Este capítulo presenta brevemente la manera en la que se organizo y la manera en la que se implementaron las clases para la transformación, visualización y manipulación de los documentos GML versión 1.0 y SVG.

6.1 Interfaces y servicios de Mappa

Debido al requerimiento que indica que el sistema debe trabajar vinculado con la Internet, la implementación de Mappa se visualizó, a falta de otra palabra, como un portal que integrara las herramientas necesarias para el eficaz desarrollo de los servicios heredados del análisis de requerimientos.

La interfaz principal de Mappa se considero como un marco doble, en la parte superior se visualizará el menú con las secciones disponibles y en la parte inferior se visualizará el contenido de las secciones (*véase A5.1*).

Las secciones disponibles son:

- Pp
  Liga a la página de presentación del proyecto.
- Tesis
  Liga al documento escrito del proyecto.
- Manejo de archivos
  Liga hacia las herramientas pertenecientes a los servicios de almacenamiento y administración de catálogos.
- Conversión GML
  Liga hacia la herramienta perteneciente al servicio Transformación de GML.
- Visualización de mapas
  Liga hacia la herramienta perteneciente al servicio Presentación y manipulación de mapas.

Cada sección relacionada con las herramientas de Mappa y sus características serán detalladas a continuación.

6.1.1 Manejo de archivos

En esta sección se encuentran las herramientas necesarias para la administración de archivos. Estas herramientas son (*véase A5.2 – fig. A4*):

- Índice (*véase A5.2 – fig. A5*)
  Listado de los documentos presentes en el servidor. La lista aparecerá divida en tres secciones: GML, XSLT y SVG. Por cada archivo se tienen las siguientes opciones:
  - Ver su descripción, en caso de tener una. Abrirá una ventana en donde se presentará un archivo de texto.
  - Renombrar el archivo. Abrirá una ventana mostrando un textfield (*véase A5.2 – fig.A6*). En caso de que la operación no falle se indicará (*véase A5.2 – fig. A7*) y se actualizará el índice, en caso contrario se mostrará un mensaje de error.

- Cargar documento GML
  Nos permite guardar un documento GML. La carga del documento puede ser tanto de forma de código escrito directamente en un *textArea* o por medio de un *file chooser* (*véase A5.2 – fig.A8*).
- Cargar plantilla XSLT
  Nos permite guardar una plantilla de transformación en el servidor La carga del documento puede ser tanto de forma de código escrito directamente en un *textArea* o por medio de un *file chooser* (la interfaz es idéntica al de la herramienta Cargar documento GML).

- Cargar Descripción
  Nos permite guardar la descripción de un documento existente. Se presenta una lista de documentos dividida en tres secciones (GML, XLST y SVG), cada archivo tiene asociado un *textarea.* Se puede actualizar una sola descripción o todas las que se encuentren en una sección (*véase A5.2 – fig.A9*).
- Eliminar archivos
  Nos permite eliminar archivos que se consideren innecesarios. Se presenta una lista, nuevamente dividida en tres secciones, con los archivos que se encuentran en el servidor. Asociado con cada archivo se encuentra un *checkbox*, el cual servirá para seleccionar los archivos que se quiera eliminar.

6.1.2 Conversión GML

Esta herramienta nos permite transformar documentos GML usando una plantilla XSLT, las opciones que se tienen son las siguientes (*véase A5.3 – fig.A11*):

- Seleccionar el documento GML que se quiere transformar
  Se presenta una lista con los documentos GML disponibles en el servidor.
- Seleccionar la plantilla con la cual se efectuara la transformación
  Se presenta una lista con las plantillas disponibles en el servidor.
- Seleccionar el tipo de salida
  La respuesta de la transformación puede ser texto (documento SVG) o una imagen SVG, esto solamente es una presentación que se le da al usuario como retroalimentación. Internamente se guarda el documento SVG en el servidor para su posterior uso.
- Asociar una descripción al resultado de la conversión.

6.1.3 Visualización de mapas

La visualización de mapas nos permite seleccionar el documento SVG que se desea analizar. Una vez cargado el documento en marco principal (*véase A5.4 – fig.A12*), una ventana de control aparecerá la cual contiene las herramientas para la manipulación del mapa (*véase A5.4 – fig.A13*).

Estas herramientas son:

- Cambiar de orden las capas
- Eliminar capas
- Ocultar capas
- Cambiar los colores de relleno o de contorno de las capas.
- Salvar las modificaciones

Todos los servicios en Mappa están representados por formas HTML, quienes envían los datos necesarios a los Java Servlets del paquete servlets. Para minimizar los fallos o conductas no deseadas debido a datos erróneos o faltantes, todas las formas son validadas por medio de código Javascript.

6.2 Transformación GML

La implementación del servicio de transformación de documentos GML se realizo de la siguiente manera:

Una vez que se tomó la decisión de utilizar plantillas XSLT para realizar la transformación, se necesita una herramienta que aplique la plantilla al documento GML. Existe una variedad de herramientas basadas en Java que nos proveen de este servicio, como lo son Xalan y Saxon.

Ambas herramientas sirven para el mismo propósito, pero cuentan con API's diferentes. Es decir, para realizar una misma tarea, cada herramienta tiene sus propios métodos para realizarla. Esto indica un gran problema debido a que si se programa orientado a Xalan y se corre el sistema en una máquina que tenga instalado Saxon, el sistema simplemente no funcionará.

En pocas palabras sus métodos no son estándar. Para resolver este problema SUN MICROSYSTEMS desarrollo un API denominado JAXP (Java API for XML Parsing) [Sun, 2003], la propuesta de esta tecnología es el uso de métodos estándar para la manipulación de XML sin importar con que herramienta se cuente.

Las clases del paquete JAXP 1.0 que se necesitan para transformar un documento XML es el siguiente:

- Javax.xml.transform.stream.StreamSource
  Actúa como un contenedor del documento fuente, el cual puede ser XML, texto o HTML.

- Javax.xml.transform.stream.StreamResult
  Actúa como un contenedor para el resultado de la transformación.

- Javax.xml.transform.TransformerFactory
  Sirve para poder crear objetos Transformer y Template.

- Javax.xml.transform.Templates
  Actúa como un contenedor de las reglas existentes en la plantilla XSLT y puede crear objetos Transformer.

- Javax.xml.transform.Transformer
  La instancia de esta clase puede ejecutar la transformación del documento.

El proceso de transformación es el siguiente:

- Se instancia un objeto de la clase TransformerFactory
  TransformerFactory fabricaTrans = TransformerFactory.newInstance();

- Se instancia un objeto de la clase StreamSource, que funcionará como el documento origen.
  fuenteXML = new StreamSource(new File(senderoGML));//recibe un nuevo FILE

- Se crea un objeto de la clase Templates por medio del objeto TransformerFactory, cuyo parámetro es la plantilla XSLT.
  editTemplates = fabricaTrans.newTemplates(new StreamSource(new File(senderoXSLT)));

- Se crea un objeto de la clase Transformer por medio del objeto Template, debido a esto el objeto Transformer ya se encuentra "consciente" de las reglas que tiene que aplicar.
  Transformer transformacion= editTemplates.newTransformer();

- Se realiza la transformación; como parámetros se tienen el documento fuente y la salida.
  transformacion.transform(fuenteXML,new StreamResult(new FileWriter(destinoSVG,false)));

En si, el proceso de transformación es simple, gracias al paquete JAXP creado por SUN MICROSYSTEMS. Lo realmente importante de la transformación son las reglas que se tienen en la plantilla XSLT. A continuación se expondrá brevemente las consideraciones que se tuvieron para el desarrollo de las plantillas de transformación para este proyecto.

6.2.1 Plantillas XSLT

La codificación de plantillas XSLT no es complicada, solamente se tiene que tener en mente en que se va a convertir cada elemento del código fuente. En el contexto de este SIG, lo que se tuvo en consideración es lo siguiente:

- El área de pintado será de 800 px por 800 px.
- Las coordenadas reales que vienen en los documentos GML, se tendrán que ajustar al área de pintado.
- El Bounding box del documento GML, será el View box del documento SVG. En caso de que el documento GML no contenga el Bounding Box, este será calculado por la misma plantilla.
- Se consideraran las siguientes entidades:
    - Ríos
    - Carreteras, se hará distinción de varios tipos de caminos: carretera, terracería, brecha, vereda y camellón.
    - Rutas de evacuación
    - Poblaciones
    - Refugios
    - Centros de Acopio
    - Centro de Coordinación Primario
    - Centro Coordinación Secundarios
    - Unidades Médicas
    - Mercados
    - Iglesias
    - Agencias MP
    - Fuentes de Agua Potable
    - Plantas Eléctricas
    - Escuelas
    - Bibliotecas

  Cada entidad tendrá colores distintivos, aunque no definitivos debido a la habilidad del sistema de cambiar colores de cada capa.
- Por cada entidad, se capturará sus atributos descriptivos de manera que con un clic en el elemento se desplieguen al usuario.
- Debido a la versatilidad del elemento “camino” (Path) de SVG, los elementos geográficos del documento GML Polygon, LineString y LinearRing serán transformados a svg:path.

En la actualidad existen herramientas basadas en Java que nos permiten navegar y manipular por el árbol de los documentos GML, como lo son JDOM o DOM.

Se escogió el uso de XSLT para la transformación de los documentos GML sobre estas herramientas por las siguientes ventajas:

- XSLT es de la familia de XML creada por el W3C y su único propósito es el de transformar documentos; JDOM y DOM son creados por otras compañías y son de propósito general.
- El lenguaje XSLT es fácil de aprender, no es necesario ser un programador experto para crear plantillas funcionales; el usuario del sistema tendría que aprender Java y familiarizarse con los API’s de JDOM y DOM.
- La flexibilidad que otorga al sistema de escoger la plantilla a usar sin necesidad de reprogramar el sistema; si se utiliza JDOM o DOM, y si se quisiera tener una salida adecuada a otras necesidades se tendría que reprogramar el sistema.

6.3 Visualización y manipulación de mapas

Como se ha visto los documentos SVG solamente describen entidades geométricas, la visualización correcta de estos documentos depende de programas externos que dibujen el contenido del documento. La aplicación más popular para la visualización de documentos SVG es el Adobe SVG Viewer. Esta aplicación es un plug in para Internet Explorer o Netscape Navigator. El documento se visualiza en la ventana del navegador y se puede hacer uso de herramientas incrustadas en el plug in, por ejemplo panear la imagen, acercar o alejar la imagen.

Esto funciona perfectamente teniendo el documento SVG como un archivo, pero los requerimientos del sistema demanda que el documento SVG estuviera asociado a una pagina electrónica. Esto se logra utilizando la siguiente etiqueta HTML.

< embed source=”mapa.svg” >

Fácilmente se puede visualizar en el cliente un documento que se encuentre en el servidor y la manipulación se puede implementar por medio de código JavaScript. Si se cambiaba un atributo, por medio de JavaScript, se rescribía directamente al árbol generado por el plug in.

Lamentablemente este enfoque tiene las siguientes desventajas:

- El número de variables que se necesitaban para poder tener control y recordar el estado de los atributos en las capas se vuelve exageradamente grande.
- No existe la forma de saber con que capas cuenta el mapa, por lo que en las opciones de manipulación puede haber elementos que ni siquiera existen en un determinado mapa.
- Las modificaciones que se hacen son en el cliente y no se podrán recuperar para actualizar el mapa.

Para resolver estos problemas se tuvo que replantear la manera en la que se deberá de presentar el mapa. Si anteriormente el uso de Servlets solamente era para incrustar el nombre del documento en la etiqueta de <embed>, ahora tendrían que tener un papel más importante.

Partiendo de la ventaja que los Servlets pueden guardar objetos, para poder distinguir sesiones, en el contexto del servidor se planeo la siguiente arquitectura de 3 niveles.

- La clase AdministradorCapas, se encarga de contener el árbol del documento SVG y de manipularlo. En esta clase se hace uso del paquete org.xml.dom, que se encuentra en el API Enterprise Edition de Java.
- La clase ServletAdministredorCapas funciona como un intermediario quien comunica las ordenes del usuario al AdministradorCapas y da respuesta al usuario.

Fig. 6.1 *Arquitectura de la visualización.*

A cualquier petición de manipulación por parte del usuario, se invocará al ServletAdministradorCapas, el cual buscará al AdministradorCapas para darle la orden de manipulación. La respuesta será manejada por el ServletAdministradorCapas y regresada al usuario.

De manera práctica, a la orden de visualizar un mapa:

- El usuario provee del nombre del mapa.
- El ServletAdministradorCapas recibe la orden de desplegar el mapa, crea un objeto AdministradorCapas, lo guarda en el contexto del servidor y le ordena que cargue el mapa indicado.
- El servletAdministradorCapas recibe el árbol DOM del documento SVG y lo presenta al usuario.

Las propiedades de este enfoque permiten que la cantidad de variables a utilizar disminuya, se sepa con exactitud que capas se encuentran en el mapa y el mapa se pueda guardar nuevamente en el servidor.

Las desventajas de este enfoque son las siguientes:

- Limitaciones en la memoria del servidor
  Si se carga un mapa con capas muy pesadas o si se tienen muchas sesiones en el mismo servidor, pude que el desplegado del mapa no funcione.
- Repintado de la imagen
  En cada modificación del mapa se le envia al Java Servlet una nueva imagen para que la presente. Entonces si se tenía la imagen con un zoom apuntando a un objeto en específico, al repintarse la imagen se perderá el zoom.

6.4 Pruebas y evaluación del sistema

En el Laboratorio de Tecnologías de Geo Información se cuentan con diversas aplicaciones que generan documentos GML, al principio del proyecto el formato de estos documentos era GML 1.0. Estos primeros documentos englobaban todas las entidades geográficas dentro de un solo FeatureCollection.

Conforme el proyecto del Plan Popocatépetl fue madurando se contempló un formato uniforme de información geográfica, con el fin de estandarizar los documentos GML que son generados en el Laboratorio. Este formato no es precisamente GML, propiamente dicho, es una aplicación XML en donde se unen elementos con namespace GML y elementos que fueron creados en el Laboratorio con namespace gisonline.

Para codificar elementos geográficos con esta nueva aplicación se requiere de mucho más código que con la codificación en GML, pero se encuentra más estructurado no dejando espacio a ambigüedades.

La forma en la que se diseño la aplicación muestra su utilidad al tener distintas versiones de documentos geográficamente descriptivos. Ya que no se tiene que analizar o rediseñar la aplicación, solamente se tiene que crear una nueva plantilla XSLT adecuada a cada versión de documento descriptivo.

Para poder evaluar el rendimiento del sistema se crearon tres plantillas XSLT:

- Plantilla general
  Esta plantilla se generó en etapas iniciales del proyecto, toma en cuenta la codificación inicial de los documentos GML. Puede procesar cualquier entidad mientras que su atributo geográfico sea un polígono, línea o punto. El formato que se le da a la figura debe estar contenido en el documento GML.

- Plantilla específica 1
  Esta plantilla se creó para poder procesar las entidades descritas en el apartado 6.2.1, toma en cuenta la codificación inicial del laboratorio.

- Plantilla específica 2
  Al cambiar al nuevo lenguaje para describir entidades geográficas, las plantillas anteriores quedaron restringidas en su campo de acción. Así que esta plantilla surge como respuesta a la necesidad de procesar los nuevos documentos que son generados en el Laboratorio.

Para constatar el desempeño de la aplicación se hicieron pruebas con distintos documentos GML, usando las plantillas mencionadas anteriormente.

Estas pruebas consistieron en:

- Cargar el documento GML al servidor.
- Realizar la conversión de GML a SVG.
- Visualizar y manipular el mapa.

El resultado a las pruebas fue satisfactorio, pero se tienen las siguientes consideraciones:

- Tanto en los documentos GML como en las plantillas de transformación su validez se revisa hasta que se realiza la transformación, se permite la carga de documentos inválidos.
- Debido al contexto web se requiere más memoria para trabajar, con documentos moderadamente grandes la aplicación se queda sin memoria provocando una excepción OutOfMemory.
- La presentación de la información descriptiva se realiza por medio de "alert's" de JavaScript, debido a que la generación de ventanas HTML se bloquea.

6.5 Caso práctico

Esta sección presenta un caso práctico del uso del sistema, el objetivo de este ejercicio es mostrar el funcionamiento del sistema y su desenvolvimiento.
El documento GML que se utilizara en este ejemplo, es representativo de las poblaciones correspondientes a la ruta de evacuación número 2 (véase ejemplo 6.1).

```
<?xml version="1.0"?>
<mapa>
    <FeatureCollection typeName="Ruta2">
        <boundedBy>
            <Box srsName="EPSG:UTMZ14"><coordinates> puntos </coordinates></Box>
        </boundedBy>
        <featureMember typeName="Poblaciones">
            <Feature typeName="Poblaciones" identifier="13" >
            <property typeName="ID" type="real" >2.0</property>
            <property typeName="POBLACION" type="string" >SanAndresCalpan</property>
            <property typeName="NUM_MUNICI" type="real" >0.0</property>
            <property typeName="RUTA" type="real" >2.0</property>
            <property typeName="ZONA" type="string" >Mediano</property>
            <geometricProperty typeName="polygonProperty" >
                <Polygon srsName="EPSG:UTMZ14">
                    <outerBoundaryIs>
                      <LinearRing><coordinates>555275,2113252</coordinates></LinearRing>
                    </outerBoundaryIs>
                </Polygon>
            </geometricProperty>
            </Feature>
        </featureMember>
    …
    </FeatureCollection>
</mapa>
```

Ejemplo 6.1 *Fragmento del documento GML "caso de prueba".*

Por cuestión de espacio solamente se presenta un fragmento del documento, modelando una población, en donde se anexan los siguientes datos descriptivos:

- Id.
- Nombre.
- Número de municipio, al que pertenece.
- Ruta de evacuación, a la que pertenece.
- Zona, de riesgo.

Este documento al ser transformado a SVG, queda de la siguiente manera:

```
<?xml version="1.0" encoding="utf-8"?>
<svg width="800" height="800" viewBox="207 0 8 4 ">
    …
    <g id="Poblaciones" visibility="visible" fill="black" stroke="blue" stroke-width="0.003 ">
        <a  xlink:href="">
            <path onclick="alert('id: 2.0' +'\n'+ 'nombre: SanAndresCalpan' +'\n'+'municipio: 0.0' +'\n'+ 'ruta: 2.0'
            +'\n'+ 'zona: Mediano')" d="M 209.50252330345035 2.680470096066056 L 209.53479866055244
            2.7107282433493687"/>
        </a>
        <a  xlink:href="">
            <path onclick="alert('id: 18.0' +'\n'+ 'nombre: SanMateoOzolco' +'\n'+ 'municipio: 0.0' +'\n'+ 'ruta: 2.0'
            +'\n'+ 'zona: Alto')" d="M 208.05639127760492 3.186025013516417 L 208.04728143770953
            3.2153789420681105 "/>
        </a>
    </g>
    …
</svg>
```

Ejemplo 6.2 *Fragmento del documento SVG "caso practico".*

El fragmento describe la capa de poblaciones (por cuestión de espacio solamente se presentan dos entidades), las poblaciones tiene el siguiente formato:

- Son visibles.
- Tienen relleno color negro.
- El contorno es de color azul con un grosor de 0.003.

Los atributos descriptivos son asociados al elemento <svg:path> en el atributo onclick, cuando se presione sobre la población, esta reaccionara presentando una ventana con su información (véase figuras 6.2).

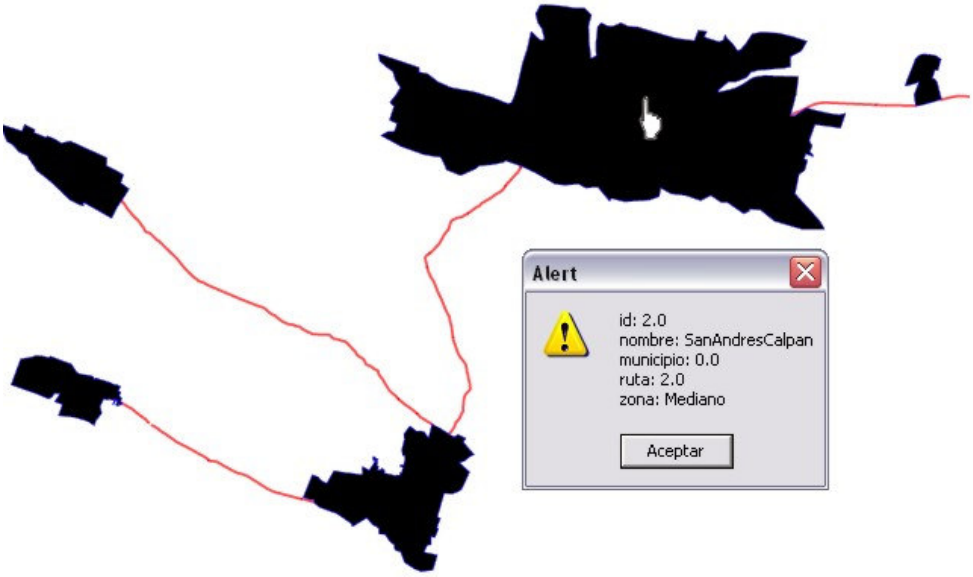

Fig. 6.2 *Información descriptiva.*

A continuación se cambiara el formato de las poblaciones a relleno azul alice y contorno negro, se ocultara la capa de rutas de evacuación (véase figura 6.3).

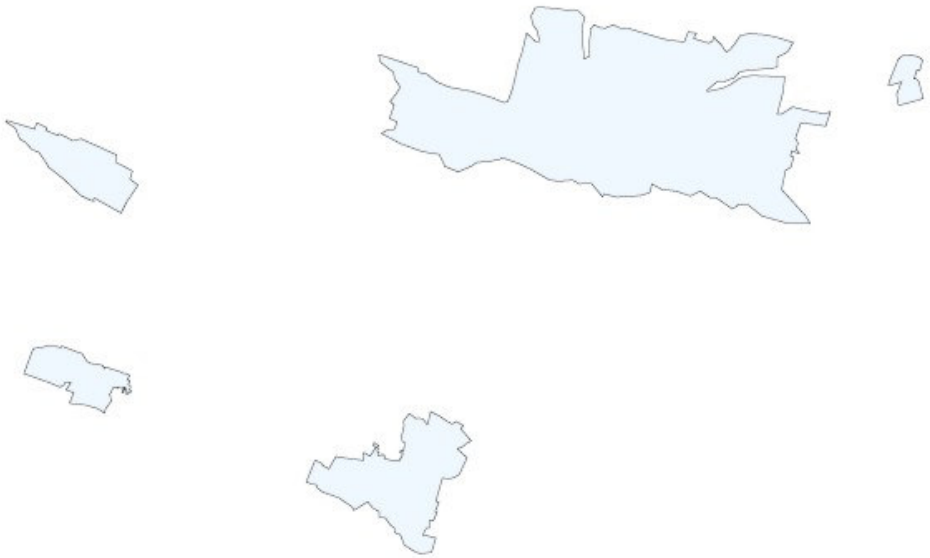

Fig. 6.3 *Manipulación del mapa*.

Una vez modificado el mapa se guarda en el catalogo de documentos SVG. .Como se puede ver las opciones que ofrece el sistema permiten la creación y manipulación dinámica de mapas utilizando una interfaz en la Internet.

6.6 Conclusiones

Esta sección presenta un resumen del proyecto junto con una serie de iniciativas para extender y mejorar el trabajo realizado.

6.6.1 Resultados

Con el desarrollo de Mappa se logró transformar y visualizar de manera apropiada la información de la geo base del proyecto del volcán Popocatépetl.

Se alcanzaron los siguientes resultados:

- Se estructuró la información geográfica con GML.
- La implementación fue realizada tomando como base estándares propuestos por OpenGis Consortium, y el W3C.
- Se crearon las interfaces para el proyecto con servicios de transformación y presentación de la información geográfica disponible.
- Se generaron mapas en dos dimensiones, presentando puntos, líneas y polígonos según el modelo GML.
- Se construyeron servicios sobre documentos, estos servicios incluyen el almacenamiento y procesamiento.

6.6.2 Trabajos a futuro

El desarrollo de este proyecto involucró diversas áreas y a continuación se mencionan las áreas de oportunidad para mejorar el sistema.

En el contexto de consultas:

- Realizar consultas con elementos bidimensionales, por ejemplo la distancia entre dos objetos o si un objeto se encuentra al interior de otro.

En el contexto de XML:

- Definir esquemas de XML para aplicaciones sobre la información en GML.
- Generar versiones mejoradas de las plantillas, ya que el actual desempeño es lento para documentos muy grandes.
- Definir nuevas plantillas de traducción que se enfoquen a diferentes contextos, por ejemplo catastro o urbanismo.

Servicios:

- Definición de documentos con metadatos incluidos.
- Mejorar las herramientas para la interacción con los mapas.

6.6.3 Conclusiones

Una herramienta que pueda servir como soporte a la toma de decisiones para desastres naturales, es de mucha utilidad, debido a que previenen desastres y reducen costos.

La facilidad de tener datos en el momento preciso es fundamental dentro del contexto donde se desenvuelve nuestra aplicación. Desgraciadamente las herramientas que se tienen para este tipo de problemática, son herramientas demasiado generales que no toman en cuenta principios que ayudarían a expresar de manera adecuada y contundente los resultados de las búsquedas. El trabajo realizado en este proyecto pretende ayudar a las autoridades de protección civil en actividades de prevención de desastres, crear mejores planes de evacuación y en caso de contingencia apoyar a la toma de decisiones de manera rápida y precisa.