Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Laboratorio de Distribuido y Paralelismo

Diseño e implementación de un visualizador de desastres naturales y eventos sísmicos geolocalizados para la Fundación Venezolana de Investigaciones Sismológicas

Trabajo Especial de Grado
presentado ante la Ilustre
Universidad Central de Venezuela
por el Bachiller:

Miguel Alfonzo Chang Li Lli Kuin
C.I.: 17.124.006
E-mail: mialchrex@gmail.com

para optar al título de Licenciado en Computación

Tutora: Profa. Joali Moreno

Mayo, 2013.

Universidad Central de Venezuela

Agradecimientos

Agradezco a los profesores de la Universidad Central de Venezuela por todo el conocimiento que me han dado durante estos años de estudio, a todos mis amigos y compañeros de carrera que me han brindado el apoyo para llegar tan lejos, a Funvisis por brindarme esta preciada oportunidad y por último a mis padres que sin ellos no hubiese podido llegar tan lejos.

Indice General

Indice de Figuras

Indice de Tablas

Introducción

La Fundación Venezolana de Investigaciones Sismológicas (Funvisis) posee actualmente un data warehouse (almacén de datos) con información de eventos sísmicos y de siniestros reportados a lo largo del territorio Venezolano. El data warehouse, es una estructura capaz de almacenar una colección de datos provenientes de varias fuentes (departamentos de Funvisis), no posee una interfaz de usuario gráfica que permita consultar los datos almacenados sin recurrir directamente a complejos comandos y opciones dentro del servidor donde se encuentra guardado.

Con el presente trabajo de grado contempla el desarrollo de una aplicación web para Funvisis llamado *"Visualizador de desastres naturales y eventos sísmicos para la Fundación Venezolana de Investigaciones Sismológicas"* en donde se permite consultar y visualizar los datos de los distintos eventos almacenados en el data warehouse de manera gráfica y resumida por medio de cualquier navegador web disponible dentro de la Fundación.

La implementación ha sido llevada a cabo usando el patrón de diseño en el proceso de desarrollo de software MVC (Modelo Vista Controlador) y varias herramientas tecnológicas diferentes, entre las que se destacan:

- Ruby on rails: herramienta principal de desarrollo de software usado en este proyecto.
- Openlayers: interfaz encargada de desplegar mapas digitales en cualquier página web.
- MonetDB: manejador de base de datos en la que se encuentra diseñado el data warehouse.

El desarrollo del proyecto fue llevado a cabo siguiendo un método basado en la programación extrema a lo largo de cinco iteraciones que se resumen a continuación:

- Investigación e instalación de las herramientas necesarias para el desarrollo del proyecto.
- Modificación e implementación de las estructuras y consultas faltantes del data warehouse en MonetDB.
- Desarrollo de la aplicación encargada de conectarse con el data warehouse y procesar datos usando Ruby on Rails
- Diseño y creación de las páginas web usadas para la consulta y despliegue de datos

- Pruebas, correcciones y modificaciones de la aplicación para la puesta en producción.

El resultado obtenido de completar las cinco iteraciones fue una aplicación Cliente-Servidor que permite realizar diferentes consultas a través de una interfaz gráfica y desplegar el resultado de dichas consultas dentro de un mapa digital interactivo de Venezuela y además mostrar indicadores resumen de los datos buscados, todo con el objetivo final de facilitar la búsqueda, análisis y toma de decisiones por parte del personal de Funvisis.

A continuación se describen los capítulos en los que se encuentra dividido el presente documento de investigación:

Capítulo I: Planteamiento del Problema, en el cual se expone el problema, los objetivos a desarrollar, importancia y justificación, propuesta de la solución y alcances de la investigación.

Capítulo II: Marco Teórico, se exponen las diferentes tecnologías investigadas para desarrollar la solución, entre ellas: Openlayers, Ruby on Rails, DataTable etc

Capítulo III: Marco Metodológico, se describe el proceso de desarrollo a utilizar, el cual se basa en la metodología ágil de programación extrema. Se describe el proceso de adaptación de la metodología para el proyecto y se especifica el formato de las historias de usuario, iteraciones, pruebas y roles a ejercer dentro del proyecto.

Capítulo IV: Marco Aplicativo, consiste en la descripción detallada del proceso de desarrollo del proyecto, cada una de las historias usuario generadas y la explicación de cada iteración.

Finalmente se presentaran las conclusiones de la investigación y la bibliografía consultada.

Capítulo I: Planteamiento del problema

En este capítulo se expone las razones principales por la que se desarrolla este proyecto de grado, se explica la problemática presentada dentro la organización Funvisis, así como, la solución planteada con los objetivos a lograr que resuelven dicha problemática.

Planteamiento del Problema

La Fundación Venezolana de Investigaciones Sismológicas cuenta en la actualidad con un data warehouse que contiene la estructura de varios sistemas dentro de la organización, entre ellos, SEISAN y Estudios y Desastres. Actualmente, la aplicación que permite consultar los datos sólo accede a unas pocas dimensiones del data warehouse, existiendo la limitación de que los datos accedidos sólo corresponde a una fracción de las dimensiones existentes, mostrando por lo tanto datos incompletos o carentes de sentido sin las dimensiones faltantes. Adicional a ello, la aplicación presenta los datos como texto plano, sin indicaciones ni resúmenes de lo que representa, lo cual no ayuda a la toma de decisiones de manera rápida y eficiente por el usuario final.

Es por ello que se plantea desarrollar una aplicación web que provea una interfaz de usuario para realizar consultas detalladas de los datos contenidos en el data warehouse y permitir, después de una consulta, la visualización de dichos datos de forma interactiva por medio de un mapa digital de Venezuela con reportes que faciliten el análisis y entendimiento de los resultados mostrados.

Objetivo general

Desarrollo de una aplicación basada en tecnología web para la visualización de eventos registrados en el repositorio de datos históricos de Funvisis.

Objetivos Específicos

1. Desarrollar la consulta de los datos de manera dinámica en función de los parámetros ingresados por el usuario.
2. Diseñar la consulta que permita buscar los datos comunes entre los datos de SEISAN y de Estudios y desastres.

3. Implementar una interfaz interactiva integrada a la aplicación web que permita el despliegue del mapa de Venezuela y los datos relevantes de los eventos sobre dicho mapa.
4. Visualizar los resultados de las consultas en forma de indicadores resumen y en gráficos de barra comparativos.
5. Proveer la opción para descargar un archivo con los datos de la consulta realizada, en forma tabulada compatible con otras aplicaciones que leen información de forma tabulada, por ejemplo: Excel.

Importancia y justificación

Tomando en cuenta que el sistema actual consiste en solo un repositorio de datos sobre eventos y la necesidad de los investigadores de la institución de hacer un uso más útil de toda esa data para obtener información científica que les permita analizar e interpretar dicha data. Se desprende que una herramienta como la propuesta en este proyecto podría permitirles la toma de decisiones, involucrando data de múltiples sistemas de información. De igual manera, la aplicación web servirá de base para nuevos desarrollos que requieran satisfacer futuras necesidades dentro de la Fundación.

Asi, la Fundación contará con una aplicación web con el que se podrán visualizar datos de los sistemas SEISAN y Estudios y Desastres, observar la relación entre los datos y generar reportes desde cualquier navegador web presente en las computadoras con acceso a la red interna de Funvisis.

Propuesta de la solución

Con la finalidad de visualizar la información contenida en el data warehouse de Funvisis, el cual posee datos de dos sistemas: SEISAN y Estudios y Desastres, se propone desarrollar una aplicación web para dicha fundación que permita representar en un mapa digital los distintos eventos presentes en el data warehouse, además de mostrar los datos consultados de una manera sencilla y fácil de analizar e interpretar.

Para ello, es necesario hacer un análisis y revisión del estado del data warehouse y de las distintas consultas implementadas que formaran la base de la aplicación web. Así como, las entrevistas necesarias para obtener los detalles y requerimientos que desea el cliente, en este caso los usuarios.

En este orden de ideas, en primer lugar se realizará el análisis y la revisión del estado del data warehouse, a partir del cual se desarrollarán las consultas que serán usadas por los usuarios finales, a través de una aplicación web.

Posteriormente se realizará el desarrollo de la aplicación web del lado del servidor, usando Ruby On Rails, que funcionará como el motor de procesamiento y manejador de peticiones de los clientes web. Dicha aplicación se desenvolverá en un ambiente web, debido a la facilidad de ejecutar un navegador web desde cualquier computadora dentro de la Fundación, independientemente de sistema operativo o hardware del que estén compuestos, permitiendo una facilidad de uso que no tienen otros sistemas, como por ejemplo, una aplicación de escritorio.

Finalmente, se desarrollaran las vistas de la aplicación web, en este caso, páginas web que podrán ser cargadas en un navegador web, dedicadas a realizar consultas y mostrar información resumida y de manera gráfica de los diferentes eventos sísmicos y siniestros almacenados dentro del data warehouse. En la Figura 1 se muestra un diagrama del sistema planteado.

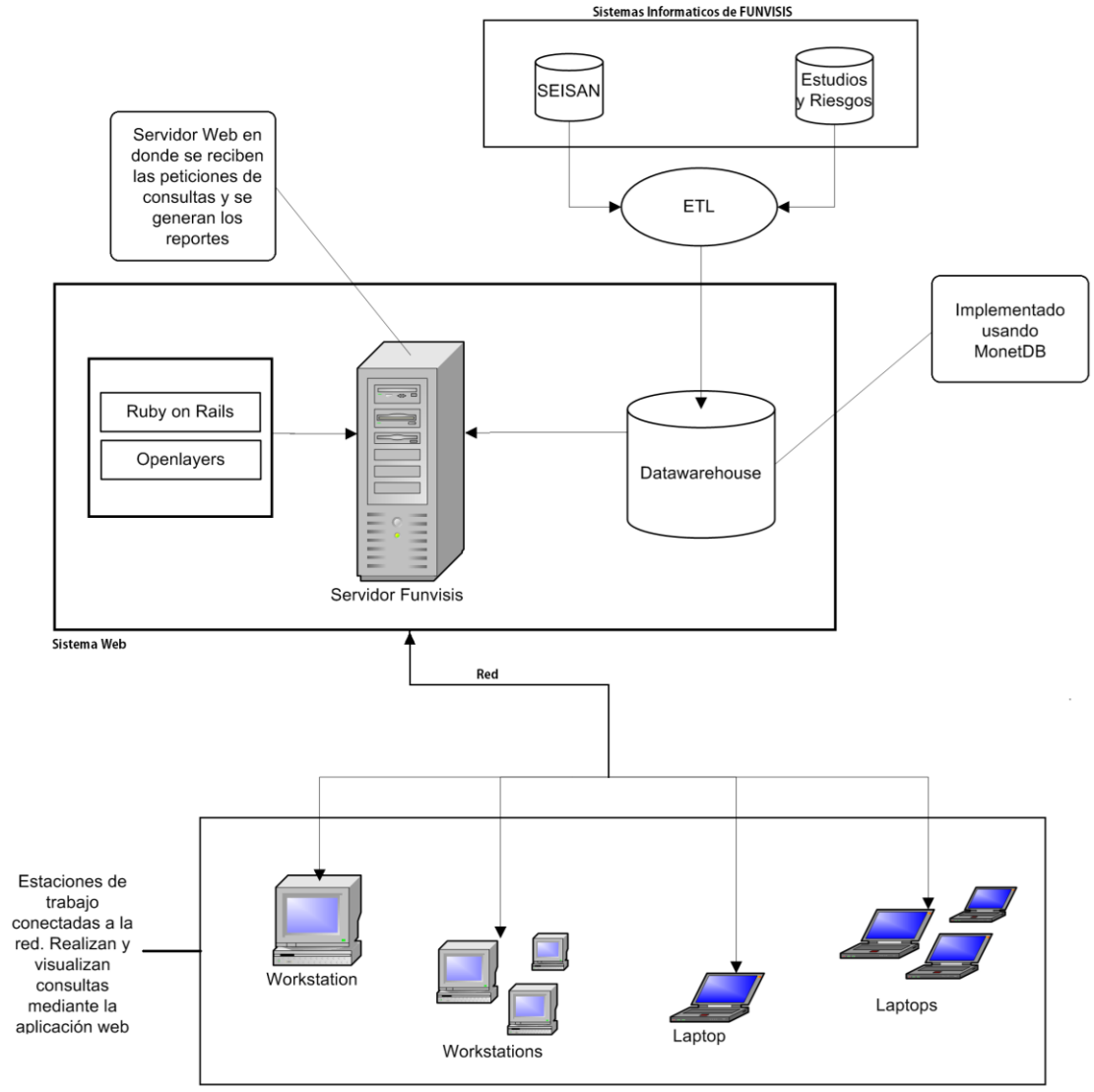


Figura 1: Estructura del Sistema a implementar en Funvisis

Plataforma

Entre las herramientas tecnológicas usadas para el desarrollo de la aplicación web, se encuentran:

- Lenguaje de programación Ruby versión 1.9.2 junto con el framework de programación web Rails versión 3.0
- El manejador de base de datos MonetDB versión 11.04
- Openlayers versión 2.2
- Navegadores Web: Mozilla Firefox version 3.6 en adelante y Google Chrome versión 12.0 en adelante

Alcances y Limitaciones

- El proyecto será desarrollado en un ámbito científico para la institución Funvisis, en base a las necesidades específicas de los investigadores ahí laboran.
- La aplicación web funcionará dentro de la red interna de Funvisis, donde el personal de la fundación podrá tener acceso y utilizarlo desde cualquier computadora.
- Los datos e información mostrados se limitarán a los datos provenientes de los sistemas Seisan y estudios y desastres.

Capítulo II: Marco Teórico

En este capítulo se exponen los fundamentos conceptuales y herramientas tecnológicas que fueron usados durante el proceso de investigación y desarrollo del presente trabajo de grado. Entre ellas se describirán los sistemas de información usados en Funvisis: SEISAN y Estudios y Desastres; se revisará el concepto de data warehouse y del manejador de base de datos por columna MonetDB, con el cual se encuentra implementado el data warehouse que actualmente reside en Funvisis y se encarga de unir ambos sistemas mencionados anteriormente. En cuanto a las herramientras tecnológicas usadas para la implementación de la aplicación web, se realizará un resumen general sobre Ruby on Rails y de las características generales que tiene para el desarrollo de software, también se detallará sobre los sistemas de información geográficos (GIS), sus funcionalidades, tipos y específicamente la librería de Openlayers que permite el uso de GIS en cualquier aplicación web y usado en este proyecto.

Fundación Venezolana de Investigaciones Sismológicas (Funvisis)

La Fundación Venezolana de Investigaciones Sismológicas (Funvisis), adscrita al Ministerio del Poder Popular para Ciencia, Tecnología e Innovación (MPPCTI), es una institución que promueve de forma permanente investigaciones y estudios especializados en sismología, ciencias geológicas e ingeniería sísmica, con el propósito de contribuir a la reducción de la vulnerabilidad en el país. Asimismo, Funvisis, se encarga de divulgar el conocimiento relacionado con las técnicas de prevención a través del programa Aula Sísmica, promueve la formación de personal especializado en el área sismológica y es el ente encargado de instalar, operar y mantener la Red Sismológica y la Red Acelerográfica Nacional [Funvisis 2013].

Misión

Ejecutar y promover, permanentemente, investigaciones y estudios sismológicos destinados a atender la demanda de seguridad en la población ante la amenaza sísmica en el territorio nacional, la formación de personal especializado y divulgar los nuevos conocimientos de las ciencias [Funvisis 2013].

Visión

Ser una organización de excelencia en el área de protección a la colectividad frente a la amenaza sísmica, de referencia nacional e internacional, distinguida por su capacidad de servicio, la calidad de su investigación y su desarrollo técnico y científico [Funvisis 2013]..

Organigrama

A continuación en la Figura 2 se muestra el organigrama representativo de la Fundación Venezolana de Investigaciones Sismológicas (Funvisis).

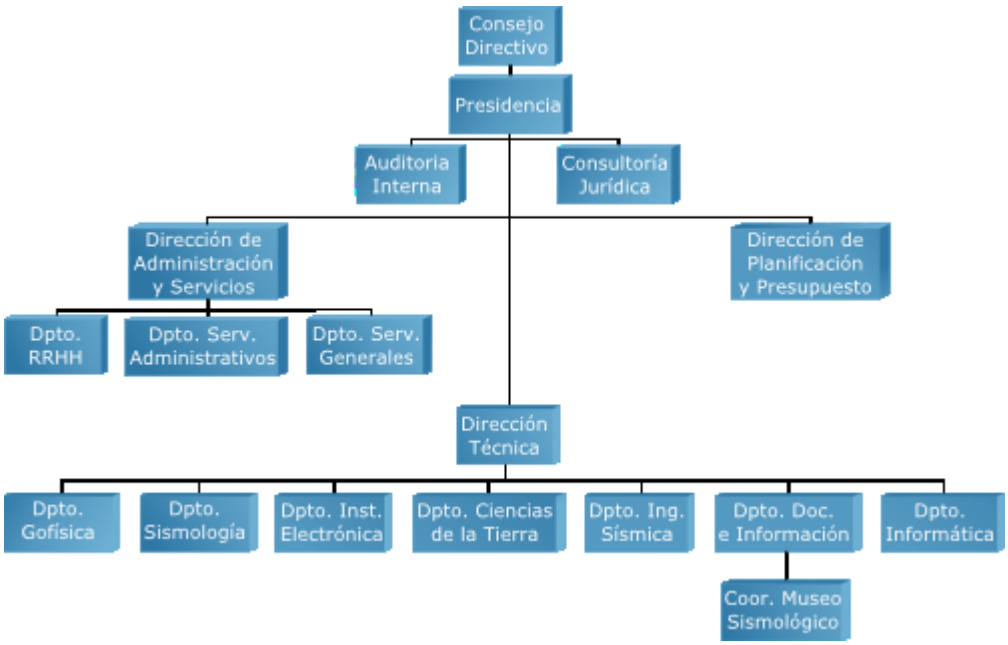


Figura 2: Organigrama Funvisis (http://www.funvisis.gob.ve/estructura_org.php)

Seismic Analysis System (SEISAN)

SeisAn (Seismic Analysis) es un sistema desarrollado por el instituto Físico de Sólidos de la Tierra de la Universidad de Bergen (Noruega) en los años 80 bajo el lenguaje Fortran, dicho sistema permite procesar, recompilar, organizar y almacenar los datos provenientes de distintas de estaciones sísmicas en un sólo directorio de archivos para su posterior análisis, esta característica junto con la naturaleza open source de la aplicación ha hecho que se haya popularizado como el principal programa que muchos institutos de sismología usan en la actualidad [Havskov and Ottemoller 2001].

Funcionamiento

El sistema SEISAN se encuentra dividido en subdirectorios dentro de un directorio principal llamado SEISMO. A continuación se detallan los subdirectorios que conforman el SEISAN y el contenido almacenado en cada uno:

- REA: Lecturas de eventos sísmicos
- WOR: Directorio de trabajo para los usuarios
- TMP: Archivos temporales
- PRO: Programas, códigos fuente y ejecutables
- LIB: Librerías y subrutinas
- INC: Archivos cabecera para programas o subrutinas de los directorios PRO y LIB
- COM: Comandos de procedimiento
- DAT: Archivos de parámentros, Ejemplo: Coordenadas de una estación sísmica
- WAV: Archivos digitales waveform (forma de onda)
- CAL: Archivos de calibración del sistema
- INF: Documentación de los programas
- ISO: Información Macrosísmica
- SUP: Archivos y programas suplementarios

La base de datos de SEISAN consiste de dos directorios, REA y WAV. El directorio REA y sus subdirectorios contienen las lecturas sísmicas y fuentes de información derivadas como hipocentros, soluciones de planos de falla, entre otros. EL directorio WAV almacena todos los Waveform o lecturas de ondas que representan cada evento sísmico almacenado.

Los eventos sísmicos son almacenados en el directorio REA como archivos planos llamados S-files, sólo puede haber un evento sísmico por cada S-file y debe cumplir con una serie de campos requeridos para poder ser almacenado, entre ellos: tiempo de arribo, amplitud, periodo, azimuth y velocidad aparente; independientemente si la fuente de los datos sea analógica o digital. Además, se incluyen los nombres de los archivos Waveforms relacionados con el evento sísmico, si este lo posee. Cada archivo S-file tiene un identificador único (ID) por nombre, el cual es modificable y por lo general representa el estado y último acceso que se ha tenido sobre el archivo. En la figura 3 se puede apreciar la estructura de un archivo s-file.

Input parameters:

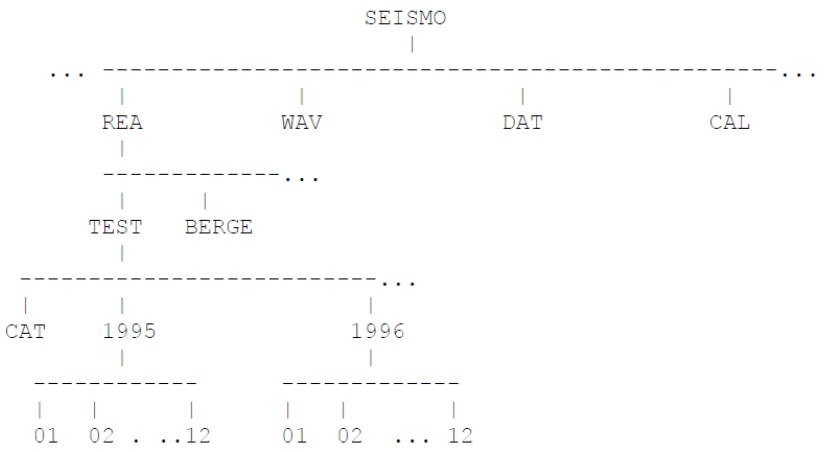
```
STAT SP    : Station and component
IPHAS      : Phase with onset
W          : Phase weight, HYPO71 style
HRMM SECON : Hour, minute and seconds
CODA       : Coda length (secs)
AMPLIT PERI: Amplitude (nm) and period (sec)
AZIM VELO  : Back azimuth (deg) and apparent velocity of arrival at
station
AIN        : Angle of incidence

Output parameters:
AR         : back azimuth residual
TRES       : Arrival time residual
W          : Weigh used in location
DIS        : Epicentral distance in km
CAZ        : Azimuth from event to station
```

```
 1993 1014 1638 24.1 L  58.285   6.356 16.0  BER  8  .8 2.6CBER 2.3LNAO        1
 ACTION:UPD 97-03-25 21:28 OP:jh   STATUS:              ID:19931014163824     I
 1993-10-14-1638-01S.NSN_23                                                    6
     535  SOUTHERN NORWAY, this line is comment                                3
 STAT SP IPHASW D HRMM SECON CODA AMPLIT PERI AZIMU VELO AIN AR TRES W  DIS CAZ7
 KMY  SZ EP       1638 43.75   99                                  .010  122 328
 KMY  SZ ES       1638 59.09                                       .810  122 328
 BLS5 SZ EP       1638 44.83  101                                  .210  127   3
 BLS5 SZ ES       1638 59.60                                      -.210  127   3
 ODD1 SZ EP       1638 53.18  122                                  .910  182   5
 EGD  SZ EP       1638 57.89  122                                 -.3 9  230 343
 EGD  SZ ES       1639 23.23                                      -.2 9  230 343
 BER  SZ EP       1638 59.23  105                                 -.3 9  241 346
 ASK  SZ EP       1639  0.91  113                                 -.2 9  254 344
 ASK  SZ ESG      1639 33.37                                     -1.9 9  254 344
 SUE  SN EP       1639  9.70  156                                  .2 9  322 343
 NRA0 S  EPn      1639 19.25       20.0  1.7  20.4  9.452.0 -8  -.1 9  401  49
 NRA0 S  EPg      1639 26.92            228.7  6.7 6.0  0 -1.9 9  401  49
 NRA0 S  ESn      1640 01.69            232.3  4.7 3.6  3  1.4 9  401  49
 NRA0 S  ELg      1640 15.09            222.4  3.4 6.2 -6  -.4 9  401  49
```

Figura 3: Estructura de un archivo S-file (http://www.geoinstr.com/pub/manuals/seisan_7.2.pdf)

Los eventos dentro de SEISAN son organizados en forma de árbol en años y meses, existiendo un directorio en donde se guardan los eventos correspondientes según la fecha en que se registran. En la figura 4 se puede apreciar con más detalle la forma en la que están almacenados los eventos dentro de los directorios del SEISAN.



\REA\TEST\	Directorio de datos principal
\REA\TEST\1996\	Directorio con los datos de 1999
\REA\TEST\1996\01\	Directorio con los datos de enero 1999
\REA\TEST\1996\01\ 27-1112-11L.S199401	Ejemplo de un evento sucedido un 27 de enero

Figura 4: Estructura de almacenamiento de datos en SEISAN (http://www.geoinstr.com/pub/manuals/seisan_7.2.pdf)

Sistema de estudios y desastres

Estudios y desastres es un sistema informático perteneciente a Funvisis, se encuentra desarrollado bajo PHP con una base de datos MYSQL y funciona bajo la plataforma Linux o Windows.

El objetivo principal del Sistema de Estudios y Desastres es almacenar información respecto a siniestros ocurridos a lo largo del territorio venezolano, cada siniestro es almacenado dentro del sistema como un "evento" para luego ser estudiado y contabilizado.

Un evento, bajo el contexto de Estudios y Desastres, es un desastre natural en donde han ocurrido pérdidas de vidas humanas y/o materiales. Cada evento posee los siguientes campos dentro del sistema:

- Tipo de evento
- Evento Especifico
- Fecha del evento
- Ubicación:
  - Estado
  - Municipio
  - Centro poblado
  - Localidad

    - Dirección
- Magnitud
- Duración
- Radio de afectación
- Fuente
- Causas
- Causa específica
- Descripción
- Personas Afectadas:
    - Nro. de muertos
    - Nro. de desaparecidos
    - Nro. de lesionados
    - Nro. de damnificados
    - Nro. de afectados
    - Nro. de viviendas dañadas
    - Nro. de viviendas destruidas
    - Nro. de Evacuados
    - Nro. de Reubicados
- Pérdidas estimadas (Descripción y monto):
    - Infraestructura Vial
    - Infraestructura escolar
    - Sector vivienda
    - Sector pecuario
    - Sector transporte
    - Sector comunicaciones
    - Sector energía
    - Sector educación
    - Sector salud
    - Sector industria y comercio
    - Sector agua potable
    - Sector agua servidas
    - Otras
- Observaciones
- Imágenes
- Estatus

A diferencia de otros sistemas informáticos presentados en Funvisis, los datos de los eventos usados en Estudios y Desastres provienen de periódicos y portales web. Esto debido a la naturaleza compleja y costosa que presenta investigar cada siniestro independientemente, por lo tanto, la información puede ser inexacta y a veces errónea. Actualmente los eventos registrados en el sistema comienzan desde la década 1940 y se remontan hasta la actualidad.

Data warehouse

Un almacén de datos (DW, *data warehouse* por sus siglas en ingles) es una colección de datos orientados a un tema, integrados, no volátiles e historiados, y organizados para el apoyo de un proceso de ayuda a la toma de decisiones [Inmon 1995]. De igual manera un DW se caracteriza por ser:

- **Integrado:** los datos almacenados en el DW deben integrarse en una estructura consistente, por lo que las inconsistencias existentes entre los diversos sistemas operacionales deben ser eliminadas.
- **Temático:** sólo los datos necesarios para el proceso de generación del conocimiento del negocio se integran desde el entorno operacional. Los datos se organizan por temas para facilitar su acceso y entendimiento por parte de los usuarios finales. Por ejemplo, todos los datos sobre clientes pueden ser consolidados en una única dimensión del DW. De esta forma, las peticiones de información sobre clientes serán más fáciles de responder dado que toda la información reside en el mismo lugar.
- **Histórico:** el tiempo es parte implícita de la información contenida en un data warehouse. En los sistemas operacionales, los datos siempre reflejan el estado de la actividad del negocio en el momento presente. Por el contrario, la información almacenada en el DW sirve, entre otras cosas, para realizar análisis de tendencias. Por lo tanto, el DW se carga con los distintos valores que toma una variable en el tiempo para permitir comparaciones.
- **No volátil:** el almacén de información de un DW existe para ser leído, pero no modificado. La información es por tanto permanente, significando la actualización del DW la incorporación de los últimos valores que tomaron las distintas variables contenidas en él sin ningún tipo de acción sobre lo que ya existía.

Características de un data warehouse

- **Orientado a temas:** una primera característica de un DW es que la información se clasifica en base a los aspectos que son de interés para la empresa. Siendo así, los datos tomados están en contraste con los clásicos procesos orientados a las aplicaciones. Los datos en la base de datos están organizados de manera que todos los elementos de datos relativos al mismo evento u objeto del mundo real queden unidos entre sí. Las diferencias entre la

orientación de procesos y funciones de las aplicaciones y la orientación a temas, radican en el contenido de los datos a nivel detallado. En el DW se excluye la información que no será usada por el proceso de sistemas de soporte de decisiones, mientras que la información de las orientadas a las aplicaciones, contiene datos para satisfacer de inmediato los requerimientos funcionales y de proceso, que pueden ser usados o no por el analista de soporte de decisiones.

- **Variante en el tiempo:** los cambios producidos en los datos a lo largo del tiempo quedan registrados para que los informes que se puedan generar reflejen esas variaciones. Los datos son relativos a un periodo de tiempo (semestre, año, entre otros) y deben ser incrementados periódicamente. Toda la información del DW es requerida en algún momento. Esta característica básica de los *data warehouse*, es muy diferente de la información encontrada en el ambiente operacional. En éstos, la información se requiere al momento de ser accedida.

- **No volátil:** la información no se modifica ni se elimina, una vez almacenado un dato, éste se convierte en información de sólo lectura, y se mantiene para futuras consultas. Los datos almacenados no son actualizados, sólo son incrementados. Cabe destacar que posee dos únicos tipos de operaciones: la carga inicial de datos y el acceso a los mismos.

- **Integrado:** la base de datos contiene los datos de todos los sistemas operacionales de la organización, y dichos datos deben ser consistentes. Integra datos recogidos de diferentes sistemas operacionales de la organización (y/o fuentes externas). Se construye mediante de fuentes de datos múltiples y heterogéneas. Cualquiera que sea la forma del diseño, el resultado es el mismo, la información necesita ser almacenada en el DW en un modelo globalmente aceptable y singular, aun cuando los sistemas operacionales subyacentes almacenen los datos de manera diferente.

Es importante señalar que en un data warehouse se aplican técnicas de limpieza e integración como lo son el asegurar la consistencia en el nombrado en las estructuras codificadas, tipos de datos de los atributos, y demás aspectos entre las múltiples bases de datos. Asimismo cuando los datos se mueven al DW, éstos deben ser transformados según sean los requerimientos operacionales.

Ventajas y desventajas de un data warehouse

Un DW puede dar lugar a una serie de importantes beneficios para una organización. En cualquier caso, su utilización permitirá que la información de gestión sea: accesible, correcta, uniforme y actualizada. A continuación se describen algunas de estas ventajas:

- **Menor coste en la toma de decisiones:** se suprime el despilfarro de tiempo que se podía producir al intentar ejecutar consultas de datos largas y complejas con bases de datos, que estaban diseñadas específicamente para transacciones más cortas y sencillas.
- **Mayor flexibilidad ante el entorno**: un DW convierte los datos operacionales en información relacionada y estructurada, que genera el "conocimiento" necesario para la toma de decisiones. Esto permite establecer una base única del modelo de información de la organización, que puede dar lugar a una visión global de la información en base a los conceptos de negocio que tratan los usuarios.
- **Mejor servicio al cliente:** el hecho de que un DW implique una mayor flexibilidad ante el entorno, tiene una consecuencia directa en una mayor capacidad para responder a las necesidades de los clientes.
- **Rediseño de procesos:** un DW ofrece a los usuarios una capacidad de análisis de la información de su negocio, que tiende a ser ilimitada y que permite con frecuencia obtener una visión más profunda y clara de los procesos de negocio propiamente dichos, lo que a su vez permite obtener ideas renovadoras para la rediseño de los mismos.

Por otra parte es importante señalar que utilizar data warehouse también plantea algunos inconvenientes, entre los cuales se encuentran:

- **Altos Costos:** a lo largo de su vida los almacenes de datos pueden suponer altos costos. El almacén de datos no suele ser estático. Los costos de mantenimiento son elevados.
- **Tiempo de vida útil:** los almacenes de datos se pueden quedar obsoletos relativamente pronto.
- **Consultas:** a veces, ante una petición de información estos devuelven una información sub óptima, que también supone una pérdida para la organización.

Arquitectura de un *data warehouse*

La arquitectura de un DW viene determinada por su situación central como fuente de información para las herramientas de análisis [Bouman and Dongen 2009], con base a ello a continuación se describen los componentes mostrados en la Figura 5.

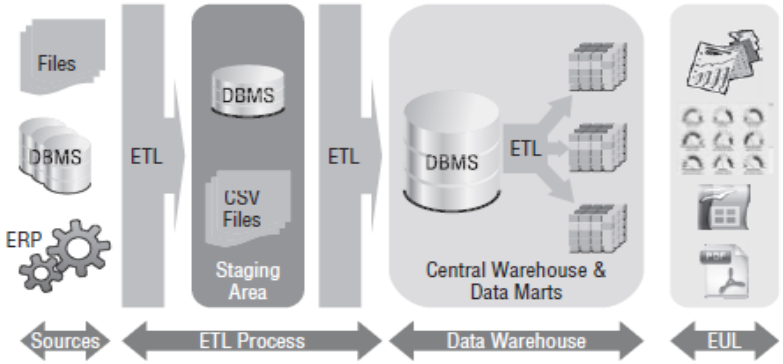


Figura 5: Arquitectura de un data warehouse (Bouman & Dongen, 2009)

- **Fuente de Datos:** uno o más sistemas de fuente, como lo son archivos, sistemas manejadores de bases de datos (DBMS, por sus siglas en ingles), entre otros.
- **Un proceso de Extracción, Transformación y Carga de los datos**: realiza las funciones de extracción de las fuentes de datos (transaccionales o externas), transformación (limpieza, consolidación, entre otros) y la carga del almacén de datos (ordenación, agregación, entre otros), e incluso refrescamiento del almacén, es decir, la operación periódica que propaga los cambios de las fuentes externas al almacén de datos. A menudo este proceso contiene un área de ensayo utilizada como área intermedia en donde se extraen los datos y se realiza la transformación de los datos iniciales y su limpieza. Los datos de ensayo pueden ser usados tanto una base de datos como en ficheros planos. En muchos casos mediante archivos planos se permite un procesamiento más rápido.
- **El almacén de datos:** que consiste en la base de datos del almacén central y cero o más data Marts. Posee información relevante, metadatos. Los metadatos son básicamente datos acerca de los datos contenidos en el DW, es la manera de describir propiedades de las bases de datos y sus atributos, incluyendo tablas y nombres de las columnas, atributos de columnas (tamaño y tipo de dato) de las tablas de las bases de datos, así como claves primarias y relaciones con claves foráneas. En sí el data warehouse es la base fundamental para establecer la completa integración de los datos de la empresa.

- **La capa de usuario final identificada como (EUL)**: se refiere a las diversas herramientas para trabajar con los datos, como lo son informes, cuadros de mando, hojas de cálculo y los documentos publicados. Generalmente, la combinación del almacén central y los data marts es considerada como el almacén de datos, y el término de data warehouse es utilizado para denotar el proceso completo de construcción, carga y la gestión de los datos del almacén.

MonetDB

El sistema manejador de bases de datos MonetDB es un sistema de gestión de alto rendimiento y de código abierto, desarrollado en el instituto de investigación nacional para las matemáticas y de informática (CWI; Centrum Wiskunde & Informática) en los Países Bajos. Fue diseñado para proporcionar alto rendimiento en preguntas complejas contra bases de datos grandes [MonetDB 2012]. MonetDB se ha aplicado con éxito en actividades de alto rendimiento para Explotación minera de datos, OLAP, pregunta XML, recuperación del texto y de las multimedias, entre otras.

La representación de datos internos de MonetDB es almacenada en la memoria, confiando en las gamas enormes del registro de dirección de la memoria de CPU contemporáneas, y así saliendo del DBMS tradicional diseña la participación de la gerencia compleja de los almacenes grandes de los datos en memoria limitada.

MonetDB usa un modelo de almacenamiento basado en la Fragmentación vertical, que guarda los datos por columnas, lo que a menudo da una velocidad de respuesta hasta diez veces más alta sobre otros manejadores de bases de datos tradicionales usando el mismo algoritmo de consulta de datos.

La familia de MonetDB consiste en:

- **MonetDB/SQL**: La base de datos relacional implementada y almacenada en columnas.
- **MonetDB/XQuery**: Implementación de base de datos usando XML como formato de datos
- **Servidor de MonetDB**: el servidor de base de datos que recibe y responde peticioens de consulta

Características de MonetDB

A continuación se presentan una serie de tablas, con un resumen de las características provistas por el sistema manejador de bases de datos MonetDB:

Tabla 1: Información General MonetDB

Información General	Mantenedor	Primera Fecha de Lanzamiento	Ultima versión	Última Fecha de Lanzamiento	Licencia de Software
MonetDB	El equipo de desarrolladores MonetDB	2004	5.6	06 2008	MonetDB Licencia Pública v1.1

Tabla 2: Sistemas Operativos Soportados por MonetDB

Sistemas Operativos	Windows	Mac OS X	Linux	BSD	UNIX	AmigaOS	Symbian	z/OS [1]
MonetDB	SI	SI	SI	NO	SI	NO	NO	NO

Tabla 3: Capacidades de MonetDB

Capacidades	MonetDB
ACID	SI
Integridad Referencial	SI
Transacciones	SI
Unicode	SI
Tabla Temporal	SI
Vista Materializada	NO

Arquitectura de MonetDB

En esta sección se presentan brevemente el servidor MonetDB y el compilador de SQL. Una creciente clase de los motores de bases de datos están orientados a la explotación de un almacén de datos orientado a columnas.

En este campo, las tablas relacionales se dividen verticalmente en cada columna que representa un atributo relacional único. Cada columna se almacena en una tabla separada o incluso una matriz común, pero con una implementación que se orienta a explotar de manera

óptima esta estructura. Este enfoque conduce a una arquitectura de sistema muy simplificado y abre muchos caminos para aumentar el rendimiento. Los beneficios provienen de una mejor racionalización del flujo de datos desde el disco a través de la memoria en la caché de la CPU. El esquema de almacenamiento de datos orientados a columnas son especialmente beneficiosos en data warehouse y aplicaciones de minería de datos, que a menudo se utilizan en campos científicos. La razón principal es que la mayoría de las aplicaciones no necesitan los cientos de columnas de una tabla relacional con los datos de medición científica, sino que simplemente requieren una labor de sólo unas pocas a la vez, para el análisis estadístico.

El beneficio inmediato del enfoque de las columnas en un data warehouse, es que sólo los datos que son relevantes para el proceso son los que se traen desde el disco. MonetDB es una pilar completamente funcional del almacén de datos desarrollado a lo largo de una década en el CWI (Centrum Wiskunde & Informática). Se trata de una arquitectura de dos capas de un servidor de base de datos y una serie de interfaces. En la actualidad dispone de interfaces que proporcionan una para SQL y una para la interfaz XQuery con el servidor de base de datos. El servidor se aborda en un lenguaje propio, llamado lenguaje para MonetDB ensamblador (LMA). LMA es un lenguaje de álgebra relacional que soporta una gran colección de primitivas relacionales, funciones y fácil vinculación con las funciones definidas por el usuario.

Este enfoque simplifica considerablemente los compiladores de interfaces, la interfaz analiza las consultas SQL y los compila en los planes de LMA semi-optimizado que explotan el lenguaje SQL y la semántica del esquema. Sólo se centran en la reducción de volumen posible. El compilador de aplicaciones para usuario también selecciona los componentes LMA optimizador para ser activado, por ejemplo, eliminación de la expresión común, la eliminación de código muerto, paralelismo, entre otros. De esta manera, un optimizador de la arquitectura de tres niveles se logra con una clara división de tareas. La capa inferior se centra en la optimización operacional con el estado actual de la máquina. La capa superior está orientada a la explotación de la semántica del esquema, y la capa media está orientada a las decisiones tácticas.

Data warehouse de eventos sismológicos y desastres naturales

El data warehouse de eventos sismológicos y desastres naturales es un repositorio histórico implementada en MonetDB y desarrollado por el Licenciado Enrique Buono como proyecto de grado, con título: "*Desarrollo de un Repositorio de Datos para la Fundación Venezolana de Investigaciones Sismológicas*".

El principal objetivo del desarrollo del data warehouse fue la consolidación de los datos existentes de dos sistemas informáticos de Funvisis, SEISAN y el sistema de Estudios y desastres, en un solo repositorio histórico de datos con la finalidad de realizar consultas sin necesidad de recurrir a la fuente original de los datos.

El resultado final del trabajo es un data Warehouse que puede extraer, cargar y almacenar eventos de ambos sistemas de información, mencionados anteriormente, e inclusive relacionar datos comunes de entre ambos sistemas. La estructura de la base de datos que une ambos sistemas se puede apreciar en la Figura 6 presentada a continuación:

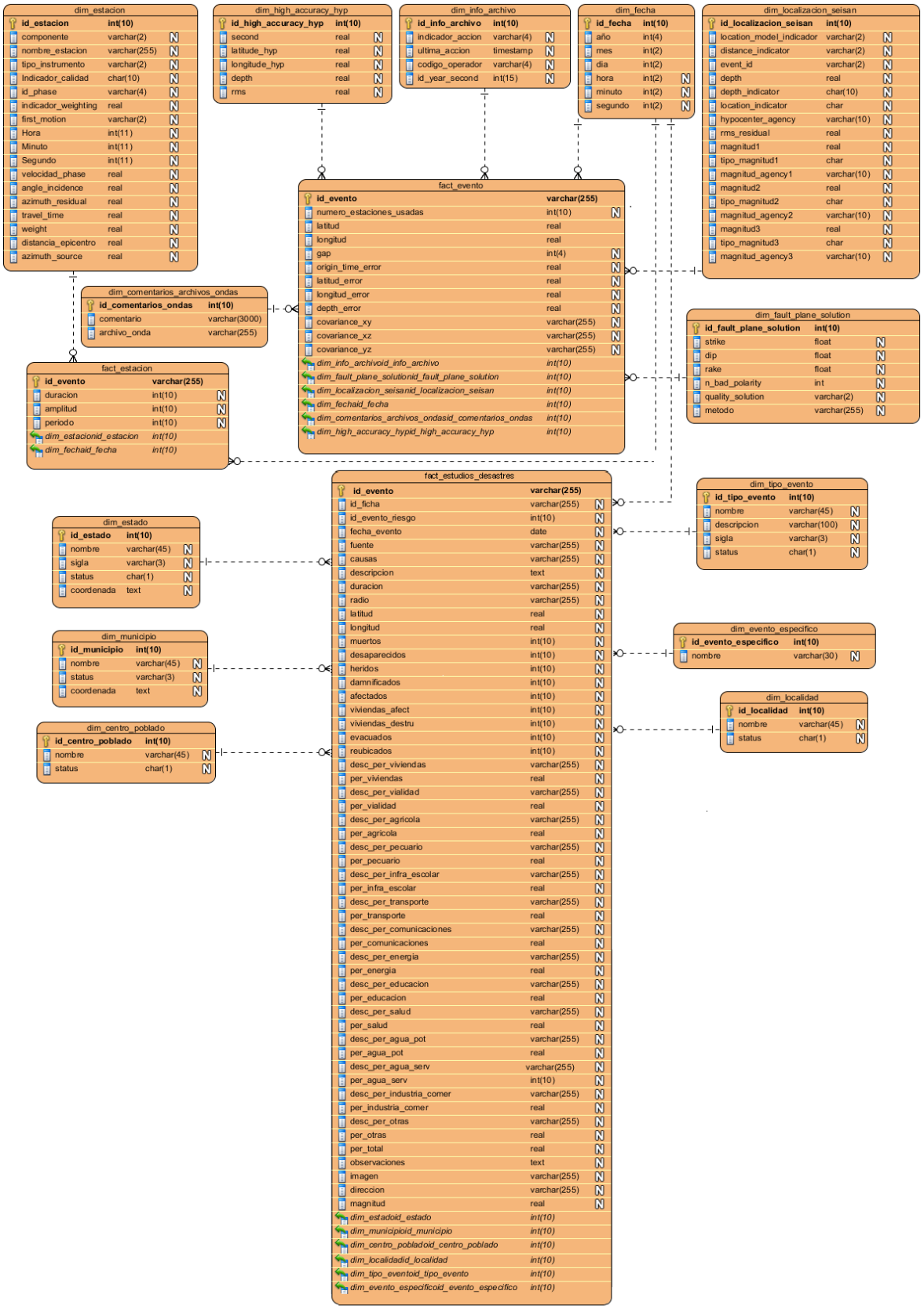


Figura 6: Estructura del data warehouse de eventos sismológicos y desastres naturales [Buono 2011]

El data warehouse se encuentra corriendo en un servidor linux con la distribución Fedora 14 y el manejador de base de datos MonetDB Versión 11.02, cuenta además con una aplicación web simple hecho en Ruby on Rails que permite consultar un número limitado de campos y un módulo de carga de datos que permite extraer, transformar y cargar los datos provenientes de los sistemas SEISAN y Estudios y desastres.

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. Permite escribir un buen código favoreciendo la convención antes que la configuración [Rails 2012].

En este concepto es importante definir la separación entre los dos entes importantes que conforman este entorno, como lo son Ruby y Rails. Ruby es un lenguaje de programación interpretado, reflexivo y orientado a objetos, creado por el programador japonés Yukihiro "Matz" Matsumoto, su implementación oficial es distribuida bajo una licencia de software libre. Según su creador, Ruby está diseñado para la productividad y la diversión del desarrollador, siguiendo los principios de una buena interfaz de usuario. Sostiene que el diseño de sistemas necesita enfatizar las necesidades humanas más que las de la máquina.

Rails es un framework para desarrollar aplicaciones web con base de datos de acuerdo con la estructura Modelo Vista Controlador (MVC). Desde el Html de la vista, a la petición y respuesta en el controlador, hasta el modelo, Rails proporciona un entorno de desarrollo en Ruby necesitando sólo una base de datos y un servidor web para mostrar contenido.

El desarrollo sobre un entorno Ruby on Rails está basado en dos filosofías, no te repitas (del inglés Don't repeat yourself, DRY) y convención sobre configuración. DRY significa que las definiciones deberían hacerse una sola vez. Dado que Ruby on Rails es un framework completo, los componentes están integrados de manera que no hace falta establecer puentes entre ellos. Mientras que convención sobre configuración significa que el programador sólo necesita definir aquella configuración que no es convencional. Ruby on Rails se ha convertido en un entorno muy poderoso para el desarrollo de aplicaciones Web y que cada día toma más auge dentro del mundo del desarrollo Web.

Es importante destacar que las características que serán descritas con mayor profundidad provienen de las versiones de Ruby 1.9.2 y Rails 3.1 por ser estas las más recientes y con las cuales será desarrollada la propuesta de la presente TEG.

Funcionamiento

Ruby on Rails funciona bajo el paradigma MVC, el cual es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos, el cual aplicado a web la vista es un página HTML, o html.erb en caso de Rails, el cual contiene el HTML y el código que provee de datos dinámicos a la página. El modelo es el Sistema de Gestión de base de datos y la Lógica de negocio, y el controlador es el responsable

de recibir los eventos de entrada desde la vista. Específicamente en Rails el modelo MVC se especifica de la siguiente manera:

- **Modelo**: En las aplicaciones web orientadas a objetos sobre bases de datos, el Modelo consiste en las clases que representan a las tablas de la base de datos. En Ruby on Rails, las clases del Modelo son gestionadas por ActiveRecord. Por lo general, la única tarea que tiene un desarrollador es hacer que los modelos hereden de la clase ActiveRecord::Base, y Rails sabrá mediante las convenciones qué tabla usar y qué columnas tiene dicha tabla.
- **Vista**: Es la lógica de visualización, es decir, cómo se muestran los datos provenientes del Controlador. Con frecuencia en las aplicaciones web la vista consiste en una cantidad mínima de código de algún lenguaje incluido en HTML. El método que se emplea en Rails por defecto es usar Ruby Embebido (archivos .rhtml, desde la versión 2.x en adelante de RoR archivos .html.erb), que son básicamente fragmentos de código HTML código en Ruby. También pueden construirse vistas en HTML y XML con Builder3 o usando el sistema de plantillas Liquid.
- **Controlador**: Las clases del Controlador responden a la interacción del usuario e invocan a la lógica de la aplicación, que a su vez manipula los datos de las clases del Modelo y muestra los resultados usando las Vistas. En las aplicaciones web basadas en MVC, los métodos del controlador son invocados por el usuario usando el navegador web. La implementación del Controlador es manejada por el ActionPack de Rails, que contiene la clase ApplicationController. Un controlador en Rails debe heredar de esta clase y definir las acciones como métodos de dicha clase.

Rails viene por defecto con una estructura de archivos definida, para facilitar las convenciones y el manejo MVC que este sostiene. El sistema de archivos que genera Rails por defecto se muestra a continuación en la figura 7, donde se define una clara separación entre archivos de aplicación (Modelos, Vistas, Controladores), archivos de configuración, pruebas, entre otros.

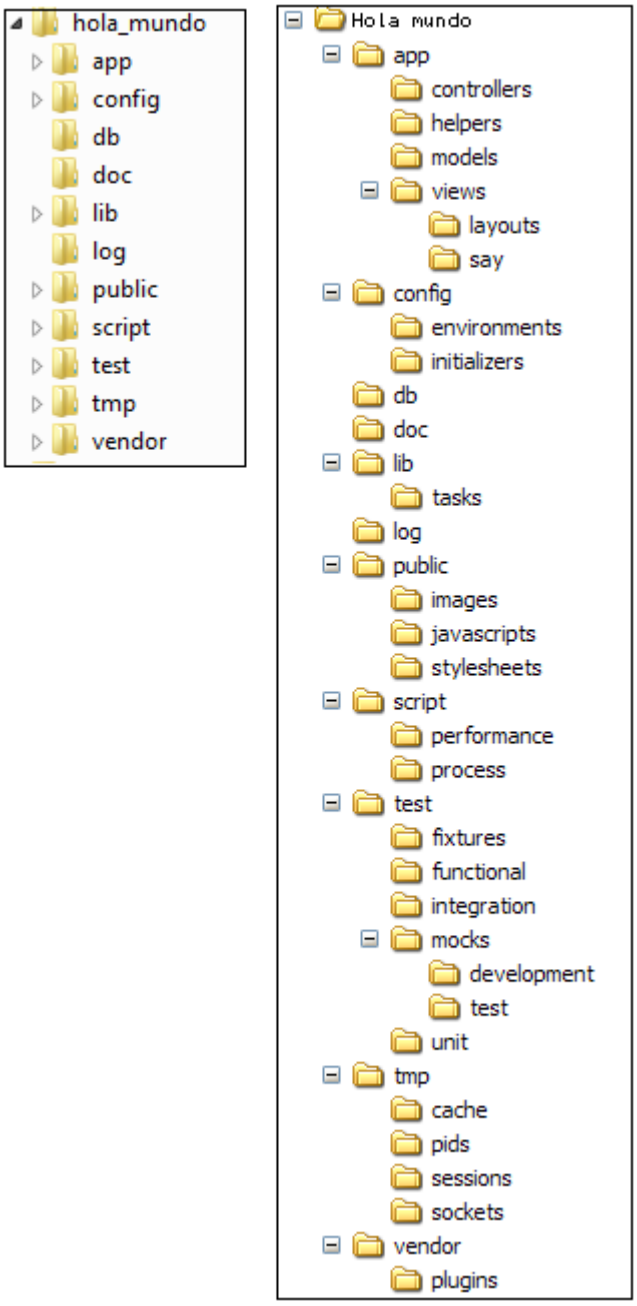


Figura 7: Estructura por defecto de una Aplicación en Ruby on Rails

Gemas

Las gemas son plugins y/o codigos añadidos a un proyecto Ruby on Rails, que proveen nuevas funcionalidades como nuevos create, nuevas funciones pre escritas (como login de usuarios) o nuevas herramientas para el desarrollo como puedan ser Haml y SASS (la primera es una nueva forma de template basada en html pero más sencilla y potente, y la segunda es igual pero para el caso de las CSS).

Para instalar una gema en ruby sólo es necesario usar el comando “gem install (nombre de la gema)” en el terminal de comandos del sistema operativo y automáticamente se inicia el proceso de descarga e instalación de las librerías a

A continuación se nombran algunas de las gemas, junto con una descripción simple, usadas para el desarrollo de este proyecto:

- **Gruff**: Es un proyecto independiente, código abierto, que aporta librerías para la creación de gráficos vectoriales en ruby. Entre sus múltiples funcionalidades, permite gráficar en varios formatos (línea, barra, pie, etc), especificar los colores de los gráficos, generar imágenes para ser insertados en una página web, y múltiples opciones de personalización de datos. En la figura 8 se puede detallar un ejemplo de una imagen generada usando esta librería.

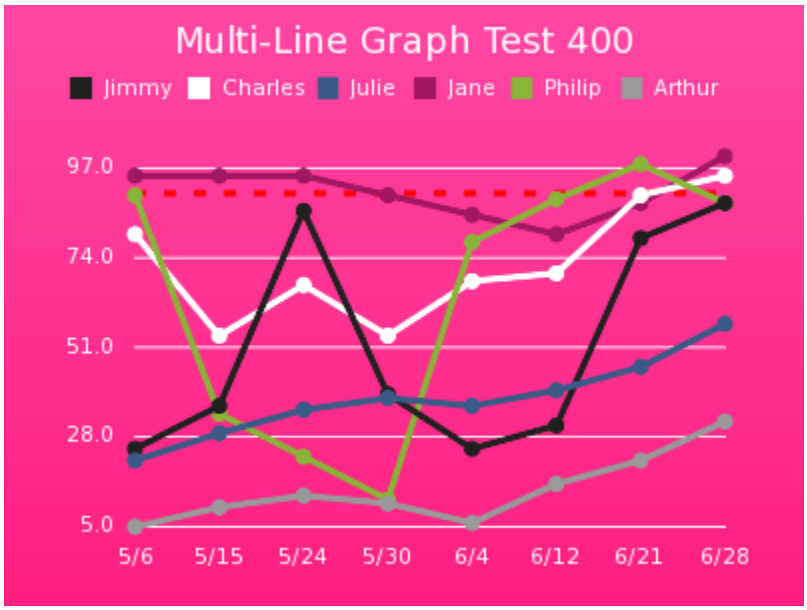


Figura 8: Ejemplo de un gráfico generado usando la gema Gruff (http://nubyonrails.com/pages/gruff)

- **Open4**: Open4 es una gema que permite abrir y ejecutar aplicaciones externas usando el propio lenguaje de ruby, útil para usar funcionalidades de programas o utilidades existentes.

- **Jquery-rails:** gema específica para el componente Rails, nos permite incorporar la librería (hecha en javascript) Jquery a cualquier aplicación hecha en rails otorgando así funcionalidades como: efectos, animaciones y opciones personalización útiles en las vistas mostradas en el navegador web.

- J**query-datatables-rails**: gema específica para el componente Rails, instala la librería Datatable el cual, junto con jquery, permite otorgar mayor esteticidad y funcionalidad a las tablas de Html.

Soporte de Bases de Datos

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

Sistemas de Información Geográficos

Se denomina información geográfica (a veces referido con el acrónimo GI) a aquellos datos espaciales georeferenciados requeridos como parte de las operaciones científicas, administrativas o legales. Dichos geodatos poseen una posición implícita (la población de una sección censal, una referencia catastral, etc.) o explícita (coordenadas obtenidas a partir de datos capturados mediante GPS, etc.).

Un Sistema de Información Geográfico o GIS del inglés Geographical Information System, es un conjunto organizado de aplicaciones (hardware, software y datos geográficos) diseñados para ser capaces de integrar, almacenar, editar, analizar, compartir y mostrar la información geográficamente referenciada de manera efectiva. Un GIS permite crear consultas interactivas, analizar la información espacial, editar datos, mapas y presentar los resultados de todas estas operaciones en tiempo real [Havskov and Ottemoller 2001].

En la actualidad los GIS se encuentran muy difundidos y empleados en investigaciones científicas, así como también en la cartografía, arqueología, urbanismo, sociología y publicidad.

Funcionamiento de un GIS

El GIS funciona como una base de datos con información geográfica (datos alfanuméricos) que se encuentra asociada por un identificador común a los objetos gráficos de un mapa digital. De esta forma, señalando un objeto se conocen sus atributos e, inversamente, preguntando por un registro de la base de datos se puede saber su localización en la cartografía.

La razón fundamental para utilizar un GIS es la gestión de información espacial. El sistema permite separar la información en diferentes capas temáticas y las almacena independientemente, permitiendo trabajar con ellas de manera rápida y sencilla, facilitando al profesional la posibilidad de relacionar la información existente a través de la topología de los objetos, con el fin de generar otra nueva que no podríamos obtener de otra forma. En la figura 9 podemos observar un ejemplo del almacenamiento de capas en un GIS de un mismo mapa.

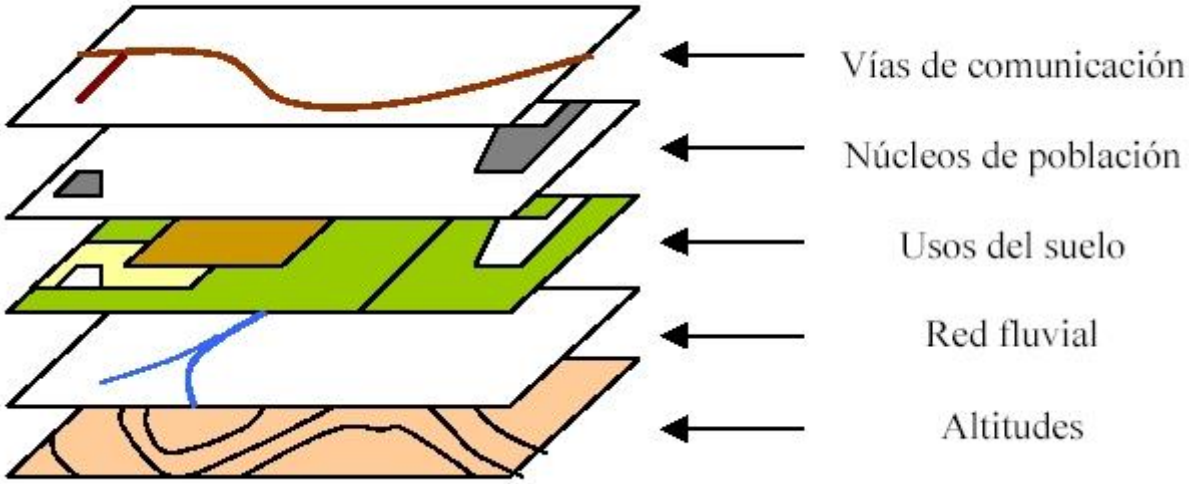


Figura 9: Estructura de capas temáticas

Las principales cuestiones que puede resolver un Sistema de Información Geográfica, ordenadas de menor a mayor complejidad, son:

- Localización: preguntar por las características de un lugar concreto.
- Condición: el cumplimiento o no de unas condiciones impuestas al sistema.
- Tendencia: comparación entre situaciones temporales o espaciales distintas de alguna característica.
- Rutas: cálculo de rutas óptimas entre dos o más puntos.
- Pautas: detección de pautas espaciales.
- Modelos: generación de modelos a partir de fenómenos o actuaciones simuladas.

Por ser tan versátiles, el campo de aplicación de los Sistemas de Información Geográfica es muy amplio, pudiendo utilizarse en la mayoría de las actividades con un componente espacial. La

profunda revolución que han provocado las nuevas tecnologías ha incidido de manera decisiva en su evolución

Representación de datos

Los datos de un GIS representan los objetos del mundo real (carreteras, el uso del suelo, altitudes). Los objetos del mundo real se pueden dividir en dos abstracciones: objetos discretos (una casa) y continuos (cantidad de lluvia caída, una elevación). Existen dos formas de almacenar los datos en un GIS: raster y vectorial.

Raster:

Un tipo de datos raster es, en esencia, cualquier tipo de imagen digital representada en mallas. El modelo de GIS raster o de retícula se centra en las propiedades del espacio más que en la precisión de la localización. Divide el espacio en celdas regulares donde cada una de ellas representa un único valor.

Cualquiera que esté familiarizado con la fotografía digital reconoce el píxel como la unidad menor de información de una imagen. Una combinación de estos píxeles creará una imagen, a distinción del uso común de gráficos vectoriales escalables que son la base del modelo vectorial. Si bien una imagen digital se refiere a la salida como una representación de la realidad, en una fotografía o el arte transferidos a la computadora, el tipo de datos raster reflejará una abstracción de la realidad. Las fotografías aéreas son una forma comúnmente utilizada de datos raster con un sólo propósito: mostrar una imagen detallada de un mapa base sobre la que se realizarán labores de digitalización. Otros conjuntos de datos raster contendrá información relativa a elevaciones (un Modelo Digital del Terreno), o de reflexión de una particular longitud de onda de la luz (las obtenidas por el satélite LandSat), etc.

Los datos raster se compone de filas y columnas de celdas, cada celda almacena un valor único. Los datos raster pueden ser imágenes (imágenes raster), con un valor de color en cada celda (o píxel). Otros valores registrados para cada celda puede ser un valor discreto, como el uso del suelo, valores continuos, como temperaturas, o un valor nulo si no se dispone de datos. Si bien una trama de celdas almacena un valor único, estas pueden ampliarse mediante el uso de las bandas del raster para representar los colores RGB (rojo, verde, azul), o una tabla extendida de atributos con una fila para cada valor único de células. La resolución del conjunto de datos raster es el ancho de la celda en unidades sobre el terreno.

Los datos raster se almacenan en diferentes formatos, desde un archivo estándar basado en la estructura de TIFF, JPEG, etc. a grandes objetos binarios (BLOB), los datos almacenados directamente en Sistema de gestión de base de datos. En un modelo raster cuanto mayores sean las dimensiones de las celdas menor es la precisión o detalle (resolución) de la representación del espacio geográfico. En la figura 10 se ilustra la pérdida de la calidad de imagen al aumentar la resolución sobre el área seleccionada.

Figura 10: Mapa Raster de Venezuela

Tabla 4: Ventajas y Desventajas de la representación Raster

Ventajas	Desventajas
La estructura de los datos es muy simple. Una imagen con todos los datos a representar.	Mayor requerimiento de memoria de almacenamiento para imágenes con resoluciones altas
Las operaciones de superposición son muy sencillas.	Las reglas topológicas son más difíciles de generar.
Formato óptimo para mostrar alta cantidad	Las representaciones gráficas son menos

de datos sin mayor consumo adicional.	interactivas.
Buen almacenamiento de imágenes digitales	Requiere volver a generar la imagen raster para cualquier variación de los datos.

Vector:

En los datos vectoriales, el interés de las representaciones se centra en la precisión de localización de los elementos geográficos sobre el espacio y donde los fenómenos a representar son discretos, es decir, de límites definidos. Cada una de estas geometrías está vinculada a una fila en una base de datos que describe sus atributos. Por ejemplo, una base de datos que describa las ciudades de un país puede contener datos sobre las calles de estos, demografía, nivel de contaminación, entre otros. Esta información puede ser utilizada para crear un mapa que describa un atributo particular contenido en la base de datos. Ejemplo de esto podría ser que las ciudades pueden tener un rango de colores en función a la densidad poblacional de las mismas. Además, las diferentes geometrías de los elementos también pueden ser comparados.

Para modelar digitalmente las entidades del mundo real se utilizan tres elementos geométricos: el punto, la línea y el polígono.

- **Puntos**: se utilizan para las entidades geográficas que mejor pueden ser expresadas por un único punto de referencia. En otras palabras: la simple ubicación. Por ejemplo, las ubicaciones de los pozos, picos de elevaciones o puntos de interés. Los puntos transmiten la menor cantidad de información de estos tipos de archivo y no son posibles las mediciones. También se pueden utilizar para representar zonas a una escala pequeña. Por ejemplo, las ciudades en un mapa del mundo estarán representadas por puntos en lugar de polígonos.

- **Líneas**: son usadas para rasgos lineales como ríos, caminos, ferrocarriles, rastros, líneas topográficas o curvas de nivel. De igual forma que en las entidades puntuales, en pequeñas escalas pueden ser utilizados para representar polígonos. En los elementos lineales puede medirse la distancia.

- **Polígonos**: se utilizan para representar elementos geográficos que cubren un área particular de la superficie de la tierra. Estas entidades pueden representar lagos, límites de parques naturales, edificios, provincias, o los usos del suelo, por ejemplo. Los polígonos transmiten la mayor cantidad de información en archivos con datos vectoriales y en ellos se pueden medir el perímetro y el área.

Los elementos vectoriales pueden crearse respetando una integridad territorial a través de la aplicación de unas normas topológicas tales como que "los polígonos no deben superponerse". Los datos vectoriales se pueden utilizar para representar variaciones continuas de fenómenos. Las líneas de contorno y las redes irregulares de triángulos se utilizan para representar la altitud u otros valores en continua evolución

En la figura 15 podemos observar un ejemplo de un mapa vectorial creado a base de puntos, líneas y polígonos. Un aspecto que se aprecia de los mapas vectoriales, es la capacidad para desplegar información geográfica sin pérdida de detalle, en este caso la ciudad de Caracas delimitada por sus principales vías y calles.

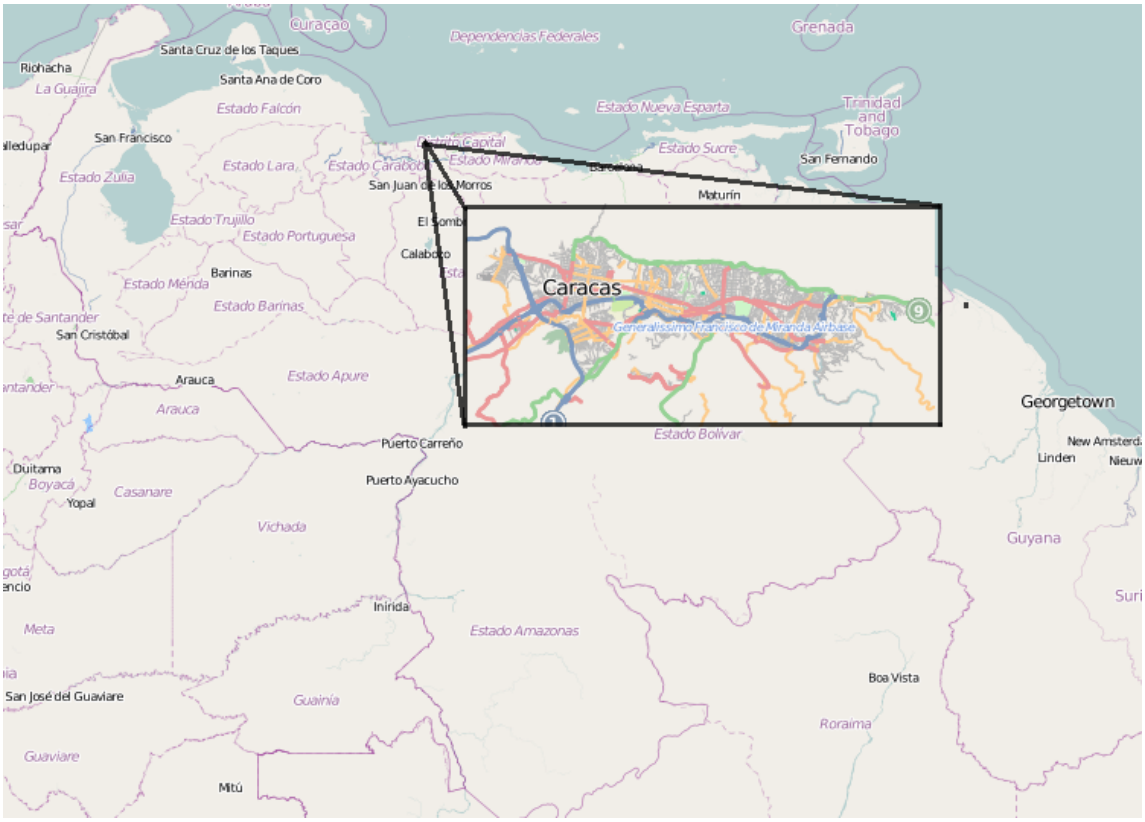


Figura 11: Mapa Vectorial de Venezuela (http://www.openstreetmap.org/)

Existen ventajas y desventajas a la hora de utilizar un modelo de datos raster o vector para representar la realidad. A continuación se presenta dos cuadros comparativos sobre las ventajas y desventajas de los datos raster y los datos vectoriales.

Tabla 5: Ventajas y Desventajas de la representación Vectorial

Ventajas	Desventajas
La estructura de los datos es compacta. Sólo Almacena los datos de los elementos digitalizados por lo que requiere menos memoria para su almacenamiento y tratamiento.	La estructura de los datos es más compleja.
Codificación eficiente de la topología y las operaciones espaciales.	Las operaciones de superposición son más difíciles de implementar y representar.
Alta calidad en la representación gráfica. Los elementos son representados como gráficos vectoriales que no pierden definición si se amplía la escala de visualización.	Eficacia reducida cuando la variación de datos es alta.
Tienen una mayor compatibilidad con entornos de bases de datos relacionales.	Es un formato más laborioso de mantener actualizado.
Las operaciones de re-escalado, reproyección son más fáciles de ejecutar.	Existe un límite en la cantidad de datos que se pueden representar. Límite dictado por la cantidad de memoria con la que se trabaje el sistema.
Los datos son fáciles de mantener y actualizar.	
Permite una mayor capacidad de análisis, sobre todo en cojunto con otras aplicaciones que permiten sobre imponer datos.	

Openlayers

OpenLayers es una librería JavaScript de código abierto que permite mostrar mapas interactivos en los navegadores web a través de una API que permite acceder a diferentes fuentes de información cartográfica en tiempo real, tales como: servicios de mapas comerciales (Ej: Google Maps, Bing, Yahoo), servidores de mapas (mapserver, arcgis), imágenes estáticas, entre otros. Adicionalmente, permite agregar datos personalizados dentro de los mapas por medio de fuentes

externas, tales como: archivos de texto, bases de datos o datos provenientes de alguna otra aplicación.

Inicialmente desarrollado por MetaCarta en el año 2006 como un proyecto código abierto con la intención de emular las capacidades de Google maps, actualmente Openlayers provee una interfaz para conectarse con una multitud de servicios y tecnologías que facilitan el desarrollo de cualquier aplicación web que involucre datos geoespaciales. Openlayers forma parte de la Open Source Geospatial Foundation, organización encargada del mantenimiento y soporte de varios proyectos de código abierto relacionados con el software geoespacial, entre los cuales están: MapServer, GeoServer, MapField, GeoTools, GRASS GIS, entre las aplicaciones más conocidas.

Funcionamiento

La librería Openlayers provee un conjunto de herramientas y funciones que permiten interpretar servicios cartográficos populares en la web, tales como: Google Maps, Bing Maps, Yahoo Maps, etc; y embeberlos en una página web sin necesidad de usar las APIs propietarias de los servicios nombrados anteriormente.

A continuación se nombran las funcionalidades proveídas por la librería de OpenLayers y usadas para este proyecto:

- Manejo de elementos por objetos
- Visualización de mapas por capas y recuadros
- Capacidad de conectarse a varios servicios de mapas: Google Maps, Bing Maps, OpenStreetMaps, etc.
- Captura de eventos por ratón y teclado
- Capacidad para sobreponer marcadores de datos sobre cualquier mapa generado
- Opciones de personalización de interfaz de usuario

Debido a que es una librería hecha en Javascript, Openlayers se ejecuta en forma de script dentro del código HTML de una página web. Esta característica permite que las mayoría de las funcionalidades se ejecuten completamente del lado del cliente, en este caso el navegador web, y se destine el servidor sólo para proveer la librería y la información que se desea mostrar.

Openlayers posee métodos para sobreponer información personalizada, ya sean estos marcadores, texto plano o formas geométricas vectoriales que se pueden incluir dinámicamente antes o después de la carga del mapa digital, permitiendo personalizar el servicio cartográfico a las necesidades de información que se requieran visualizar. Por ejemplo: usar el servicio de mapas de Google para mostrar información relacionada a los sismos ocurridos en Venezuela.

Capítulo III: Marco Metodológico

En el presente capítulo se describirá el método de desarrollo usado, siendo este una adaptación de la metodología de programación extrema o XP, en donde se expondrán sus características y aspectos más resaltantes.

Programación Extrema

La programación extrema es un método ligero, iterativo e incremental, el cual se utiliza para desarrollar software en pequeños grupos de programadores, donde la codificación es la actividad primordial sobre la documentación exhaustiva (Pressman, 2007). Mediante este método se libera rápidamente a producción un sistema sencillo y, de igual manera, se liberan continuamente nuevas versiones en periodos cortos. Tanto los jefes de proyecto, los clientes y programadores, son parte del equipo y están involucrados en el desarrollo del software.

A continuación se destacan las características principales de este método de trabajo [Anderson & Hendrickson 2000]:

- Planificación incremental.
- Programación en parejas.
- Propiedad colectiva del código.
- Comunicación constante con el cliente.
- Desarrollo guiado en pruebas.
- Continúa integración.
- Estándares de codificación.
- Refactorización de código.

Proceso de Desarrollo XP

- **Iteraciones**

  Las iteraciones simbolizan los cambios incrementales generados a través de las pruebas y retroalimentaciones repetidas, que a futuro dan como resultado un sistema estable pero en evolución. Las iteraciones pueden ser de dos tipos principalmente: por objetivo o por lapsos de tiempo.

- **Historias de Usuarios**

Las historias de usuario son un elemento primordial en el desarrollo y planificación dentro del método XP, permiten establecer un vínculo comunicacional entre el cliente y los miembros del equipo. Ayuda a priorizar y equilibrar las necesidades con la finalidad de mejorar la toma de decisiones, en cuanto a que se debe desarrollar primero. En lo que a nuestro caso se refiere, se trabajaran en función del tiempo (utilizando los días como unidad de medición) y con el formato de: un número que servirá de identificador, un nombre, el tipo (nueva o modificación/mejora), una prioridad (alta, media o baja) una estimación del tiempo y una breve descripción sobre la historia de usuario. El formato es el que a continuación se muestra:

Tabla 6: Formato de registro para una Historia de Usuario

Número: -	**Nombre:** -	
Prioridad: -	**Tipo:** -	**Tiempo Estimado:** -
Descripción: -		

- **Actores y Responsabilidades**

Los actores son todas las personas involucradas en el desarrollo del proyecto, los cuales a su vez cumplen distintos roles o responsabilidades según su importancia y nivel de participación. A continuación se destacan los roles existentes en el proceso de desarrollo:

- **Programador:** es el pilar fundamental del desarrollo en XP, tiene grandes habilidades en cuanto a la comunicación y al desarrollo en equipo. Adicionalmente, tiene la capacidad de poder abordar de forma simple y sencilla problemas complejos.
- **Cliente:** es el encargado de proveer las historias de usuario, realizar las pruebas de aceptación, requisitos funcionales y no funcionales deseables en la aplicación y la toma de decisiones acertadas sobre las características esenciales de la aplicación.
- **Probador:** su función se centra en realizar las pruebas de integración al sistema del código provisto por los programadores y de verificar el correcto funcionamiento de la aplicación. También realiza pruebas regulares y da mantenimiento siempre sustentando los resultados con informes precisos.
- **Rastreador:** se encarga de dar seguimiento al proceso general del grupo, calculando el tiempo que toman sus tareas y el progreso general a las metas que se quieren

alcanzar. Realiza estimaciones de tiempo y da la retroalimentación al equipo con el fin de mejorar el rendimiento.

Tabla 7: Esquema de actores y roles que desempeñan

	Programador	Cliente	Probador	Rastreador
Adriana Liendo		X	X	
Andrés Sanoja		X		X
Joali Moreno		X		
Mirna Freitez		X	X	
Miguel Chang	X			X

- **Planificación**

La actividad de planificación comienza creando una serie de historias de usuario que describen las funcionalidades requeridas por el cliente, proporcionando a su vez una estimación del tiempo necesario para el desarrollo.

Se utiliza un esquema, Tabla 8, al inicio de cada iteración el cual contiene el número de la iteración, una descripción, el número y nombre de las historias de usuarios a desarrollar, la fecha de comienzo de cada historia, la fecha de inicio y la fecha de fin de la iteración.

Tabla 8: Esquema de planificación de cada iteración

Iteración -			
Descripción	-		
Fecha Inicio / Fecha Fin	-		
Número	**Fecha**	**Historia**	**Tipo**
-	-	-	-

- **Diseño**

El diseño en XP sigue de forma rigurosa el principio de simplicidad, prefiriendo siempre un diseño simple respecto de una presentación más compleja. Además el diseño debe ofrecer una guía de implementación para una historia de usuario determinada.

Basados en las prácticas XP, en cada iteración de la presente etapa se realizarán prototipos mostrando las interfaces a desarrollar que permitan mejorar la comprensión de las historias planteadas.

- **Codificación**

Este método sugiere la programación en pareja, la cual consiste en que dos programadores trabajen juntos en una estación de trabajo al momento de crear el código de una historia de usuario, siguiendo en todo momento los estándares de programación, lo cual es otro aspecto de gran importancia en el método XP.

El método XP también recomienda realizar frecuentes integraciones de código entre los grupos de trabajo, de tal forma que no se produzcan problemas de compatibilidad, ni de interfaz. En este sentido se adoptará la programación individual para desarrollar el código de las principales historias de usuario y siempre trabajando en una misma iteración. Aquellas historias de usuario que no sean de mayor complejidad y no representen una funcionalidad primordial en el sistema serán desarrolladas y luego integradas a dicho sistema.

En esta etapa también se realiza la instalación y configuraciones del ambiente necesario para trabajar, además de toda la codificación de las historias de usuario de cada una de las iteraciones.

- **Pruebas**

El método XP establece realizar pruebas de todo aquello que se codifique, recomienda no dejar ninguna característica del sistema sin que haya sido probada. Dicho método propone pruebas unitarias (unit test), pruebas del programador (programmer test) y pruebas de aceptación (customer test), estas últimas son especificadas por el cliente y se enfocan en las características generales y la funcionalidad del sistema, elementos visibles y revisables por el cliente [Anderson & Hendrickson 2000].

Con respecto a este punto se realizarán pruebas del programador y pruebas de aceptación. Para las pruebas del programador se empleará una técnica simple que consistirá en evaluar parámetros de entrada seleccionado por los programadores y observar las salidas constatando que cumplan con lo esperado. Las pruebas de aceptación serán realizadas por los clientes, con el fin de comprobar que sus requerimientos hayan sido cumplidos satisfactoriamente. Además se utilizará un formato para registrar cada una de las pruebas que se realicen, presentado en la Tabla 9.

Tabla 9: Formato de registro de Prueba de Aceptación

Código	Historias de Usuario involucradas	Descripción del Caso de Prueba	Resultado Esperado	Resultado Obtenido
-	-	-	-	-

Adaptación del Proceso de Desarrollo XP

Para el presente proyecto se ha modificado la metodología de desarrollo XP, adaptándola a las limitaciones y necesidades de desarrollo presentados, entre los principales cambios presentes en la adaptación se encuentran las siguientes:

- Programación o codificación Individual.
- Iteraciones orientadas a objetivos concretos en donde los entregables de la iteración es la completación de dichos objetivos, lo cual puede o no involucrar un entregable de la aplicación web.
- Uso de prototipos y/o bocetos durante las iteraciones para mostrar los resultados que se pretenden obtener en el próximo entregable de la aplicación.
- Historias de usuarios con alcances específicos tomando como medida de tiempo base los días.

Capítulo IV: Marco Aplicativo

Este proyecto de trabajo especial de grado al estar basado en un método de desarrollo ágil, cuenta con una serie de historias de usuario distribuidas en un conjunto de iteraciones. Dichas historias están agrupadas de forma tal de alcanzar un objetivo en cada iteración, bajo unos criterios de codificación y con un grupo de casos de prueba.

El proyecto y su desarrollo, comprende un conjunto de 6 (seis) iteraciones, cada una de ellas abarcando un aspecto a desarrollar del proyecto. El periodo en el que se llevo a cabo dicho proyecto se realizó a partir de abril del 2011 hasta agosto de 2012.

En primera instancia se presentarán las historias de usuario recompiladas a lo largo del desarrollo del proyecto, seguido del detalle de cada una de las iteraciones junto con sus respectivas pruebas unitarias.

Historias de Usuario

A continuación se especifican las historias de usuario recompiladas durante el desarrollo del proyecto:

Número: 1	**Nombre:** Propuesta de Trabajo Especial de Grado	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 1 día
Descripción: Discusión del proyecto especial de grado con el profesor Andrés Sanoja, basado en proyectos de la Fundación Venezolana de Investigaciones Sismológicas , en donde se definieron los alcances del trabajo, el planteamiento del problema y el método de desarrollo a usar.		

Número: 2	**Nombre:** Definir Propuesta de Trabajo Especial de Grado	
Prioridad: Media	**Tipo:** Nuevo	**Tiempo Estimado:** 1 día
Descripción: Establecer el índice de contenido del trabajo especial de grado, especificando el contenido a desarrollar como bases teóricas, así como las páginas dedicadas a cada uno de dichos tópicos.		

Número: 3	**Nombre:** Reunión Oficial en Funvisis	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 1 día
Descripción: Discusión con la jefa del departamento de Informática de Funvisis, Adriana Liendo, sobre las características del proyecto y los requerimientos funcionales y no funcionales a desarrollar. Designación de la Ingeniera Mirna Freitez como tutora institucional.		

Número: 4	**Nombre:** Investigación de servidores de mapas y Sistemas de Información Geográfica open source	
Prioridad: Media	**Tipo:** Nuevo	**Tiempo Estimado:** 2 días
Descripción: Investigación sobre los distintos servidores que existen para la implementación de una página web con la habilidad de mostrar un mapa de Venezuela en una página web, entre algunas de ellas: Google Maps, Yahoo maps, Openstreetmaps, NearMaps y Bingmaps. Investigación sobre los distintos Sistemas de información Geográfica que existen para listar datos en una página web, entre ellas: Mapnik, Mapguide, Openlayers y Mapserver.		

Número: 5	**Nombre:** Creación de prototipos de páginas web usando las APIs de los distintos servidores de mapas investigados	
Prioridad: Media	**Tipo:** Nuevo	**Tiempo Estimado:** 1 día
Descripción: Se crearon prototipos de páginas web con los distintos servicios de mapas investigados, usando las herramientas y guias proporcionadas en las páginas de cada servicio. Todos los prototipos fueron completados con éxito.		

Número: 6	**Nombre:** Instalación y Testeo de varios Sistemas de Información Geográficos	
Prioridad: Media	**Tipo:** Nuevo	**Tiempo Estimado:** 1 día
Descarga, instalación y prueba de los sistemas de información Geográficos investigados con anterioridad.		

Número: 7	**Nombre:** Reunión con el Profesor Sanoja y la Ingeniera Adriana Liendo	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 1 dia
Descripción: Discusión con los líderes del proyectos sobre las herramientas a usar para el desarrollo de la página web conforme a los requerimientos establecidos por Funvisis.		

Número: 8	**Nombre:** Creación de un Prototipo Web del sistema conforme con los requerimientos establecidos	
Prioridad: Media	**Tipo:** Nuevo	**Tiempo Estimado:** 1 día
Descripción: Creación de una página web preliminar o prototipo de la aplicación final para la muestra y toma de decisiones entre los usuarios finales. Primera fase de la implementación que incluye lenguaje HTML y hojas de estilo.		

Número: 9	**Nombre:** Instalación de herramientas para el hosting, prueba y desarrollo de aplicaciones web	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 1 día
Descripción: Instalación y configuración Ruby 1.9.2 y Rails 3.0 bajo el ambiente windows, herramientas para el desarrollo del sistema web y de los futuros prototipos a desarrollar para la organización.		

Número: 10	**Nombre:** Implementación de un Prototipo de Web II	
Prioridad: Media	**Tipo:** Modificación	**Tiempo Estimado:** 1 día
Descripción: Creación de una página web preliminar o prototipo de la aplicación final para la muestra y toma de decisiones entre los usuarios finales. Segunda fase de la implementación en donde se incluyen varios servicios de mapas usando OpenLayers, listado y navegación dentro del mapa con información no real (eventos ficticios de sismos y siniestros)		

Número: 11	**Nombre:** Reunión con el Tesista Enrique Buono	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 1 día
Descripción: Reunión con el Tesiste Enrique Buono, encargado del desarrollo del data warehouse, para la discusión de los detalles pertinentes al proyecto. Especialmente el diseño de las tablas del data warehouse y del material reunido por sus entrevistas con la Ingenieras Ketty y Gloria, Jefes de los departamentos de Estudios y desastres y ciencias de la tierra, respectivamente.		

Número: 12	**Nombre:** Presentación del Prototipo Web	
Prioridad: Media	**Tipo:** Nuevo	**Tiempo Estimado:** 1 dia
Descripción: Presentación del prototipo más actualizado en reuniones separadas con la Ingeniera Mirna Freitez y el Profesor Sanoja para mostrar las características de la página web y consulta de opiniones sobre esta.		

Número: 13	**Nombre:** Implementación de un Prototipo de Web III	
Prioridad: Baja	**Tipo:** Modificación	**Prioridad:** Baja
Descripción: Ajustes en los parámetros de openlayers, librería usada para el despliegue de mapas dentro del prototipo web, fijando la posición central de inicio en las coordenadas en la latitud:8 y la longitud:-66, el zoom del mapa en 8, visualización de la latitud y longitud en la parte inferior derecha del mapa. Como consecuencia del cambio, el prototipo web carga y muestra el centro de venezuela al momento de cargar el prototipo.		

Número: 14	**Nombre:** Implementación de una interfaz de consultas en el prototipo web	
Prioridad: Media	**Tipo:** Nuevo	**Tiempo Estimado:** 1 día
Descripción: Implementación de una interfaz de consulta de datos usando la librería Jquery para interactuar con el usuario y extraer la información necesaria para consultar el data warehouse.		

Número: 15	**Nombre:** Instalación del sistema operativo Fedora	
Prioridad: Media	**Tipo:** Nuevo	**Tiempo Estimado:** 1 día
Descripción: Instalación del servidor de pruebas, Fedora versión 14, en una máquina virtual para la realización de las pruebas del sistema.		

Número: 16	**Nombre:** Instalación Ruby On Rails	
Prioridad: Media	**Tipo:** Nuevo	**Tiempo Estimado:** 1 día
Descripción: Instalación del leguaje Ruby 1.9.2 y del framework Rails 3.0 en la Máquina virtual Fedora.		

Número: 17	**Nombre:** Instalación de Monetdb	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 3 días
Descripción: Instalación del manejador de Bases de Datos Monetdb y creación de una copia del data warehouse dentro de la máquina virtual.		

Número: 18	**Nombre:** Actualización y mejora del Prototipo de Web	
Prioridad: Media	**Tipo:** Mejora	**Tiempo Estimado:** 1 día
Descripción: Implementación de una página web preliminar o prototipo de la aplicación final para la muestra y toma de decisiones entre los usuarios finales. Se instaló el prototipo web dentro del servidor de pruebas para verificar el funcionamiento de este bajo el ambiente Linux, adicionalmente se realizaron pruebas usando el servidor web de Rails con resultados poco satisfactorios (Imposibilidad de desplegar el mapa hecho en OpenLayers dentro del ambiente de producción de Ruby on Rails).		

Número: 19	**Nombre:** Implementación de una interfaz de consultas en el prototipo web II	
Prioridad: Media	**Tipo:** Mejora	**Tiempo Estimado:** 3 días
Descripción: Modificaciones dentro del prototipo web, específicamente agregado de varias funcionalidades de la librería Jquery, tales como: Cascade Dropdown, DateTimiPicker y cambios de estilos dinámicos.		

Número: 20	**Nombre:** Estudio de códigos en Javascript para la lectura de datos a partir del formato JSON	
Prioridad: Baja	**Tipo:** Nuevo	**Tiempo Estimado:** 1 día
Descripción: Revisión de códigos en Javascript sobre la capacidad de abrir archivos planos de texto para la creación, lectura y escritura en formato JSON.		

Número: 21	**Nombre:** Adquisición del Código fuente del data warehouse desarrollado por Enrique Buono	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 1 día
Descripción: Obtención de todo el material desarrollado por el Tesista Enrique Buono respecto al data warehouse entregado a la organización Funvisis, entre ellos: Código fuente de todos los scripts necesarios para generar el data warehouse, Modelo E-R del data warehouse, informe de seminario e informe de Tesis.		

Número: 22	**Nombre:** Estudio del modelo del data warehouse	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 5 días
Descripción: Estudio y comprensión del modelo de base de datos usado como data warehouse, creado por el Tesista Enrique Buono. Especificamente el comportamiento de las relaciones entre las distintas tablas.		

Número: 23	**Nombre:** Reunión con la Profesora Joali Moreno	
Prioridad: Baja	**Tipo:** Nuevo	**Tiempo Estimado:** 1 dia
Descripción: Reunión con la profesora Joali Moreno, responsable como tutora académica en sustitución del profesor Andrés Sanoja. Se discutieron los avances hasta la última fecha del proyecto, así como de los futuros requerimientos a implementar.		

Número: 24	**Nombre:** Estudio del modelo del data warehouse II	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 2 días
Descripción: Estudio de la metodología de consultas usada por el licenciado Enrique Buono para realizar consultas dentro del data warehouse a través del servidor Rails. Estudio de las diferencias de MonetDB con respecto a otras bases de datos. Específicamente la versión del lenguaje SQL que usa (SQL 1999).		

Número: 25	**Nombre: Reunión con la Ingenieros Gloria y Miguel Palma**	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 1 día
Descripción: Reunión con la Ingenieros Gloria y Miguel Palma (ambos miembros del departamento de ciencias de la tierra) para la toma de requerimientos sobre el visualizador web, especialmente las opciones de consulta. Además de una demostración simple del funcionamiento del sistema SEISAN.		

Número: 26	**Nombre:** Investigación de los métodos de detección de puntos existentes dentro los Sistemas de Información Geográfica	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 2 días
Descripción: Estudio e investigación de los distintos sistemas de información geográficas, con la intención de responder la pregunta: ¿Es posible buscar eventos por estados dentro de las coordenadas establecidas?. La investigación arrojo como respuesta la existencia de tal posibilidad, sin embargo es necesario el cambio de las herramientas usadas por otras que permitan tal fin.		

Número: 27	**Nombre:** Actualización del Prototipo de Web	
Prioridad: Baja	**Tipo:** Mejora	**Tiempo Estimado:** 1 día
Descripción: Incremento de la capacidad de visualización del mapa para incluir países como Cuba, Haití, entre otros. Requerimiento suministrado por la Ingeniera Gloria.		

Número: 28	**Nombre:** Cambio de Enfoque de Desarrollo de Top-Down a Bottom-Up	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado: 3** días
Descripción: Debido a la baja compatibilidad de trabajo entre las herramientas tecnológicas OpenLayers, Ruby On rails y MonetDB. Además de problemas de acoplamiento entre la base de datos y el prototipo web creado, se ha decidido invertir la metodología de desarrollo del proyecto a un enfoque “Bottom-UP” en donde las aplicaciones de nivel de usuario van acordes más con las estructuras existentes de la base de datos o el data warehouse. Este cambio de enfoque, fuerza a dejar la implementación de la interfaz visual existente, debido a que no es compatible con ruby on rails, aunque se encuentre acorde a los requerimientos del cliente.		

Número: 29	**Nombre:** Inserción de datos de prueba en el data warehouse	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 4 días
Descripción: Debido a la poca variedad de datos, se ha decidido realizar la inserción de datos de prueba para tener una mayor cantidad de registros para las consultas, así como la posibilidad de realizar bechmarking del data warehouse en un futuro usando el sistema web.		

Número: 30	**Nombre:** Creación de consultas especializadas que se usarán dentro del formulario de la interfaz web	
Prioridad: Baja	**Tipo:** Nuevo	**Tiempo Estimado: 3** Horas
Descripción: Creación de Queries para consultas especializadas que permitan mostrar los tipos de eventos disponibles, estados donde ocurrieron los eventos, entre otros. Principalmente para su uso dentro de combo box de selección en la interfaz de consulta.		

Número: 31	**Nombre:** Reunión con la Ingeniera Mirna Freitez	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado: 1 día**
Descripción: Reunión con la ingeniera Mirna Freitez, en donde se discutió el estado del proyecto y se planteó la posibilidad de hacer la interfaz escalable en un futuro, se planteó la solución de manejar la interfaz y sus objetos como clases con posibilidad de crear más clases en un futuro para nuevas funcionalidades.		

Número: 32	**Nombre:** Estudio de las capacidades de MonetDb sobre las vistas	
Prioridad: Media	**Tipo:** Nuevo	**Tiempo Estimado:** 1 día
Descripción: Estudio de la sintaxis, estructura, creación, funcionamiento y pruebas de las vistas en MonetDB.		

Número: 33	**Nombre:** Reformulación de la solución para obtener una mayor escalabilidad	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 1 día
Descripción: Estudio del modelo de clases de Ruby , del lenguaje SQL 1999 y la potencia del motor de bases de datos MonetDB, para formular una solución orientada a objetos en donde se extraen los datos a través de las vistas, procedimientos almacenados y funciones almacenadas del data warehouse y se muestren estos datos mediante una interfaz de usuario usando el modelo MVC de Rails.		

Número: 34	**Nombre:** Creación de vistas materializadas	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 4 días
Descripción: Creación de vistas materializadas que reflejan los campos que van a ser usados en la interfaz web. Entre las vistas creadas están: estados, municipios, localidades, tipos de eventos, etc. Adicionalmente, se creo una vista de todos los eventos encontrados en el data warehouse (uniendo todas las tablas existentes), para realizar fácilmente consultas sin necesidad de realizar la unión de las diferentes tablas. Se usaron los consultas formulados con anterioridad para la creación de estas vistas.		

Número: 35	**Nombre:** Estudio de las capacidades de MonetDb y el lenguaje SQL para la creación y uso de procedimientos almacenados y funciones	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 4 días
Descripción: Estudio y prueba de la sintaxis y semántica de la creación de procedimientos almacenados y funciones versión del lenguaje SQL soportado por MonetDB y su comportamiento dentro del motor de base de datos.		

Número: 36	**Nombre:** Modificación del campo Fecha de base de datos	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 3 días
Descripción: La dimensión fecha dentro del data warehouse se encuentra separado en varias columnas, cada una representando un valor distinto de una fecha: Dia, Mes, Año, Hora, Minuto y Segundo; todos guardados con el tipo de dato int (entero). Esto causa una problemática a la hora de realizar una busqueda dentro del DW al usar rango de fechas, ejemplo: between (1999-05-25 and 2011-03-25), usando un formato de fecha como datetime o timestamp esto no representaria problemas, sin embargo, debido a la implementación de los campos dia, mes y año valores enteros independientes, es necesario hacer un query distinto: ejemplo: dia between (25 and 25) and mes between (05 and 03) and año between (1999 and 2011). El ejemplo anterior causa un error de busqueda dentro de los datos, sumistrando valores entre los meses: 05,04 y 03 descartando los demás y en los días solo suministra aquellos que caigan justo en el dia 25, causando un grave problema al devolver los resultados. Para evitar este problema, se decidió crear una vista materializada con todos las columnas de la tabla fecha agregándole adicional un nuevo campo llamado 'fecha' que básicamente es la suma del año, mes y día mediante la siguiente formula: año *10000+mes *100+día; permitiendo así hacer comparaciones entre un rango numérico, ejemplo: fecha between (19990525 and 20110325). En subsiguientes queries y funciones se empezará a usar la nueva vista fecha en substitución a la tabla fecha.		

Número: 37	**Nombre:** Creación de funciones almacenadas en el data warehouse	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 2 días
Descripción: Creación de 94 funciones dentro del DW que se encargan de responder las preguntas formuladas por la Ingeniera Ketty. Un ejemplo de preguntas respondidas es cantidad de muertos por estado dado algun tipo de de evento.		

Número: 38	**Nombre:** Reunión con la Profesora Joali Moreno para discutir el problema existente de la falta de datos	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 1 dia
Descripción: Reunión con la profesora Joali Moreno, para la discusión de los avances del proyecto y el problema de la falta de datos del data warehouse, asi como la dificultad de la consulta de fechas en la tabla dim_fecha actual.		

Número: 39	**Nombre:** Reunión con las Ingenieras Mirna Freitez y Adriana Liendo	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 1 dia
Descripción: Presentanción de los avances del proyecto y discusión de problemas y cambios a realizar en el data warehouse, entre ellos: Adición del campo fecha de tipo date en la tabla Fecha, Soporte de tipos de datos geográficos, Análisis del funcionamiento del data warehouse entre otros.		

Número: 40	**Nombre:** Modificación de la tabla dim_fecha del data warehouse	
Prioridad: Alta	**Tipo:** Modificación	**Tiempo Estimado:** 1 día
Descripción: Modificación de tabla dim_fecha del data warehouse, agregando un nuevo campo llamado fecha de tipo date que posee la finalidad de permitir las comparaciones entre fechas entre los eventos del data warehouse, detalles que se obvio durante el diseño del mismo.		

Número: 41	**Nombre:** Modificación de funciones almacenadas en el data warehouse	
Prioridad: Alta	**Tipo:** Modificación	**Tiempo Estimado:** 1 día
Descripción: Modificación de las 94 funciones creadas anteriormente para el soporte del campo fecha de tipo Date en el data warehouse.		

Número: 42	**Nombre:** Investigación sobre tipos de datos geográficos de MonetDB	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 1 día
Descripción: Investigación de los tipos de datos que soporta MonetDB, específicamente los tipos de datos geográficos como: polígonos, o figuras geométricas. Del resultado de la investigación, se concluye que MonetDB posee la capacidad de manejar los tipos de datos geográficos y/o espaciales requeridos para el proyecto, mediante un parche o actualizando el manejador de bases de datos.		

Número: 43	**Nombre:** Redacción del manual de usuario	
Prioridad: Media	**Tipo:** Nuevo	**Tiempo Estimado:** 3 días
Descripción: Redacción del manual de usuario, a entregar al finalizarse el proyecto, con instrucciones para la instalación de MonetDB, detalles de uso, creación del data warehouse y programación del mismo.		

Número: 44	**Nombre:** Documentación del código	
Prioridad: Media	**Tipo:** Nuevo	**Tiempo Estimado:** 3 días
Descripción: Documentación de los scripts que crean las vistas y funciones almacenadas, detallando los parámetros que usan y explicando la funcionalidad que tienen dentro del data warehouse.		

Número: 45	**Nombre:** Estudio de la página de consulta hecha por Enrique Buono.	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 2 días
Descripción: Estudio de la aplicación de consulta web creada por el Licenciado Enrique Buono. Revisando la estructura general de la aplicación: el modelo de datos, los controladores, las vistas, el CSS, entre otros; con la finalidad de ver si existe código reusable para crear la nueva interfaz de consulta.		

Número: 46	**Nombre:** Probar el alcance del módulo de consultas del Lic. Enrique Buono.	
Prioridad: Media	**Tipo:** Nuevo	**Tiempo Estimado:** 1 días
Descripción: Realización de consultas varias para probar el alcance de la interfaz de consulta creada por Enrique Buono, entre ellas: testear la cantidad de información que es posible extraer en una consulta, la cantidad de parámetros aceptados para realizar la consulta, creación dinámica de queries, etc		

Número: 47	**Nombre:** Creación de un nuevo proyecto en Ruby on Rails como Visualizador web del data warehouse	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 1 día
Descripción: Creación de un proyecto nuevo usando el Framework Rails, el cual servirá de base para el visualizador web del data warehouse, usando la metodología MVC (Implementación nativa del Ruby On Rails). El nuevo proyecto va a tener como nombre "Tesis".		

Número: 48	**Nombre:** Configuración de la Aplicación Web usando los parámetros establecidos por la Aplicación de Enrique Buono	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 2 días
Descripción: Configuración de la nueva aplicación de consulta con los parámetros de base de datos, rutas, gemas, carpetas y archivos necesarios para que funcione de manera similar a la aplicación creada por Enrique Buono.		

Número: 49	**Nombre:** Diseño e implementación de un layout genérico	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 1 día
Descripción: Diseño y creación de un layout genérico que servirá como base para todas las páginas servidas por la aplicación, este modelo base incluye los banners y fondo de pantalla y mensajes en texto sobre la fundación.		

Número: 50	**Nombre:** Desarrollo de la página de resultados	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 2 días
Descripción: Diseño e implementación de una página para la muestra de los resultados hechos a partir de las consultas. Se decidió usar una tabla Html como estructura para mostrar los datos, por lo tanto el nombre de la página es llamado "tabla.htm.erb", en donde los datos de los distintos eventos consultados se presentarán en forma de texto y también dentro de un mapa digital Openlayers en el cual se podrán identificar la posición del evento en el mapa.		

Número: 51	**Nombre:** Implementación del controlador de las diferentes vistas de la aplicación web	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 3 días
Descripción: Programación del controlador de la aplicación web, desarrollando los distintos métodos para consultar el data warehouse, usando métodos heredados de la aplicación hecha por Enrique Buono. Refactorización de varios modelos y clases para realizar la extracción de datos desde la base de datos, específicamente usando la gema open4 para ejecutar una aplicación en Java encargada de consultar los queries y traer toda la información de la base de datos en MonetDB. Esta implementación de consultas sobre una aplicación externa en Java trae como consecuencia que no exista un vinculo directo entre Ruby On Rails y MonetDB, por lo tanto no se pueden aprovechar muchas de las virtudes del Framework Rails, como por ejemplo: la creación de clases a partir de la estructura lógica de la base de datos, obtención de datos de la base de datos usando un simple "Get", Gemas y comandos del Rails que son útiles para el diseño e implementación de la aplicación Web, entre otros.		

Número: 52	**Nombre:** Pruebas diversas sobre la vista “tabla.html”	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 1 día
Descripción: Programación de un query genérico (“Select * from funvisis.view_eventos”) para ser usado como consulta predefinida de respuesta a la página tabla.html y los resultados a mostrar sobre ella. La tabla de resultados se mostrarán directamente sobre la página html usando el tag table para que los muestre en forma de tabla. Haciendo pruebas preliminares sobre la aplicación los resultados se imprimían correctamente, pero debido a la extensiva cantidad de datos y columna, se obtenía un desbordamiento de columnas haciendo que la página fuese difícil de manipular.		

Número: 53	**Nombre:** Estudio y aplicación de la librería javascript “Datatable” buscando mejorar la usabilidad en la página “tabla.html”	
Prioridad: Baja	**Tipo:** Nuevo	**Tiempo Estimado:** 2 día
Descripción: Estudio, realización de pruebas y aplicación de la librería DataTable, esto con la finalidad de aplicarla una máscara sobre los datos de la consulta del data warehouse y mostrarse de forma más atractiva y usable para el usuario.		

Número: 54	**Nombre:** Reunión con Mirna Freitez y Adriana Liendo para discutir los avances del proyecto	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 1 día
Descripción: Reunión con las Ingenieras Adriana Liendo y Mirna Freitez en donde se discutieron los avances del proyecto, entre ellos: la página de la aplicación llamada “tabla.html”, la lentitud de los avances y la poca frecuencia de las visitas. Se acordaron reuniones los miércoles de cada semana para presentar los avances del proyecto y plantear un modelo de interfaz de consulta para la siguiente reunión.		

Número: 55	**Nombre:** Creación del modelo Evento en la aplicación de Ruby on Rails	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 2 días
Descripción: Creación de una clase en Ruby llamada “Evento.rb”, contenedora de todos los atributos asociados a los eventos Seisan y de Sistema de Estudios y Desastres. Esta clase permitirá guardar la información obtenida del data warehouse y realizar operaciones lógicas sobre los eventos, estos necesarios para presentar los resultados sobre las vistas. La generación de clases por medio de Rails queda deshabilitada, por lo tanto es necesario crear manualmente el modelo, definir el tipo de dato y realizar comprobaciones para determinar que el dato almacenado dentro de la clase corresponde a la columna del data warehouse.		

Número: 56	**Nombre:** Diseño de prototipos de consultas para la aplicación Web	
Prioridad: Media	**Tipo:** Nuevo	**Tiempo Estimado:** 1 día
Descripción: Diseño y dibujo en papel de varios prototipos de la interfaz gráfica a usar para la aplicación web, tomando en cuenta los requerimientos suministrados por Ketty y Gloria.		

Número: 57	**Nombre:** Reunión con Mirna Freitez, Adriana Liendo y Joali Moreno para discutir los prototipos de consulta y los avances del proyecto	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 1 día
Descripción: Reunión con los involucrados del proyecto en donde se discutió sobre la interfaz de consulta, los problemas asociados al proyecto y los tiempos de entrega. De la reunión se llegó a los siguientes puntos: - Definir la búsqueda de eventos por Estado cuando se consulta por SEISAN (en donde los eventos de este sistema solo se almacenan por latitud y longitud). Para buscar los eventos que pertenezcan a un Estado, estos se compararan contra las coordenadas limítrofes o los puntos más externos del Estado, si se encuentran dentro del rectángulo que forman estas coordenadas limítrofes entonces se almacenarán para una subsiguiente comprobación usando un algoritmo de búsqueda de punto sobre polígono. - Existirán al menos dos pestañas de consulta, una para Estudios y Desastres y el otro para SEISAN. La búsqueda será cruzada, es decir, los resultados de una consulta sobre el formulario SEISAN también traerán todos los eventos de Estudios y Desastres que estén relacionados y viceversa, es decir, para el caso que solo se busque por los eventos de Estudios y Desastres.		

Número: 58	**Nombre:** Desarrollo de la interfaz de consulta	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 2 días
Descripción: Desarrollo de la vista que sirve de interfaz de consulta al data warehouse. Usando los prototipos como guía, se aplicaron varias técnicas de javascript y css para obtener una ventana modal donde se presentarán las opciones de consulta. La página principal contendrá un botón que al realizar presionarse sobre ella desplegará la pantalla modal descrita anteriormente y permitirá generar la consulta.		

Número: 59	**Nombre:** Desarrollo de varias funciones para la vista de consulta	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 2 días
Descripción: Desarrollo de varias funciones que obtienen datos diversos del data warehouse (tales como los nombres de los Estados, Municipios, Parroquias y tipos de evento) que serán mostrados como opciones a elegir dentro de la vista de consultas, delimitando los parámetros del formulario a sólo los que aparezcan en el data warehouse.		

Número: 60	**Nombre:** Reunión con Mirna Freitez y Adriana Liendo para discutir los avances del proyecto	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 1 día
Descripción: Reunión con las Ingenieras Adriana Liendo y Mirna Freitez para mostrar los avances de la aplicación		

Número: 61	**Nombre:** Reunión con Joali Moreno para discutir los avances del proyecto	
Prioridad: Alta	**Tipo:** Mejora	**Tiempo Estimado:** 1 día
Descripción: Reunión con la Profesora Joali Moreno para discutir los avances del proyecto. Se sugirieron diversos cambios en la interfaz gráfica, entre ellos: Modificación del tamaño de la letras, cambios a dentro de la guía de usuarios y ajustes sobre el diseño general de la página principal.		

Número: 62	**Nombre:** Desarrollo de la interfaz de consulta II	
Prioridad: Alta	**Tipo:** Mejora	**Tiempo Estimado:** 2 días
Descripción: Modificación del aspecto gráfico de la vista de consultas usando CSS y Javascript, entre los cambios se destacan: - Eliminación de la imagen de fondo. - Modificación del tamaño y fuente de las letras usadas en la página principal. - Rellenado de los combos de selección con información traída del data warehouse. - Funciones para ocultar o mostrar distintos tipos de botones según la elecciones tomadas por el usuario en el formulario.		

Número: 63	**Nombre:** Desarrollo de una clase generadora de consultas	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 3 días
Descripción: Desarrollo de una clase generadora de queries a partir de las opciones elegidas por el usuario en la interfaz de consulta llamada “query.rb” será usada desde la clase principal del proyecto e interactuará con la clase “evento.rb” para almacenar los datos obtenidos del data warehouse como objectos de clases a mostrar para las vistas.		

Número: 64	**Nombre:** Desarrollo de la página de resultados II	
Prioridad: Alta	**Tipo:** Mejora	**Tiempo Estimado:** 1 día
Descripción: Implementación del mapa en javascript usando la librería de OpenLayers. Implementación de varias funciones dinámicas para el cargado de eventos dentro del mapa de manera dinámica. Modificación de la tabla de resultados, ajustes de css, agregado de campos de búsquedas y cambios estéticos sobre la página.		

Número: 65	**Nombre:** Reunión con Mirna Freitez y Adriana Liendo para discutir los avances del proyecto	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 1 día
Descripción: Reunión con las Ingenieras Adriana Liendo y Mirna Freitez para mostrar los avances de la aplicación. Se solicitaron gráficos de comparación para cada uno de los indicadores mostrados.		

Número: 66	**Nombre:** Desarrollo del escritor de archivos xls	
Prioridad: Media	**Tipo:** Nuevo	**Tiempo Estimado:** 2 días
Descripción: Desarrollo de una función de escritura de los eventos consultados sobre un archivo de Excel consultable por el usuario		

Número: 67	**Nombre:** Desarrollo de la página de resultados III	
Prioridad: Media	**Tipo:** Mejora	**Tiempo Estimado:** 2 días
Descripción: Implementación de barras laterales en la página de resultados, contenedoras de varios botones con diferentes funcionalidades: - Botón Retornar: permite regresar a la pantalla principal - Botón Descargar archivo de datos: permite descargar el archivo en Excel de los datos consultados		

Número: 68	**Nombre:** Implementación de un log de consultas	
Prioridad: Baja	**Tipo:** Nuevo	**Tiempo Estimado:** 1 día
Descripción: Desarrollo de un log de consultas en el cual se registran las opciones consultadas cada vez que se accede a la aplicación.		

Número: 69	**Nombre:** Desarrollo de la interfaz de consulta II	
Prioridad: Alta	**Tipo:** Mejora	**Tiempo Estimado:** 1 día
Descripción: Desarrollo de una tabla en html con el registro de las últimas 10 consultas realizadas, presentadas como enlaces que redireccionan usando un url almacenado en un log guardado en el servidor.		

Número: 70	**Nombre:** Reunión con Mirna Freitez y Adriana Liendo para discutir los avances del proyecto	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 1 día
Descripción: Reunión con las Ingenieras Adriana Liendo y Mirna Freitez para mostrar los avances de la aplicación. Se solicitaron cambios estéticos a nivel de interfaz, como el cambio de algunas imágenes, colores y palabras de la página web.		

Número: 71	**Nombre:** Estudio de varias librerías javascripts graficadores de datos	
Prioridad: Media	**Tipo:** Nuevo	**Tiempo Estimado:** 3 días
Descripción: Estudio y prueba de varias librerías hechas en javascript para el mostrado de datos de forma gráfica. Entre ellas: • Bluff • Flot • Graphael • PlotKit • Raphael • Sparklines • Gruff		

Número: 72	**Nombre:** Instalación de la gema Gruff de Ruby on Rails para la generación de gráficos del lado del servidor	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 1 día
Descripción: Instalación de la gema Gruff para Ruby on Rails usando el comando “gem install gruff”. Adicionalmente fue necesario instalar el componente generador de gráficos para Linux fedora: “yum install ImageMagick”		

Número: 73	**Nombre:** Desarrollo del graficador de indicadores	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 2 días
Descripción: Desarrollo de los generadores de gráficos dentro de la aplicación web para representar de forma gráfica varios de los indicadores de los eventos del departamento de Estudios y desastres. Las imágenes se generan del lado del servidor, evitando sobrecargar los clientes o navegadores web del lado del cliente.		

Número: 74	**Nombre:** Desarrollo de la página de resultados IV	
Prioridad: Alta	**Tipo:** Mejora	**Tiempo Estimado:** 1 día
Descripción: Implementación de los gráficos generados del lado del servidor dentro de la página de resultados.		

Número: 75	**Nombre:** Prueba de la aplicación web	
Prioridad: Media	**Tipo:** Nuevo	**Tiempo Estimado:** 1 día
Descripción: Se realizaron varias consultas, probando cada opción dentro de la interfaz de consulta. Se descubrieron problemas en las clases Java usadas por el Licenciado Enrique Buono para la extracción de los datos del data warehouse, existiendo una limitación del número de filas a retornar para cada consulta, establecido en 5.		

Número: 76	**Nombre:** Reunión con Mirna Freitez y Adriana Liendo para discutir los avances del proyecto	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 1 día
Descripción: Reunión con las Ingenieras Adriana Liendo y Mirna Freitez para mostrar los avances de la aplicación. Se pidió que existiese un desglose de los eventos por Estados, Municipios, Centros Poblados y Localidades. Se discutió sobre la problemática que existe en las clases creadas por el Licenciado Enrique Buono, específicamente la limitación de cinco elementos a extraer de la base de datos. Además se solicitó agrandar el mapa digital y unir las dos barras laterales en una sola.		

Número: 77	**Nombre:** Creación del manual de usuario	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 2 días
Descripción: Escritura del manual de consulta y acceso a la aplicación. Imágenes de la aplicación con la explicación de cada una de las funcionalidades y eventos.		

Número: 78	**Nombre:** Desarrollo de una estructura de datos para almacenar los eventos por Estado, Municipio, Localidad y centro poblado.	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 2 días
Descripción: Implementación de una lista donde se almacenará los datos de los diferentes eventos y el tipo al que pertenecen (Estado,Municipio,Localidad o Centro Poblado). Desarrollar iteradores para recorrer elementos retornados de una solicitud de consulta e identificar el tipo al que pertenecen.		

Número: 79	**Nombre:** Desarrollo de una estructura de datos para almacenar los eventos por Estado, Municipio, Localidad y centro poblado. (Segunda parte)	
Prioridad: Alta	**Tipo:** Modificación	**Tiempo Estimado:** 2 días
Descripción: Problemas con la estructura creada anteriormente impiden realizar el calculo de los indicadores por cada uno de los Estados, Municipios, Localidades o Centros Poblados. Se definió una nueva solución consistiendo en 4 estructuras separadas, una para cada grupo de estados, municipios, localidades y centros poblados. Desarrollo de la funciones encargadas del cálculo de indicadores por cada grupo.		

Número: 80	**Nombre:** Desarrollo de la página de resultados V	
Prioridad: Alta	**Tipo:** Mejora	**Tiempo Estimado:** 1 día
Descripción: Implementación de los indicadores por Estados, Municipios, Localidades y Centros Poblados. Implementación de varias funciones en javascript que permiten ocultar y mostrar un grupo de indicadores si el usuario ha hecho click sobre alguno de ellos. Se aumento el tamaño del mapa digital hecho en openlayers y modificación de varios css.		

Número: 81	**Nombre:** Implementación de la guía de usuario	
Prioridad: Baja	**Tipo:** Nuevo	**Tiempo Estimado:** 2 días
Descripción: Implementación de la guía de usuario, presentable como un panel de navegación en la barra lateral derecha. Cada elemento del panel despliega una imagen con información referente a los distintos botones y opciones de la aplicación web y su respectiva explicación.		

Número: 82	**Nombre:** Ajustes de interfaz	
Prioridad: Media	**Tipo:** Modificación	**Tiempo Estimado:** 1 día
Descripción: Diversos ajustes de interfaz usando CSS, cambio de nombre de varios títulos y modificación de varias funcionalidades que producen errores en la interfaz de usuario.		

Número: 83	**Nombre:** Entrega de aplicación	
Prioridad: Alta	**Tipo:** Nuevo	**Tiempo Estimado:** 1 día
Descripción: Entrega de la aplicación al departamento de informática de FUVISIS para la realización de pruebas y puesta en producción.		

Iteraciones

La presente tesis especial de grado se desarrolló en seis iteraciones. Cada una de estas Iteraciones representa un objetivo cumplido que incrementa la culminación del proyecto. A continuación de describen cada una de la iteraciones.

Iteración 0

- **Planificación:**

Iteración 0			
Descripción		Propuesta inicial, investigación de las tecnologías a usar y construcción del anteproyecto.	
Fecha Inicio / Fecha Fin		01/03/2012 - 06/04/2012	
Número	**Fecha**	**Historia**	**Tipo**
1	01/03/2011	Propuesta de Trabajo Especial de Grado	Nuevo
2	02/03/2011	Definir Propuesta de Trabajo Especial de Grado	Nuevo

3	12/04/2011	Reunión oficial en Funvisis	Nuevo
4	06/04/2011	Investigación de servidores de mapas y Sistemas de Información Geográfica open source	Nuevo

- **Codificación**

En la presente iteración, la etapa de codificación correspondió a la definición del proyecto a desarrollar, así como con el establecimiento de las actividades a ser realizadas para cumplir con los requerimientos definidos por la Fundación Venezolana de Investigaciones Sismológicas.

- **Pruebas**

Las pruebas realizadas en esta iteración, consistió en definir el plan de trabajo a seguir para el proyecto; esto a través de los requerimientos establecidos, con base a la información suministrada por el profesor Andrés Sanoja y la Jefa del Departamento de Informática la Ingeniera Adriana Liendo.

Iteración 1

- **Planificación:**

Iteración 1			
Descripción		Instalación del ambiente de trabajo, diseño de prototipos, obtención de requerimientos del proyecto y planteamiento del enfoque a trabajar.	
Fecha Inicio / Fecha Fin		11/04/2011 - 05/10/2011	
Número	**Fecha**	**Historia**	**Tipo**
5	11/04/2011	Creación de prototipos de páginas web usando las APIs de los distintos servidores de mapas investigados	Nuevo
6	12/04/2011	Instalación y prueba de varios Sistemas de Información Geográficos	Nuevo
7	12/05/2011	Reunión con el Profesor Sanoja y la Ingeniera Adriana Liendo	Nuevo
8	13/05/2011	Creación de un Prototipo Web del sistema conforme con los requerimientos establecidos	Nuevo
9	15/05/2011	Instalación de herramientas para el hosting, prueba y	Nuevo

		desarrollo de servicios web	
10	16/05/2011	Implementación de un Prototipo de Web II	Modificación
11	17/05/2011	Reunión con el Tesista Enrique Buono	Nuevo
12	17/05/2011	Presentación del Prototipo Web	Nuevo
13	24/05/2011	Implementación de un Prototipo de Web III	Modificación
14	26/05/2011	Implementación de una interfaz de consultas en el prototipo web	Nuevo
15	08/09/2011	Instalación del sistema operativo Fedora	Nuevo
16	10/09/2011	Instalación Ruby On Rails	Nuevo
17	10/09/2011	Instalación de Monetdb	Nuevo
18	12/09/2011	Implementación de un Prototipo de Web IV	Nuevo
19	12/09/2011	Implementación de una interfaz de consultas en el prototipo web II	Mejora
20	13/09/2011	Estudio de códigos en Javascript para la lectura de datos a partir del formato JSON	Mejora
21	15/09/2011	Adquisición del Código fuente del data warehouse desarrollado por Enrique Buono	Nuevo
22	15/09/2011	Estudio del modelo del data warehouse	Nuevo
23	15/09/2011	Reunión con la Profesora Joali Moreno	Nuevo
24	21/09/2011	Estudio del modelo del data warehouse II	Nuevo
25	31/10/2011	Reunión con la Ingenieros Gloria y Miguel Palma	Nuevo
26	01/10/2011	Investigación de los métodos de detección de puntos existentes dentro los Sistemas de Información Geográfica	Nuevo
27	03/10/2011	Implementación de un Prototipo de Web V	Mejora
28	05/10/2011	Cambio de Enfoque de Desarrollo de Top-Down a Bottom-Up	Nuevo

- **Diseño**

En esta iteración se llevo a cabo el planteamiento de una solución general del sistema, en donde se definen las herramientas a usar y el enfoque de desarrollo.

El desarrollo del proyecto inició con un enfoque Top-Down, comenzando a desarrollarse primero las vistas para el usuario, segundo la aplicación web encargada de mostrar las vistas, consultar los datos y realizar los cálculos correspondientes y por último las modificaciones del data warehouse para poder conectarse con la aplicación desarrollada.

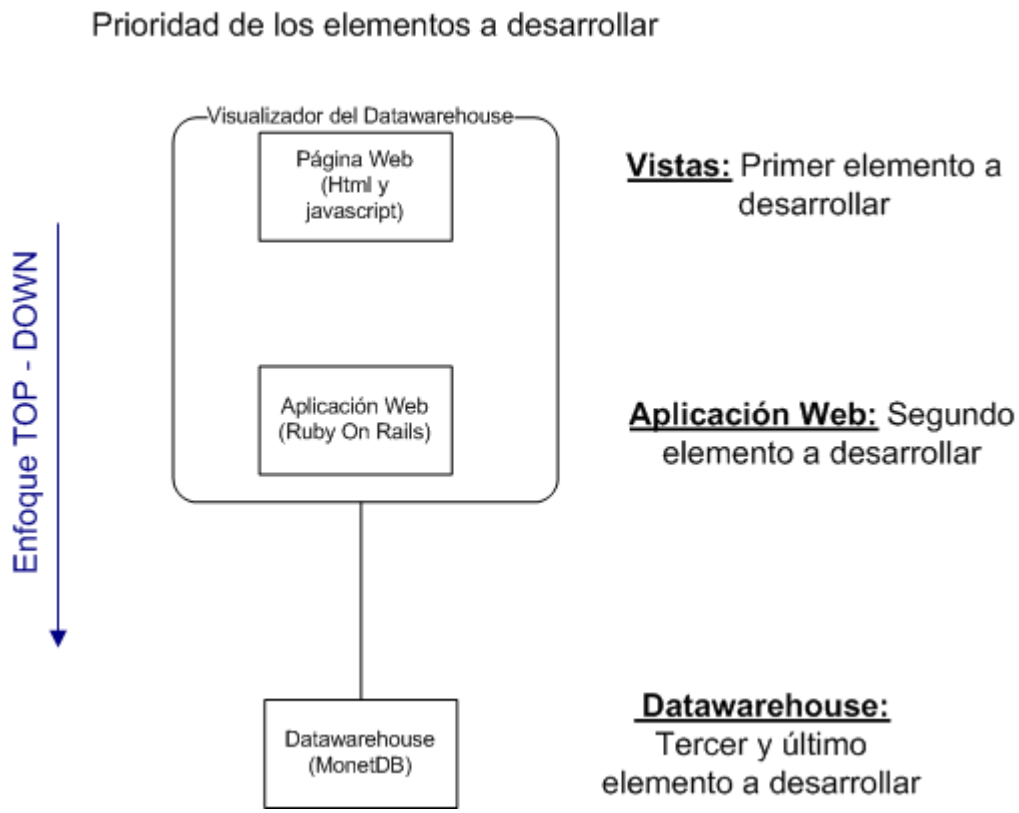


Figura 12: Enfoque de desarrollo planteado al inicio del proyecto

La principal razón de elegir este enfoque de desarrollo fue debido a la carencia del data warehouse desarrollado por Enrique Buono. Para la fecha de inicio de este proyecto, aún no se tenía ni el código de la base de datos, ni la documentanción del proyecto en su totalidad. Otra de las razones de usar el enfoque Top-Down, era el hecho de empezar el desarrollo de las vistas de la aplicación con la finalidad de realizar prototipos rápidos de muestra para los usuarios.

Se diseño un prototipo inicial con el diseño de lo que sería la página web, presentada en la figura 15.

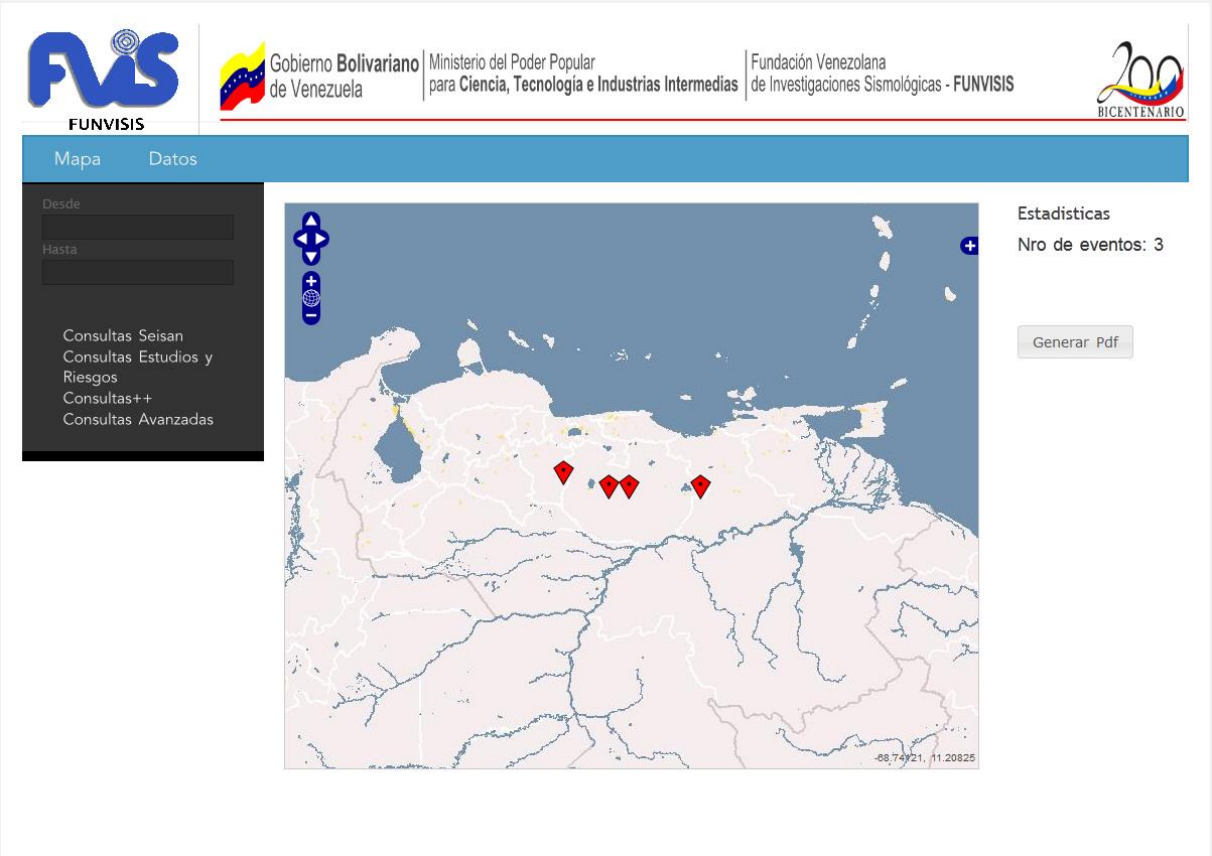


Figura 13: Diseño prototipo de la página web

Una vez recibido el código y documentación del data warehouse por parte de Enrique Buono, incluyendo entrevistas con las ingenieras Gloria y Ketty, se empezó a estudiar la solución de la base de datos y también del midleware para comenzar con la adaptación de ambos proyectos, el visualizador y el data warehouse.

Parte de la solución de Enrique Buono para conectar MonetDB con Ruby on Rails, consistió en usar una aplicación intermedia, hecha en Java, para realizar las distintas consultas dentro del data warehouse. Básicamente, una aplicación en Ruby on Rails que quiera obtener consultas de MonetDB, tiene que parar su ejecución momentáneamente y esperar/depender de la aplicación externa, mencionada anteriormente, para obtener cualquier dato del data warehouse. Esto debido a que no existe un conector natural entre MonetDB y Ruby On Rails.

Usando la solución anterior, se crea un problema en Ruby On Rails a la hora de crear cualquier aplicación. Ruby on Rails, siendo un framework de desarrollo en donde la convención está por encima de la configuración, no ofrece la misma capacidad para desarrollar sin un conector natural entre él y la base de datos; lo que trae como consecuencia que se pierdan las herramientas del framework, entre las cuales se destaca la generación de modelos, vistas y consultas automáticas a partir de la estructura de la base de datos.

Esto da la imposibilidad de seguir desarrollando la aplicación a partir de un enfoque Top-Down, ya que al no existir una abstracción proveída por el framework, es necesario entonces partir desde la base datos, en este caso el data warehouse MonetDB, y realizar las consultas necesarias o que se van a usar. A partir de allí, se debe codificar la aplicación (el controlador, el modelo y las vistas por separado) en ese mismo orden.

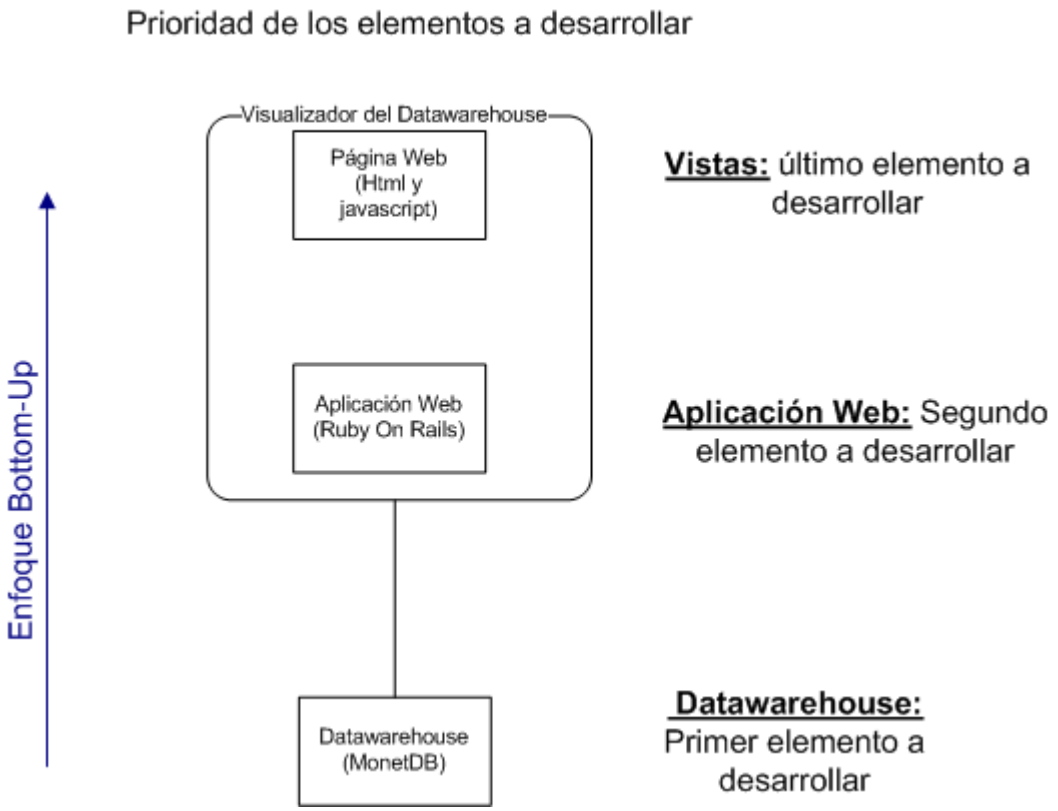


Figura 14: Nuevo Enfoque de desarrollo Bottom-up

- **Codificación**

En esta iteración se desarrollaron varios prototipos de mapas usando las tecnologías investigadas: Openlayers, Google Maps Api, Mapserver, entre otras.

Se realizó la instalación de las herramientas necesarias para el desarrollo del proyecto, entre ellas:

- Sistema Operativo Fedora 14
- Ruby 1.9.2
- Rails 3.0
- Monetdb 11.02
- Squirrel

- Scripts de tablas del data warehouse

- **Pruebas**

Código	Historias de Usuario involucradas	Descripción del Caso de Prueba	Resultado Esperado	Resultado Obtenido
1	12	Interacción con el mapa desarrollado en OpenLayers	El mapa interactivo debe responder a cualquier interacción válida con el usuario, por ejemplo: zoom, mostrar información de marcadores, cambios servicio de mapas, entre otros	Cada elemento de interacción se ejecuta exitosamente

Iteración 2

- **Planificación:**

Iteración 2			
Descripción		Desarrollo y adaptación del Datwarehouse	
Fecha Inicio / Fecha Fin		11/10/2012 - 28/02/2012	
Número	**Fecha**	**Historia**	**Tipo**
29	11/10/2012	Inserción de datos de prueba en el data warehouse	Nuevo
30	19/10/2012	Creación de queries especializadas de consulta que se usarán dentro del formulario de la interfaz web	Nuevo
31	31/10/2010	Reunión con la Ingeniera Mirna Freitez	Nuevo
32	5/11/2010	Estudio de las capacidades de MonetDb sobre las vistas	Nuevo
33	6/11/2010	Reformulación de la solución para obtener una mayor escalabilidad	Nuevo

34	11/11/2011	Creación de vistas materializadas	Nuevo
35	15/11/2011	Estudio de las capacidades de MonetDb y el lenguaje SQL para la creación y uso de procedimientos almacenados y funciones	Nuevo
36	25/11/2011	Modificación del campo Fecha de base de datos	Modificación
37	02/12/2011	Creación de funciones almacenadas en el data warehouse	Nuevo
38	05/12/2011	Reunión con la Profesora Joali Moreno para discutir el problema existente de la falta de datos	Nuevo
39	28/01/2012	Reunión con las Ingenieras Mirna Freitez y Adriana Liendo	Nuevo
40	01/02/2012	Modificación de la tabla dim_fecha del data warehouse	Modificación
41	05/02/2012	Modificación de funciones almacenadas en el data warehouse	Modificación
42	06/02/2012	Investigación sobre tipos de datos geográficos de MonetDB	Nuevo
43	12/02/2012	Redacción del manual de usuario	Nuevo
44	24/02/2012	Documentación del código	Nuevo

- **Diseño**

Durante esta iteración se desarrollaron todas las consultas concernientes a adaptar el data warehouse, con la finalidad de que pueda ser usado de manera sencilla. Esto evita recurrir a consultas complejas, construidos en la aplicación web en tiempo de ejecución.

El data warehouse posee diferentes dimensiones que requieren consultas complejas para extraer los datos. No existe método o función que permite consultar un evento de manera directa y sencilla. Por lo tanto, fue necesario construir una consulta dinámica de los datos sin tener que armar una consulta específica por cada dimensión.

Para ello fue necesario construir una consulta de la cual poder obtener la información de todas las tablas comunes a un evento, ya sea que el evento de SEISAN o de Estudios y desastres. La idea es unir mediante una consulta varias de las dimensiones en una vista o función almacenada que pueda ser utilizada en el futuro. En las siguientes

figuras se puede apreciar gráficamente como sería la unión de las tablas a la hora de realizar una consulta.

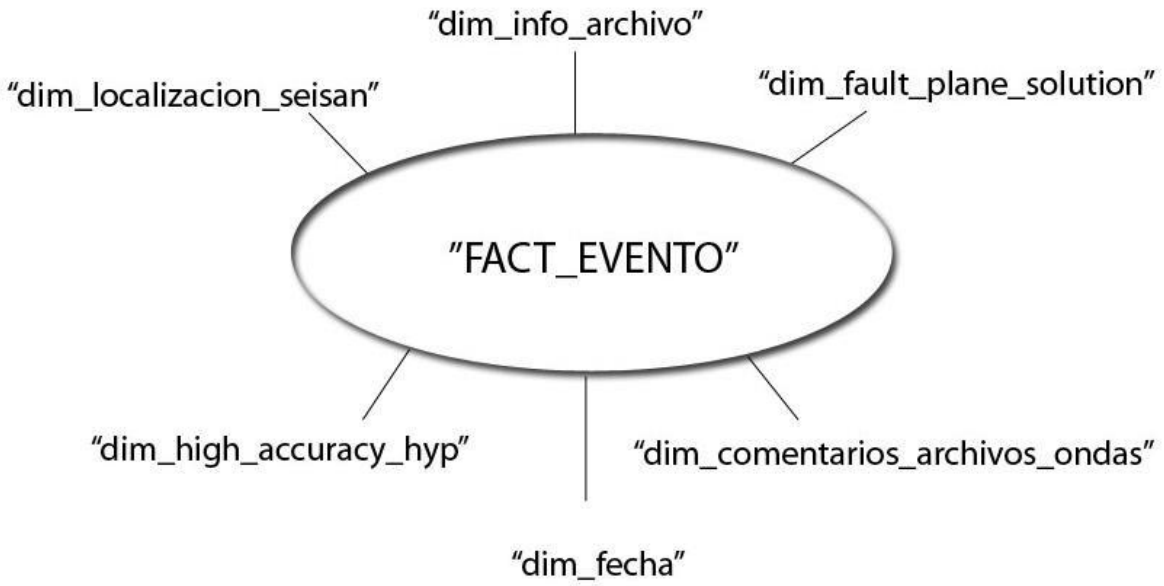


Figura 15: Relación de la tabla evento SEISAN dentro del data warehouse

Figura 16: Relación de la tabla estudios y desastres dentro del data warehouse

Durante el desarrollo de esta iteración fue necesario rediseñar una de las tablas de la base de datos, en concreto, la tabla "dim_fecha" usada tanto para la consulta en SEISAN como para consultas en Estudios y Desastres. Inicialmente la tabla fecha almacenaba las días, meses y años de los eventos en campos distintos, generando como consecuencia una imposibilidad de consultar los eventos de manera directa. Fue necesario agregar un

nuevo campo fecha de tipo datetime al esquema del data warehouse actual, para poder facilitar las consultas sobre rangos de fechas.

Figura 17: Modificación de la tabla fecha (antes y después)

- **Codificación**

  Durante esta iteración se desarrollaron cada uno de las consultas a usarse en la aplicación web. La consulta que se encargará de unir las diversas tablas fue almacenada dentro del data warehouse en forma de vista, llamada " view_eventos", esto para facilitar la utilización del data warehouse en otras aplicaciones.

  Se crearon 92 funciones, también almacenadas dentro del data warehouse, que responden preguntas simples de la base de datos de Estudios y Desastres. También se crearon vistas específicas que tienen la finalidad de devolver campos del data warehouse útiles para la aplicación web, entre ellas: Estados existentes en el data warehouse, tipos de eventos existentes, entre otros. Además se desarrollo el manual del data warehouse, donde se explica la unión de las tablas de manera detallada y creación de varias de las funciones.

Iteración 3

- **Planificación:**

Iteración 3			
Descripción		Creación de la aplicación Web en Ruby On Rails y desarrollo del modelo de datos a usar.	
Fecha Inicio / Fecha Fin		09/03/2012 - 11/05/2012	
Número	**Fecha**	**Historia**	**Tipo**
45	09/03/2012	Estudio de la página de consulta hecha por Enrique	Nuevo

		Buono.	
46	16/03/2012	Probar el alcance del módulo de consultas del Lic. Enrique Buono.	Nuevo
47	18/03/2012	Creación de un nuevo proyecto en Ruby on Rails como Visualizador web del data warehouse	Nuevo
48	18/03/2012	Configuración de la Aplicación Web usando los parámetros establecidos por la Aplicación de Enrique Buono	Nuevo
49	28/03/2012	Diseño e implementación de un layout genérico	Nuevo
50	29/03/2012	Desarrollo de la página de resultados	Nuevo
51	02/04/2012	Implementación del controlador de las diferentes vistas de la aplicación web	Nuevo
52	08/04/2012	Pruebas diversas sobre la vista “tabla.html”	Nuevo
53	14/04/2012	Estudio y aplicación de la librería javascript “Datatable” buscando una representación más estética en la página “tabla.html”	Nuevo
54	02/05/2012	Reunión con Mirna Freitez y Adriana Liendo para discutir los avances del proyecto	Nuevo
55	04/05/2012	Creación del modelo Evento en la aplicación de RoR	Nuevo
56	05/05/2012	Diseño de prototipos de consultas para la aplicación Web	Nuevo
57	11/05/2012	Reunión con Mirna Freitez, Adriana Liendo y Joali Moreno para discutir los prototipos de consulta y los avances del proyecto	Nuevo

- **Diseño**

  En esta iteración se diseño una interfaz simple de usuario para mostrar los datos obtenidos del data warehouse.

Figura 18: Interfaz que muestra los datos procedentes del data warehous

Se realizaron varios bocetos en papel de la interfaz de consultar para la página web. A continuación se muestran varios de ellos:

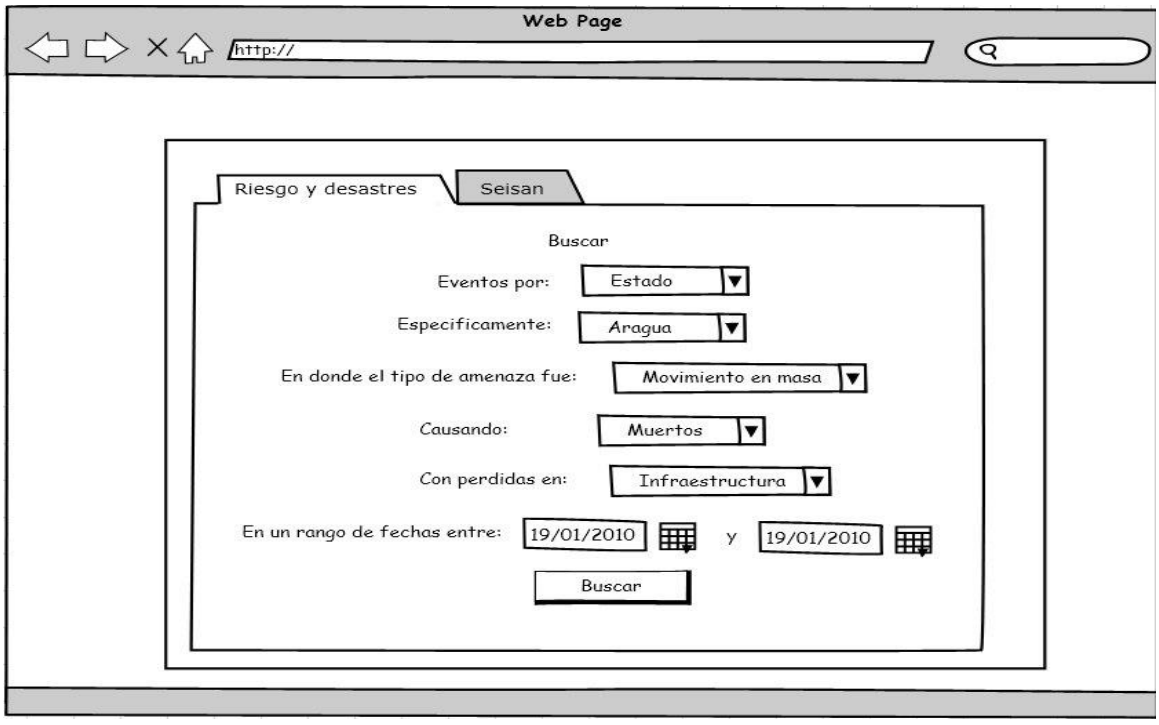


Figura 19: Boceto de interfaz de consulta para datos de estudios y desastres

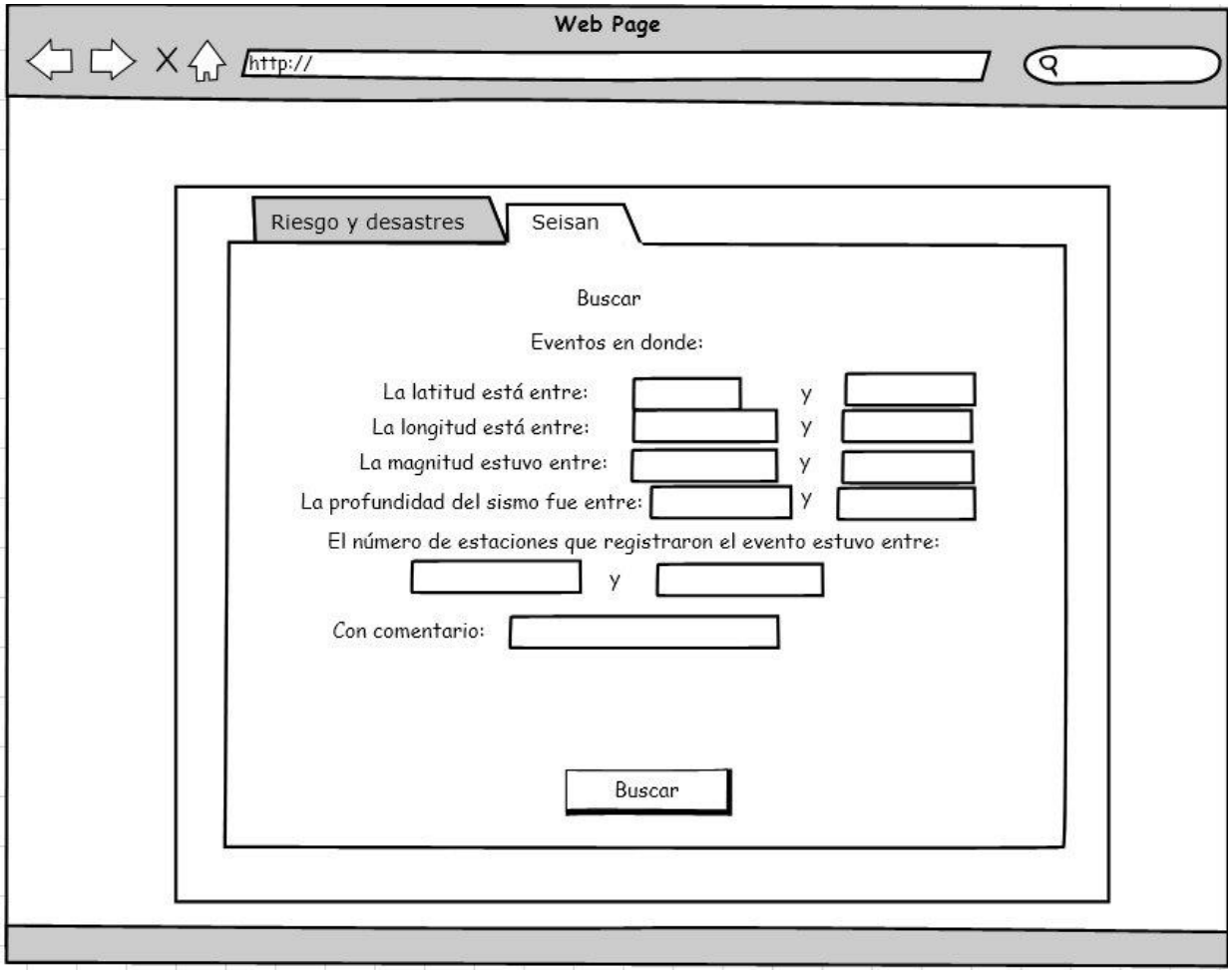


Figura 20: Boceto de interfaz de consulta para datos de SEISAN

- **Codificación**

  En esta iteración se creó la aplicación web, se desarrollaron parte de los controladores y varias vistas simples para poder visualizar los datos.

- **Pruebas**

Código	Historias de Usuario involucradas	Descripción del Caso de Prueba	Resultado Esperado	Resultado Obtenido
2	57	Mostrar datos del data warehouse	Los datos se reflejen en la vista	Los datos fueron imprimidos exitosamente en la pantalla

Iteración 4

- **Planificación:**

Iteración 4			
Descripción		Desarrollo de la interfaz gráfica de la página de consulta	
Fecha Inicio / Fecha Fin		13/05/2012 – 25/06/2012	
Número	**Fecha**	**Historia**	**Tipo**
58	13/05/2012	Desarrollo de la interfaz de consulta	Nuevo
59	14/05/2012	Desarrollo de varias funciones para la vista de consulta	Nuevo
60	16/05/2012	Reunión con Mirna Freitez y Adriana Liendo para discutir los avances del proyecto	Nuevo
61	19/05/2012	Reunión con Joali Moreno para discutir los avances del proyecto	Nuevo
62	18/05/2012	Desarrollo de la interfaz de consulta II	Mejora
63	21/05/2012	Desarrollo de una clase generadora de consultas	Nuevo
64	22/05/2012	Desarrollo de la página de resultados II	Mejora
65	23/05/2012	Reunión con Mirna Freitez y Adriana Liendo para discutir los avances del proyecto	Nuevo
66	25/05/2012	Desarrollo del escritor de archivos xls	Nuevo
67	27/05/2012	Desarrollo de la página de resultados III	Mejora
68	30/05/2012	Implementación de un log de consultas	Nuevo
69	05/06/2012	Desarrollo de la interfaz de consulta II	Mejora
70	06/06/2012	Reunión con Mirna Freitez y Adriana Liendo para discutir los avances del proyecto	Nuevo
71	10/06/2012	Estudio de varias librerías javascripts graficadores de datos	Nuevo
72	11/06/2012	Instalación de la gema Gruff de RoR para la generación de gráficos del lado del servidor	Nuevo

73	15/06/2012	Desarrollo del graficador de indicadores	Nuevo
74	18/06/2012	Desarrollo de la página de resultados IV	Mejora
75	25/06/2012	Pruebas de la aplicación web	Nuevo

- **Diseño**

En esta iteración fue necesario diseñar el modelo de datos que soporta la aplicación, por lo general Rails puede generarlo importándolo de la base de datos, sin embargo debido a la ausencia del conector entre Ruby on Rails y MonetDB este paso es imposible de realizar.

Se desarrolló un modelo de datos en donde un evento es una clase genérica capaz de retener todos los campos, tanto de SEISAN como de Estudios y Desastres. Durante esta iteración se crearon y refinaron las interfaces de consulta y de resultados. A continuación, se muestran los diseños de la página de consultas y de la página de resultados.

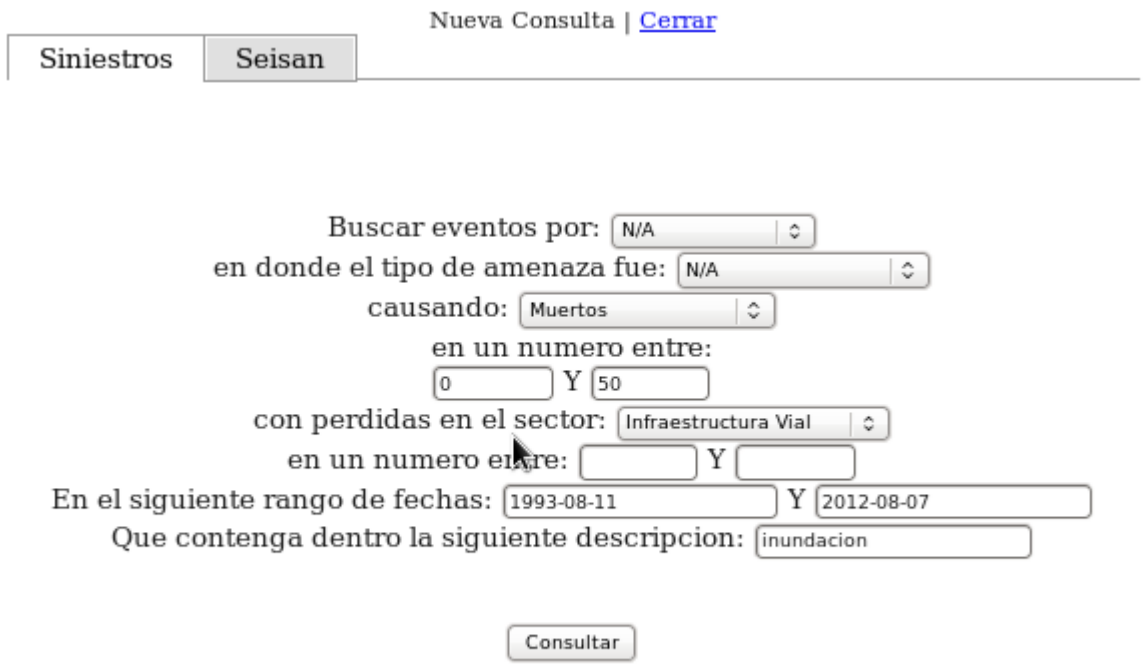


Figura 21: Interfaz de consulta de la aplicación web

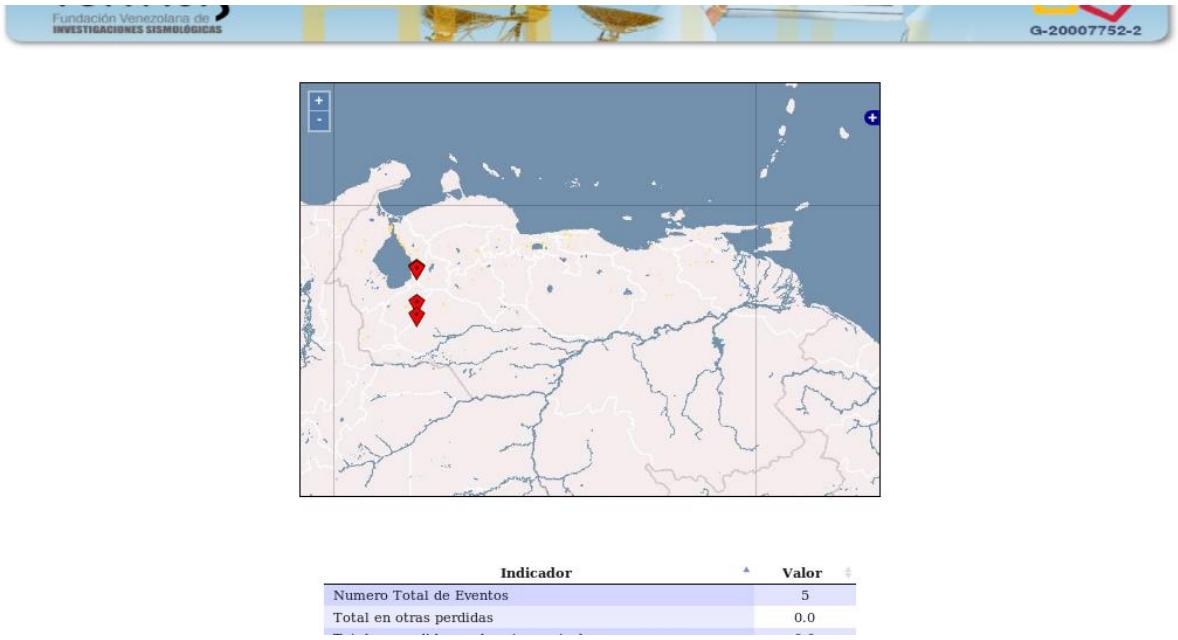


Figura 22: Interfaz de resultados de la aplicación web

- **Codificación**

Gran parte de la codificación de esta iteración consistió en el modelado de un "*evento",* como una clase capaz de ser usada para almacenar un registro retornado del data warehouse. Debido a la ausencia del conector entre Ruby on Rails y MonetDB, cada cadena de resultados retornada por la aplicación Java tuvo que ser procesada, en este caso: separar la cadena por saltos de línea y contar cada posición de la cadena para determinar que dato es y almacenarlo en una lista de tipo "*evento".*

Adicionalmente, se modificaron los controladores de la aplicación, agregándoles el procesamiento de los eventos para poder obtener resultados y desplegarlos como indicadores en las vistas.

Se desarrolló un módulo de graficación de indicadores en donde se muestran los resultados de manera gráfica y otro módulo encargado de generar un archivo xls con todos los eventos de una consulta realizada.

Finalmente, se agregaron varias de las librerías javascript y se mejoraron las vistas, agregando el mapa interactivo en OpenLayers, marcadores de datos obtenidos de la lista de eventos y variados efectos estéticos sobre la interfaz.

- **Pruebas**

Código	Historias de Usuario involucradas	Descripción del Caso de Prueba	Resultado Esperado	Resultado Obtenido
3	60	Consultas mediante la interfaz de consultas de la aplicación	Permita acceder a la interfaz de resultados con la consulta buscada por el usuario	Despliegue de la interfaz de resultados de la consulta hecha
4	60	Visualización de los indicadores	Los resultados de los indicadores se muestran en la interfaz	Cada uno de los indicadores se muestra y despliega el resultado correcto
5	64	Eventos desplegados en el mapa interactivo	Se deben mostrar cada uno de los eventos en el mapa OpenLayers	Visualización de los 5 eventos almacenados en el data warehouse
6	69	Despliegue de la tabla de indicadores	Los indicadores de deben mostrar bajo un nuevo formato más visual para el usuario	La tabla de indicadores se despliega correctamente

Iteración 5

- **Planificación:**

Iteración 5			
Descripción		Creación de la guía para el usuario y entrega de la aplicación.	
Fecha Inicio / Fecha Fin		26/06/2012 - 20/09/2012	
Número	**Fecha**	**Historia**	**Tipo**
76	26/06/2012	Reunión con Mirna Freitez y Adriana Liendo para discutir los avances del proyecto	Nuevo
77	28/06/2012	Creación del manual de usuario	Nuevo

78	05/07/2012	Desarrollo de una estructura de datos para almacenar los eventos por estado, municipio, localidad y centro poblado.	Nuevo
79	08/07/2012	Desarrollo de una estructura de datos para almacenar los eventos por estado, municipio, localidad y centro poblado. (Segunda parte)	Mejora
80	12/07/2012	Desarrollo de la página de resultados V	Nuevo
81	10/08/2012	Implementación de la guía de usuario	Nuevo
82	19/09/2012	Ajustes de interfaz	Modificación
83	20/09/2012	Entrega de aplicación	Nuevo

- **Diseño**

  En esta iteración se incluyeron varios ajustes y modificaciones a la interfaz de usuario, entre ellas, el despliegue de los indicadores en forma de gráfica, barras laterales para acceso a las guías de usuario, agregado de botones para acceder a los indicadores por diferentes renglones y finalmente accesos directos a consultas pasadas estilo historial.

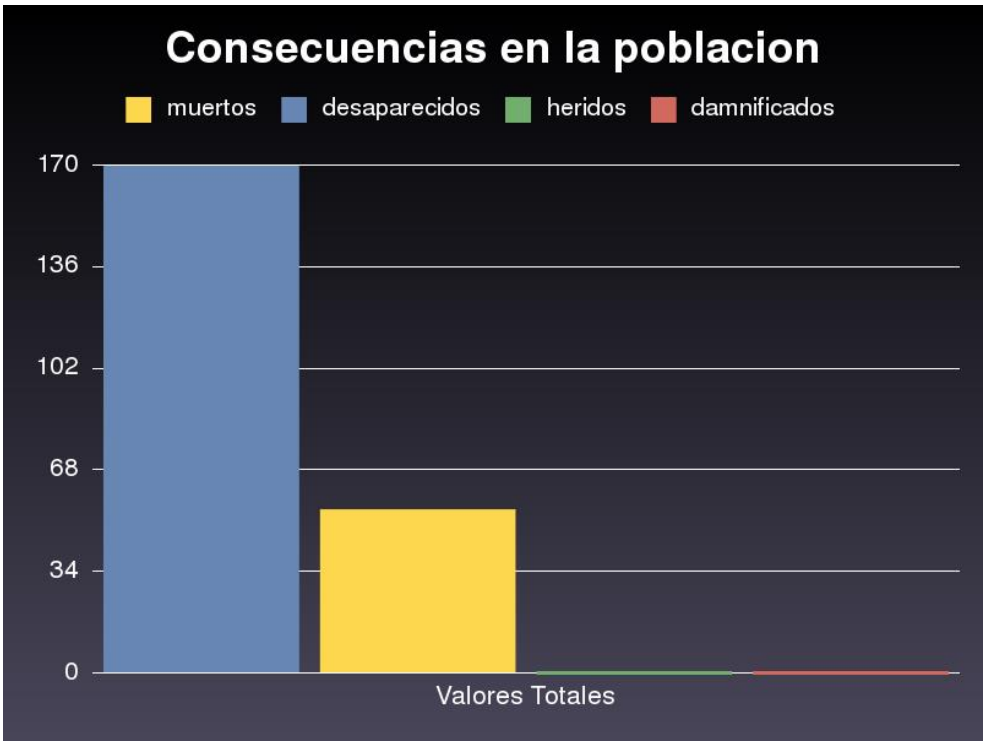


Figura 23: Gráfico de indicadores como se muestra en la interfaz de resultados

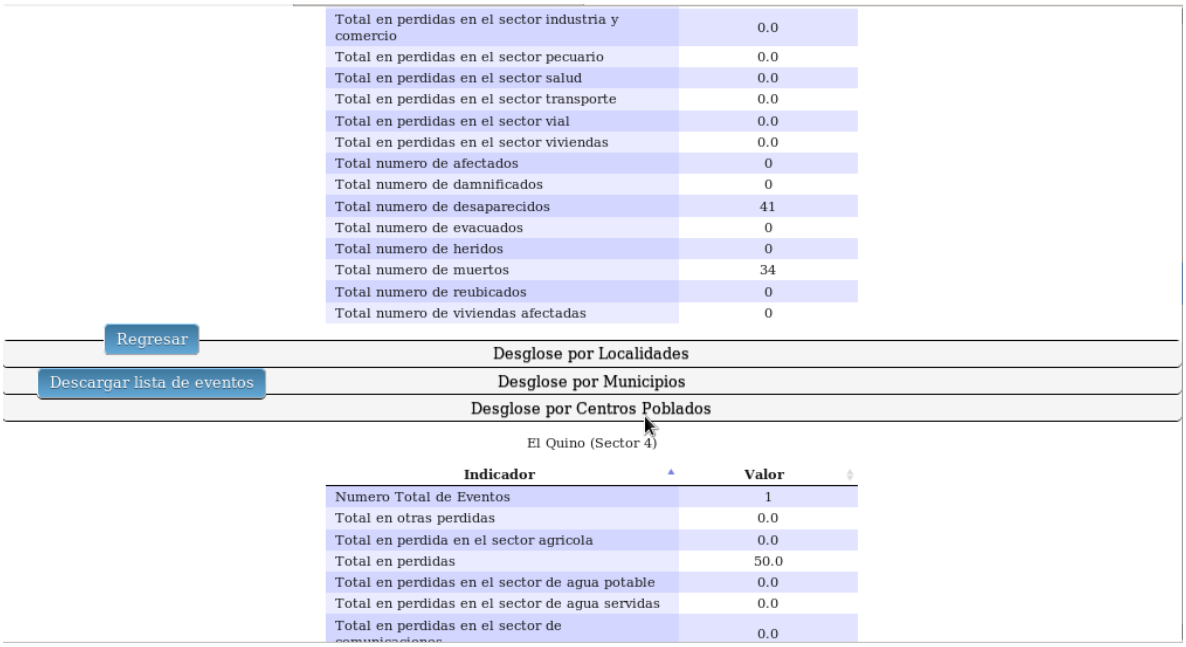


Figura 24: Barras laterales y botones de desglose de indicadores

- **Codificación**

  En esta iteración se desarrolló el cálculo de los indicadores por Estado, Municipio, Centro Poblado y Localidades, siendo necesario crear varias listas dinámicas en donde almacenar dichos datos y exponerlos mediante métodos hacia las vistas donde deben ser mostradas.

  Por otra partse se crearon las guías de usuario y se agregaron dentro de la aplicación.

- **Pruebas**

Código	Historias de Usuario involucradas	Descripción del Caso de Prueba	Resultado Esperado	Resultado Obtenido
7	75	Visualización de los indicadores gráficos	Gráficos de indicadores desplegados correctamente	Todos los gráficos se despliegan correctamente
8	75	Descarga del archivo XLS con los datos de los eventos	Archivo en Excel de los datos descargado de la aplicación	Archivo descargado y con todos los datos de la consulta
9	82	Despliegue de las imágenes que sirvn de guía para el usuario	Las imágenes que sirven de guías deben desplegarse correctamente	Imágenes de las guías de usuario desplegadas con el paso del ratón

Conclusiones

En el presente trabajo especial de grado se desarrollo una aplicación para visualizar los distintos a partir de eventos registrados en el data warehouse de la Fundación Venezolana de Investigaciones Sismológicas (Funvisis). Dicho data warehouse fue revisado y estudiado detalladamente para, en una primera instancia, entender su funcionamiento y permitir, a través del conocimiento adquirido, actualizar y/o modificar sus componentes para necesidades futuras en el visualizador.

Paralelamente se investigaron y probaron diferentes tecnologías web que permitieron crear visualizadores de datos en mapas dinámicos. Esta información fue de gran utilidad en la creación de las interfaces gráficas para el presente proyecto, brindado un mayor control sobre la data que se despliega al usuario.

Siguiendo los requerimientos, establecidos inicialmente en el proyecto, se ha logrado desarrollar exitosamente una aplicación web para Funvisis que puede ser usada desde cualquier computadora de la Fundación, dentro de la red interna a través de un navegador web. Ahora los datos del data warehouse se visualizan de una manera gráfica mediante indicadores y un mapa digital en el que se muestran los eventos tanto del sistema SEISAN como del sistema de estudios y desastres. La aplicación no sólo permite acceder a la data del data warehouse sino que también representa un punto de partida, para futuras necesidades y/o requerimientos dentro de la fundación que involucren aspectos como: la interfaz de visualización de datos usando gráficos, las herramientas GIS similares a OpenLayers y/o el mismo data warehouse.

El proceso de desarrollo de este proyecto presento dificultades y retos desde un principio, que se resolvieron paulatinamente, por ejemplo se destaca el hecho en la relación de las herramientas Ruby on Rails y MonetDB, los cuales han necesitado de conectores especiales y un desarrollo muy específico para unir ambas herramientas, brindando conocimiento al programador de ambas herramientas y un mejor entendimiento del proceso de programación extrema.

Por último solo queda destacar que la aplicación se entrega completamente funcional, junto con todas las herramientas de desarrollo, manuales de usuario y consultas hechas para el data warehouse, ofreciendo una cantidad de conocimiento y apoyo técnico los cuales están a disposición de los próximos desarrolladores que se involucren en el proyecto.

Referencias Bibliográficas

[Funvisis 2013] Funvisis, *Página principal de FUNVISIS* . En línea disponible en: http://www.funvisis.gob.ve/index.php Fecha de consulta: (20/09/2012)

[Havskov and Ottemoller 2001] Havskov and Ottemoller, *SeisAn Earthquake analysis software*. En línea disponible en: http://www.geoinstr.com/pub/manuals/seisan_7.2.pdf. Fecha de consulta (20/09/2012)

[Inmon 1995] Bill Inmon*, ¿What is a Datawarehouse?.* 1995

[Bouman and Dongen 2009] Bouman, R., & Dongen, J. v., *Pentaho Solutions Business Intelligence and Data Warehousing with Pentaho and MySQL*. Indianapolis: Wiley Publishing, Inc. 2009

[MonetDB 2012] MonetDB, *The column-store pioneer | MonetDB*. En línea disponible en: www.**monetdb**.org/Home. Fecha de consulta (20/09/2012)

[Rails 2012] *Web Developmente that doesn´t hurt* . En línea. Disponible en: http://rubyonrails.org/. Fecha de consulta (20/09/2012)

[Buono 2011] Enrique Buono, *Desarrollo de un Repositorio de Datos para la Fundación Venezolana de Investigaciones Sismológicas.* Modelado del data warehouse. 2011

[Anderson & Hendrickson 2000] Anderson, J., & Hendrickson, *Extreme Programming Installed.* Editorial Addison-Wesley Longman Publishing Co. Año de publicación: 2000

Burbeck, S. (1992). *Model-View-Controller Architecture.* Recuperado el 10 de Diciembre de 2010, de Model-View-Controller Architecture: http://st-ww.cs.illinois.edu/users/smarch/st-docs/mvc.html

Pressman, R. (2007). Ingeniería del Software Un enfoque práctico, Sexta Edición. Mc Graw Hill.

Victor Velar, Desarrollo de aplicaciones geográficas web. OpenLayershttp://victorvelarde.files.wordpress.com/2011/01/curso-openlayers-victorvelarde.pdf. Fecha de consulta (20/09/2012)

Mapserver , *Open Source Web Mapping*. http://mapserver.org/. Fecha de consulta (20/09/2012)