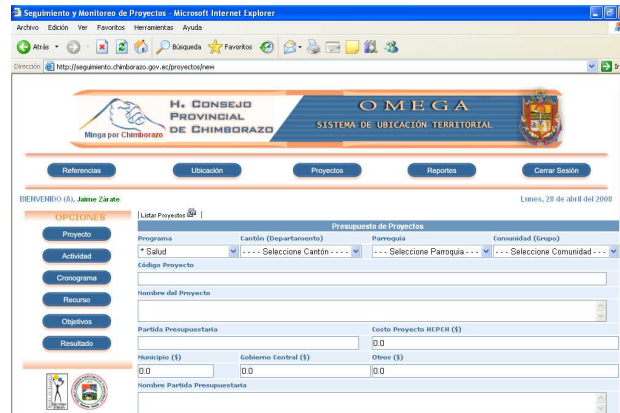


Menú Ubicación

- Tanto el Cantón, Parroquia, Comunidad, existen dato ingresados de todas las comunidades con su respectiva parroquia y cantón

PROYECTOS

SUBMENÚ: Proyecto, Actividades, Cronograma, Recursos, Objetivos, Resultados

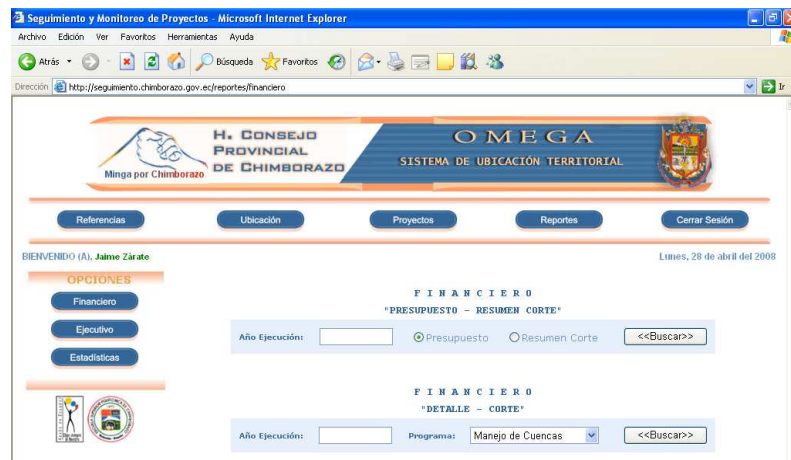


Menú Proyectos

- En primera instancia se debe ingresar un nuevo proyecto, para posteriormente ingresar una actividad, cronograma, objetivos, y resultados, los cuales depende del proyecto ingresado, caso contrario el sistema nos dará mensajes de error diciéndonos que no existe el proyecto en cada uno de submenús.

REPORTES

SUBMENÚ: Financiero, Ejecutivo, Estadísticas (Solo pueden ver los usuarios que tengan una cuenta, ya que en la página principal este reporte no lo observarán los usuarios que no estén registrados)

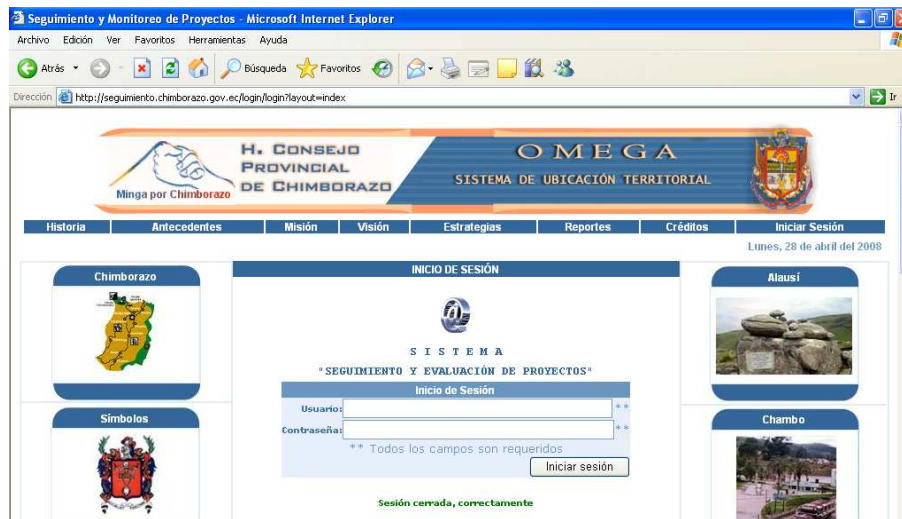


Menú Reportes

- Los reportes Financiero y Ejecutivo son exactamente igual como se explicó anteriormente, para el reporte estadísticas debemos dar clic en el botón en mención he ingresar los datos que pide el sistema para su respectiva visualización, dependiendo de la opción escogida, si no existiera algún dato ingresado el sistema nos devolverá un mensaje de error diciéndonos que no existe dicha opción.

CERRAR SESIÓN

Si damos clic en este botón, automáticamente saldremos a la página principal con un mensaje que nos hemos desconectado del sistema.



Cerrar Sesión

BIBLIOGRAFÍA

LIBROS

- **BLANCO**, Sergio. La Plataforma Libre, ed. Madrid: Proyecto Mono, 2006.
pp. 120.
- **HANSSON**, Heinemeier. Pragmatic Bookshelf. 2da. ed. Texas: Agile Web
Development With Ruby On Rails, 2005. pp. 203.
- **MAXWELL**, S. Centos Server: Herramientas para la administración de redes.
3ra. ed. Madrid: McGraw-Hill, 2004. pp. 497.
- **PETERSON**, R. Red Hat Linux, Manual del Administrador. 3ra. ed. Madrid:
McGraw-Hill, 2004. pp. 396.
- **PRESSMAN**, R. Ingeniería de Software. 4ta. ed. Madrid: McGraw-Hill, 2000.
pp. 800.

BIBLIOGRAFÍA INTERNET

- **ACTIVERECORD Y CALLBACKS**

<http://www.riojasoft.com/articles/04/ruby-on-rails-bases-de-datos-parte-ii>

(2007/06/12)

- **DOCUMENTACIÓN DE LA ECMA EN INTEL**

<http://developer.intel.com/software/idap/ecma>

(2006/10/27)

- **PROGRAMAR EN RUBY**

<http://desdeguate.com/blog/2006/07/10/19-trucos-ruby>

(2006/02/17)

- **PROYECTO MONO**

<http://www.go-mono.com>

(2007/08/28)

- **PROYECTO MONO FAQ**

<http://www.go-mono.com/faq.html>

(2007/09/29)

- **RUBY ON RAILS: RUBY GEM: LICENCIAS DE RUBY ON RAILS**

<http://es.wikipedia.org/wiki/Ruby>

(2007/08/03)

- **WHO IS ALREADY ON RAILS?**

<http://www.rubyonrails.com/>

(2007/09/14)