UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

SISTEMA DE GESTIÓN Y VISUALIZACIÓN DE MUESTRAS MEDIOAMBIENTALES EN CONTEXTO GEOGRÁFICO

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN

MATÍAS ADRIÁN CISTERNA MADRID

PROFESOR GUÍA:
SERGIO OCHOA DELORENZI

MIEMBROS DE LA COMISIÓN:
NANCY HITSCHFELD KAHLER
BENJAMÍN BUSTOS CÁRDENAS

SANTIAGO DE CHILE
2017

Resumen

SISTEMA DE GESTIÓN Y VISUALIZACIÓN DE MUESTRAS MEDIOAMBIENTALES EN CONTEXTO GEOGRÁFICO

En el contexto de la gran minería se debe mantener un constante monitoreo de diversas variables medio ambientales (por ej., el caudal del agua que se utiliza, la temperatura del agua, concentración de sustancias, etc.) por razones de seguridad, de impacto medioambiental y de previsión de disponibilidad de los recursos naturales en el futuro inmediato. Para dicho monitoreo las empresas mineras utilizan estaciones de medición, que típicamente son una estructura con diversos sensores. Dichas estaciones permiten tomar las mediciones de los datos requeridos, los que posteriormente son recogidos para ser almacenados y usados para apoyar procesos de toma de decisiones.

La recolección de estos datos se debe hacer de manera periódica, ya que las estaciones de medición están capturando y almacenando esta información constantemente. Actualmente la gestión y visualización de datos no se encuentra automatizada en la empresa minera donde se realizó esta memoria, por lo que se debe invertir tiempo en el almacenamiento y posterior interpretación de los datos obtenidos, antes de poder tomar una decisión en este ámbito.

Bajo ese contexto, en el presente trabajo se desarrolló una herramienta computacional basada en Tecnologías de la Información, la cual apoya la recuperación y gestión de datos de mediciones medio ambientales, así como la entrega de información geográfica asociada a estos datos y su visualización en el contexto geográfico correspondiente. La aplicación desarrollada consiste en un sistema Web, que permite al usuario realizar la gestión de los datos obtenidos desde las estaciones de medición instaladas por la empresa; y a través de la representación visual de diversos indicadores, apoya los procesos de toma de decisión. Esta aplicación contiene dos grandes componentes: (1) un gestor de datos medio ambientales, y (2) un *dashboard* interactivo para la visualización de datos en contexto geográfico y apoyo a la toma de decisiones.

Debido a limitaciones propias de un trabajo de memoria y la complejidad del problema abordado, el proyecto realizado involucró el desarrollo de la herramienta a nivel de prototipo piloto, con el fin de demostrar su factibilidad de implementar para la empresa en cuestión. La aplicación cuenta con las funcionalidades necesarias y suficientes para demostrar que una solución con tales características puede ser implementada y puesta en producción, y para ellos se la probó con distintos casos de uso y potenciales usuarios finales. Si bien las pruebas realizadas revelaron que ésta necesita más inversión de horas de ingeniería para poder ponerse en producción, la aplicación está encaminada para convertirse en un producto viable en el futuro, no sólo para la empresa destino, sino también para otras empresas similares.

Tabla de Contenido

Capítulo 1: Introducción

Soporta Limitada es una empresa dedicada al desarrollo e integración de soluciones de TI para distintas industrias; pero principalmente para la gran minería y el gobierno. Particularmente, la minería es una de las actividades más importantes dentro de la economía en Chile, por lo que las empresas dedicadas al rubro de la explotación de mineral han crecido sustancialmente durante los últimos 30 años, liderando diversos temas en el ámbito nacional, como ingresos, desarrollo productivo e innovación.

Muchas actividades se realizan en el contexto de una explotación minera, como por ejemplo la extracción del mineral, así como la fundición y refinación de éste. Todas estas actividades requieren el uso de maquinaria pesada de alta tecnología, así como el recurso humano que se encarga del manejo y mantenimiento de ésta. Debido a este crecimiento, nace la necesidad de contar con grandes cantidades de suministros, como lo son agua y energía de manera constante, servicios que no pueden ser suministrados por empresas dedicadas al rubro. Esta limitante se da por razones de escala y también por temas logísticos, ya que por la naturaleza de la empresa minera ésta debe localizarse en el lugar del yacimiento minero, razón por la cual surge la necesidad de auto-abastecerse. Con este fin se construyen estructuras hidráulicas cerca de ríos, las cuales permiten derivar el cauce del agua y así poder usarla como abastecimiento para la generación de energía, el enfriamiento de instalaciones, etc.

En estas estructuras, llamadas *bocatomas*, se debe mantener un constante monitoreo de diversas variables (por ej. el caudal, la temperatura del agua, concentración de sustancias, etc.) por razones de seguridad, de impacto medioambiental, y de previsión de disponibilidad de ese recurso en el futuro inmediato, entre otras. Para dicho monitoreo las empresas mineras utilizan estaciones de medición, que típicamente son una estructura con diversos sensores. Dichas estaciones permiten tomar las mediciones de los datos requeridos, los que posteriormente son recogidos para ser almacenados y usados para apoyar procesos de toma de decisiones; generalmente orientado a asegurar el auto-abastecimiento y la continuidad operativa de la faena minera.

La recolección de estos datos se debe hacer de manera periódica, ya que las estaciones de medición están capturando y almacenando estos datos constantemente. Actualmente la gestión y visualización de ellos no se encuentra automatizada, por lo que se debe invertir tiempo en el almacenamiento y posterior interpretación de los datos obtenidos. Esto impide realizar un análisis ágil, por ejemplo, para detectar anomalías en los datos (picos o valles fuera de las bandas de control) que podría conllevar a la generación de contratiempos operativos en el corto y mediano plazo.

Si bien el problema expuesto anteriormente nace desde la minería, no sólo aplica a este tipo de empresas. El mismo problema se puede ver en empresas dedicadas a la oceanografía, donde se usan cadenas de termistores para medir la temperatura del agua a distintas profundidades, aplicando el mismo principio; o sea, se almacenan datos en la estación de medición y luego estos deben ser

colectados, procesados, representados e interpretados. Con esto se puede ver que la necesidad de resolver este problema está presente en diversos ámbitos de aplicación. Si bien esto representa un desafío, también es una oportunidad de desarrollar una solución generalizable, para cubrir la necesidad de diversas empresas que necesitan un sistema de gestión y visualización (en contexto geográfico) de muestras medioambientales.

Bajo este contexto surge la oportunidad de desarrollar un sistema que le permita a la empresa Soporta Limitada, almacenar, gestionar y visualizar los datos obtenidos por estaciones de medición, pudiendo así cargar los datos, gestionar una base de datos con estos, configurar alertas/alarmas en caso de desborde de bandas de control por parte de alguna variable monitoreada. Se busca que todo esto esté embebido en un *dashboard* interactivo, donde se pueda visualizar un mapa con cada una de las estaciones de trabajo y sus respectivos datos mostrados en tablas y gráficos.

La solución implementada en el presente trabajo consta de una aplicación Web bajo la arquitectura Modelo-Vista-Controlador y base de datos relacional para la gestión de datos medio ambientales y geográficos, así como un despliegue de los mismos en una interfaz de usuario final en forma de *dashboard* interactivo. Para la consulta de datos se implementó una capa de servicios REST que se consultan directamente desde la interfaz cliente. Estos servicios no consumen los datos directamente de la base de datos relacional debido al requisito de que estas consultas sean eficientes, debido a la gran cantidad de datos esto no es posible Para esto se implementó una solución de Inteligencia de Negocios transformando el modelo relacional en un Data Mart, procesando y llevando los datos a un Cubo OLAP de dimensiones acorde a lo necesario y así hacer consultas de manera eficiente.

1.1 Justificación de la solución

Desde el punto de vista computacional, el plantear una solución al problema expuesto involucra tener que abordar distintas aristas. Por ejemplo, se debe poder manejar grandes cantidades de datos, ya que las estaciones de medición están en un proceso constante de recolección. Estos datos deben almacenarse periódicamente en una base de datos, asegurando la integridad de los mismos y considerando la velocidad de generación de estos, para que no se pierda ninguno.

Por otra parte, se debe poder mostrar esta información de manera simple, para que el usuario pueda comprenderla rápidamente y poder tomar decisiones en base a ella. Particularmente se pretende entregar al usuario un *dashboard* interactivo, para revisar indicadores por demanda. Dada la cantidad de variables a monitorear y la cantidad de estaciones de monitoreo que podría haber en una explotación minera, este aspecto de visualización de datos representa un importante desafío a sortear. Finalmente, se debe contar con una forma eficiente de consulta de datos, asegurando mínima carga del lado del cliente, y realizando la mayor parte del cómputo en el servidor.

Actualmente los datos tomados desde las estaciones de medición son guardados en hojas de cálculo, por lo que obtener datos históricos de las muestras se vuelve un trabajo tedioso. Además, el esfuerzo obtener dicha información y la confiabilidad de la misma depende meramente del empleado de turno, a cargo de organizar estas hojas de cálculo. Esto se debe a que tampoco existe un estándar o protocolo de almacenamiento de los datos, por lo que estos podrían ser etiquetados con la fecha de medición, sólo si el empleado de turno está de humor para hacerlo.

Con la implementación del sistema propuesto en esta memoria, la obtención de datos históricos viene dada por defecto, y el trabajo que antes no era riguroso ni confiable, ahora sería preciso y lograble a través de un par de clicks. La necesidad de visualización de datos en forma gráfica es uno de los mayores desafíos del proyecto, ya que debido al gran volumen de datos y a la interdependencia que hay entre ellos, no resulta factible consultarlos directamente desde la base de datos relacional a través de un servicio Web. Vale la pena aclarar que la mayoría de los indicadores medioambientales se calculan teniendo en cuenta interrelaciones entre las variables medioambientales monitoreadas. Además, el volumen de información a manejar plantea diversos desafíos al momento de verificar que los datos ingresados y/o modificados siguen las reglas impuestas de antemano, en particular la existencia de bandas de control en cada una de las mediciones. Para esto se propone contar con una solución de Business Intelligence, basada en un Cubo OLAP, donde los datos serían pre procesados al momento de ser cargados, generando así una nueva base de datos multidimensional con el propósito de apoyar el análisis instantáneo de los datos recolectados.

Para la visualización en contexto geográfico es necesario utilizar una base de datos geográfica, con todo el desafío que esto conlleva. Particularmente, el proponente deberá familiarizarse con una nueva tecnología, así como manejar muy bien los estándares usados.

1.2 Objetivos de la memoria

El objetivo general de este trabajo de memoria es sistematizar el proceso de compilación de los datos generados por estaciones de monitoreo, dejando a disposición del usuario final tanto un sistema de gestión de esos datos, como un sistema de visualización de los mismos. Para esto se diseñará e implementará una aplicación piloto consistente en un sistema Web, que permita al usuario realizar la gestión de los datos obtenidos desde las estaciones de medición instaladas por la empresa, y a través de la representación visual de diversos indicadores, apoyar los procesos de toma de decisión.

Para alcanzar este objetivo general se plantean los siguientes objetivos específicos:

1) Diseñar e implementar un gestor de muestras de medición donde se puedan cargar y luego administrar los datos ingresados al sistema.

2) Diseñar e implementar un *dashboard* interactivo, destinado a la visualización (en un contexto geográfico) de las estaciones de medición, sus respectivos datos recolectados e indicadores asociados.
3) Validar el trabajo realizado. Para esto someter a prueba la aplicación bajo el contexto de usuarios de distintos perfiles, como son profesionales del área de geografía, del área de desarrollo de software y usuarios comunes.

1.3 Alcance de la memoria

Los alcances de este trabajo se limitan a desarrollar una aplicación piloto de muestra, con el fin de poder exponer lo que un sistema de estas características puede lograr y el valor que puede agregar a los datos ya existentes.

Este trabajo no busca realizar un sistema de información completo para su puesta en producción, por lo que se hacen simplificaciones y que serán analizadas en una futura versión del sistema.

En el siguiente capítulo se desarrolla un marco teórico donde se introducen los principales conceptos utilizados durante el desarrollo de esta memoria, esta introducción se realiza de manera concisa para no desviar el tema principal. En el capítulo 3 se introducen las principales características de la solución, dadas por sus requerimientos de usuario y de software. En el capítulo 4 se muestra en detalle el diseño de la solución en cada una de las secciones. En el capítulo 5 se muestra la implementación de la aplicación. En el capítulo 6 se presenta una validación de la aplicación hecha con potenciales usuarios, los que manipularon las funcionalidades implementadas y luego contestaron un breve cuestionario. En el capítulo 7 se realizan las principales conclusiones que se obtuvieron a lo largo del desarrollo del presente y se analiza el trabajo a realizar a futuro, y finalmente el capítulo 8 contiene la bibliografía utilizada a lo largo del desarrollo del proyecto y el capítulo 9 los anexos.

Capítulo 2: Marco Teórico

El sistema desarrollado es un Sistema de Información Geográfica, concepto no necesariamente familiar para las ciencias de la computación, por lo que se procederá a una breve introducción para el mejor entendimiento del proyecto completo. En el anexo 8.1 se pueden encontrar conceptos más básicos de las geociencias.

2.1 Sistema de Información Geográfica

Un Sistema de Información Geográfica (SIG) es una tecnología que involucra una gran variedad de usos y aplicaciones en distintos ámbitos de la industria. En estricto rigor un SIG se define como la integración de hardware, software, recursos humanos y procedimientos, destinados a la captura, almacenamiento, gestión, análisis y despliegue de datos referenciados espacialmente en forma gráfica y alfanumérica. Sin embargo, en términos del alcance de este trabajo se entenderá por SIG el conjunto de herramientas de software destinadas a la gestión y despliegue gráfico de los datos referenciados espacialmente. Ejemplos de uso de SIG:

- Elementos que se pueden encontrar en un territorio en particular: fauna, vegetación, población, monumentos naturales, etc.
- Ocurrencia de hechos concretos: migración de animales, desastres naturales, celebración de festividades, etc.
- Distribución espacial de fenómenos: Contaminación atmosférica, propagación de enfermedades.
- Tendencias temporales: evolución del espacio urbano, crecimiento de sectores de la población, avance de una exploración minera.

El uso del SIG es transversal para toda actividad humana[1], de ahí el valor que adquiere la implementación de estos.

2.2 Inteligencia de Negocios

Formalmente se define la Inteligencia de negocios como al conjunto de estrategias, aplicaciones, datos, productos, tecnologías y arquitectura técnica, los cuales están enfocados a la administración y creación de conocimiento sobre el medio, a través del análisis de los datos existentes en una organización o empresa.

En lo que respecta a este trabajo de memoria la inteligencia de negocios que se aplicará será procesamiento analítico en línea, con el objetivo de optimizar consultas hechas a almacenes de

[1] Un ejemplo actual sobresaliente es *Google Earth*.

datos diseñados para este propósito, en particular se utilizará una estructura de datos multidimensional, un cubo OLAP el que será alimentado vía un proceso ETL desde el modelo de datos principal de la aplicación, para su posterior consulta de manera eficiente. Se procederá a explicar conceptos asociados al desarrollo de la solución de inteligencia de negocios del presente trabajo.

2.2.1 Data Warehouse y Data Mart

Data Warehouse es una colección de datos orientada a un determinado ámbito (empresa, organización, etc.), que ayuda a la toma de decisiones en la entidad en la que se utiliza. La ventaja principal de este tipo de sistemas se basa en su concepto fundamental, la estructura de la información. Este concepto significa el almacenamiento de información homogénea y fiable, en una estructura basada en la consulta y el tratamiento jerarquizado de la misma, y en un entorno diferenciado de los sistemas operacionales. Según definió Bill Inmon (creador de todos estos conceptos), el Data Warehouse se caracteriza por ser:

- ***Integrado***: los datos almacenados en el Data Warehouse deben integrarse en una estructura consistente, por lo que las inconsistencias existentes entre los diversos sistemas operacionales deben ser eliminadas. La información suele estructurarse también en distintos niveles de detalle para adecuarse a las distintas necesidades de los usuarios.
- ***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.
- ***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 Data Warehouse sirve, entre otras cosas, para realizar análisis de tendencias. Por lo tanto, el Data Warehouse se carga con los distintos valores que toma una variable en el tiempo para permitir comparaciones.
- ***No volátil***: el almacén de información de un Data Warehouse existe para ser leído, y no modificado. La información es por tanto permanente, significando la actualización del Data Warehouse la incorporación de los últimos valores que tomaron las distintas variables contenidas en él sin ningún tipo de acción sobre lo que ya existía.

Por otro lado el Data Mart es una colección de datos que cumple la misma función que el data Warehouse con todas sus características, pero es más pequeño ya que no incluye toda la información de la empresa sino que solo la necesaria para las consultas y toma de decisiones.

2.2.2 Proceso ETL

Es la sigla para *extract*, *transform*, *load*. Es el proceso que se ocupa transferir datos desde una fuente a otra. Como la sigla los dice, los pasos que sigue para la transferencia son los siguientes:

1. Extrae los datos desde la fuente de origen.
2. Transforma los datos procesándolos para que estos se ajusten a la nueva estructura de la fuente de destino.
3. Carga los datos ya transformados en la fuente de destino.

En el contexto de la memoria se definió un proceso ETL para la carga del Data Mart que alimenta el cubo OLAP, desde la fuente de origen que es la base de datos relacional.

2.2.3 Cubo OLAP

OLAP es el acrónimo en inglés para procesamiento analítico en línea (*On-Line Analytical Processing*) y su objetivo es agilizar la consulta de grandes cantidades de datos. Para ello utiliza estructuras de datos diversas, normalmente multidimensionales, que contienen datos resumidos de grandes bases de datos. Una de estas estructuras de datos multidimensionales es el cubo OLAP, base de datos multidimensional en la cual el almacenamiento físico de los datos se realiza en un vector multidimensional, el que garantiza un acceso muy rápido a estos. En la figura 2.1 se muestra un cubo OLAP de cantidad de ventas con tres dimensiones: color, producto y año. Se puede apreciar que el año 2010 se vendieron 996 bicicletas plateadas, a diferencia de las negras que solo se vendieron 425. Se puede ver en el ejemplo las tendencias de los compradores y por lo tanto ayuda a las empresas en la toma de decisiones.

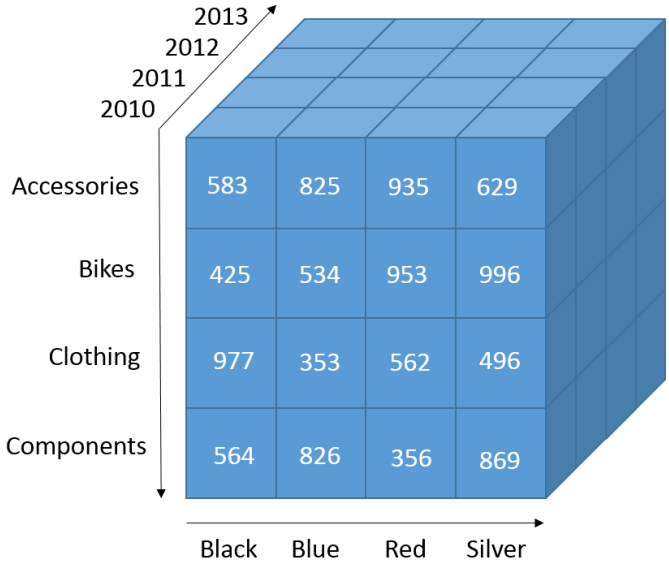

Figura 2.1: Cubo OLAP de cantidad de ventas en una compañía.

2.2.4 Lenguaje de consulta MDX

MultiDimensional eXpressions es un lenguaje de consultas especializado para base de datos OLAP. Una consulta MDX es muy similar a una consulta SQL, nos devuelve un conjunto de celdas, que es resultado de tomar un subconjunto de las celdas del cubo original. Sin embargo, contiene diferentes funciones y al utilizar varias dimensiones puede volverse bastante compleja.

MDX utiliza en varias situaciones las jerarquías. Por ejemplo, si una dimensión se denomina región, esta puede contener países. Los países a su vez contienen provincias y las provincias ciudades. Para manejar estos componentes MDX tiene funciones como *Children* (hijos en inglés), *cousin* (primos) y *parents* (padres). En el ejemplo de la dimensión región, el país sería el padre, los hijos las provincias.

Consulta MDX básica (sintaxis):

```
SELECT <especificación de eje> ON COLUMNS,
<especificación de eje> ON ROWS
FROM <especificación de cubo>
WHERE <especificación Slicer (rebanador)>
```

2.3 Formato de intercambio: KML

KML es un subconjunto de XML definido por Google (y estándar de la OGC, especificado en [4]) enfocado principalmente en la visualización de información geográfica, incluyendo anotaciones en mapas vectoriales o raster. Es utilizado principalmente en visualizadores de mapas como *Google Earth*, *Google Maps* y *Google Maps Mobile*.

Ejemplo KML

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Placemark>
    <name>Simple placemark</name>
    <description>Attached to the ground. Intelligently places itself
       at the height of the underlying terrain.</description>
    <Point>
      <coordinates>-122.0822035425683,37.42228990140251,0</coordinates>
    </Point>
  </Placemark>
</kml>
```

Figura 2.2: Ejemplo de KML

Del ejemplo visto en la figura 2.2 es posible apreciar que, junto con la información espacial de los vectores, KML agrega información asociada como anotaciones. El elemento *Placemark* representa una de las funcionalidades más utilizadas por *Google Earth*, y que corresponde a etiquetas de información georreferenciada. En el ejemplo particular está compuesto por un nombre y una descripción, asociados a un punto en coordenadas geográficas (angulares, o *latlon*).

2.4 Representación WKT

En las especificaciones para manejo de geometrías, el OGC propone dos maneras estándar para manejar objetos espaciales [6]: Well-Known Binary (WKB) y Well-Known Text (WKT). Ambas incluyen información acerca del tipo de geometría y las coordenadas que definen al objeto.

En el presente trabajo se usará WKT: es utilizada para dar una representación alfanumérica de cualquier geometría. Esta representación debe ser conforme a una gramática (definida en anexo 9.2). Ejemplos de geometrías en WKT en la figura 2.3.

Ejemplos WKT

```
POINT(0 0)
LINESTRING(0 0,1 1,1 2)
POLYGON((0 0,4 0,4 4,0 4,0 0),(1 1, 2 1, 2 2, 1 2,1 1))
```

Figura 2.3: Ejemplo de geometrías en WKT

Capítulo 3: Concepción de la solución

En este capítulo se presenta la conceptualización de la idea planteada que se utilizará para solucionar el problema encontrado, en forma de requerimientos de usuario para la aplicación y posteriormente requerimientos de software.

3.1 Requerimientos de usuario

Los principales requisitos de usuario (RU) que la aplicación debe cubrir son los siguientes:

Código	Descripción del requerimiento
RU1	*Obtención de los datos*: La plataforma debe ser capaz de obtener los datos de interés de manera manual (a través de la carga de los datos mediante un archivo) u obteniéndolos desde algún otro servicio web, para posteriormente ser almacenados en una base de datos.
RU2	*Almacenamiento de datos geoespaciales*: La aplicación debe ser capaz de cargar y almacenar datos geoespaciales (líneas, puntos, polígonos).
RU3	*Preparación de los datos*: Dado que muchas veces las mediciones son demasiadas para trabajar directamente haciendo consultas sobre ellas en la base de datos, la plataforma debe ser capaz de hacer un procesamiento de los datos para que se pueda acceder a ellos de una manera mucho más rápida y fácil a la hora de tener que desplegar esta información a los usuarios.
RU4	*Variables dinámicas*: Se deben poder guardar distintos tipos de variables, tanto el tiempo dado por una fecha específica como el valor numérico de la concentración de cierto compuesto en el agua.
RU5	*Ubicación espacial de las zonas/puntos de medición y los sectores de estudio*: Dado que los puntos donde se toman los datos están distribuidos en distintos lugares, la plataforma proveerá un mapa donde visualizar las zonas de medición, y por ende también, los sectores de estudio.
RU6	*Despliegue gráfico de mediciones*: Dada una zona de medición en particular se debe desplegar de forma agregada todas las mediciones cargadas hasta la fecha asociadas a esa zona. Dicha agregación debe ser dinámica por parte del usuario.
RU7	*Información general sobre zonas de medición y sectores de estudio*: Una vez ubicadas las zonas de medición sobre el mapa, se podrá obtener información detallada sobre dicha zona/punto, como por ejemplo el nombre, su ubicación, fecha de toma de datos (inicio y último registro), frecuencia con que se toman los datos, entre otros.
RU8	*Alertas*: La aplicación debe permitir configurar bandas de control para las variables medidas, y alertar sobre no conformidades relacionadas a estas restricciones (mediciones fuera de las banda de control).
RU9	La aplicación debe ser escalable, y en un futuro permitir recolectar los datos directamente desde las estaciones.

Tabla 3.1: Principales requisitos de usuario

3.2 Requerimientos de software

Los requerimientos de software (RS) que nacen a partir de los requerimientos de usuario son los siguientes:

Código	RU Asociado	Descripción de requerimiento
RS1	RU1	Implementar módulo de carga masiva de datos a partir de un archivo tabulado, que posteriormente almacene estos en la base de datos.
RS2	RU2	Implementar módulo de carga de geometrías a partir de un archivo, que posteriormente los almacene en la base de datos.
RS3	RU3	Diseñar nuevo modelo que se ajuste de mejor manera a la creación de un cubo OLAP.
RS4	RU3	Implementar solución de inteligencia de negocios que use el modelo implementado en el requisito anterior.
RS5	RU3	Implementar módulo de consultas al cubo OLAP, usando lenguaje mdx y retornando objetos serializables.
RS6	RU3, RU6	Implementar capa de servicio REST que comunique la aplicación cliente con el módulo de consultas al cubo OLAP, entregando los datos solicitados serializados y listos para su despliegue.
RS7	RU4	Considerar en el diseño del modelo de datos que los valores de las variables de medición puedan tener distinto tipo (date, float, string)
RS8	RU5	Diseñar e implementar *dashboard* donde se despliegue un mapa, en el cual se muestran las geometrías cargadas en un contexto geográfico.
RS9	RU5	Implementar capa de servicios REST que comunique la aplicación cliente con las geometrías almacenadas en la base de datos, entregándolas en formato acorde para su despliegue.
RS10	RU6	Implementar módulo en la capa de cliente que consulte servicio REST de consultas al cubo OLAP, con filtros pertinentes, y que utilícelos datos que recibe como respuesta para generar gráficos.
RS11	RU6	Considerar en el diseño del *dashboard* un módulo que despliegue gráficos de los valores de las variables medidas en cada zona de medición.
RS12	RU7	Considerar en el diseño del *dashboard* un módulo que muestre datos asociados a cada uno de las zonas de medición.
RS13	RU7	Implementar capa de servicios REST que comunique la aplicación cliente con los datos asociados a las geometrías almacenadas en la base de datos, entregándolos en formato acorde para su despliegue.
RS14	RU8	Considerar en el modelo de datos el almacenamiento de bandas de control asociadas a las variables.
RS15	RU8	Configurar un *trigger* en la aplicación que se dispare al momento de recibir un dato que se escapa de las bandas de control.

RS16	RU8	Implementar módulo de envío de correos electrónicos para envío de alertas al momento de disparo de *trigger* de requisito RS15.
RS17	RU9	Modularizar la implementación de la aplicación, seguir buenas prácticas con el objetivo de obtener un resultado mantenible y escalable.

Tabla 3.2: Requisitos de software asociados a los principales requisitos de usuario.

Con el fin de ordenar la relación existente entre los requisitos de usuario y los de software, se presenta a continuación una matriz de trazado de estos.

RS/RU	**RU1**	**RU2**	**RU3**	**RU4**	**RU5**	**RU6**	**RU7**	**RU8**	**RU9**
RS1	x								
RS2		x							
RS3			x						
RS4			x						
RS5			x						
RS6			x			x			
RS7				x					
RS8					x				
RS9					x				
RS10						x			
RS11						x			
RS12							x		
RS13							x		
RS14								x	
RS15								x	
RS16								x	
RS17									x

Tabla 3.3: Matriz de trazado

Capítulo 4: Diseño de la solución

En el presente capítulo se describe en detalle la arquitectura de la solución, así como la tecnología utilizada para la implementación. También se describe en detalle el diseño del modelo de datos utilizando un modelo relacional y como parte de este es luego transformado a un modelo estrella para su manejo como Data Mart.

4.1 Arquitectura de la solución

La arquitectura de esta aplicación está inspirada en el patrón modelo-vista-controlador (MVC), ya que al ser una aplicación Web obtendrá todos los beneficios de modularidad que la arquitectura puede ofrecer, tales como el manejo desacoplado de los datos y las interacciones entre el servidor de base de datos y las vistas. Para este propósito se utilizó el framework ASP.NET, el cual está basado en el lenguaje C#. Se escogió este framework debido a restricciones impuestas por la empresa donde se desarrolla la aplicación. Todo esto fue desarrollado en el IDE Visual Studio 2015, que es una herramienta de Microsoft especialmente hecha para desarrollar aplicaciones Web con el framework ASP.NET.

Para la capa de datos la decisión fue optar por un modelo relacional, ya que como se iba a usar datos geoespaciales, los motores de bases de datos relacionales ya vienen con esas implementaciones lo cual facilita el diseño del modelo de datos. Para esto se manejaban alternativas de motores de base de datos como PostgreSQL y SQL Server, la primera necesitaba una extensión para soportar datos geoespaciales (PostGIS), mientras que la segunda las incluye por defecto. Además como la empresa tiene licencias de productos Microsoft, pudiendo aprovechar así todas las características que entrega SQL Server (como SQL Server Management Studio), se decidió utilizar esta última opción como herramienta para el desarrollo de la capa de datos.

Para las vistas se utilizó la herramienta que provee el framework ASP.NET Razor, que permite (a través de una sintaxis particular) crear vistas HTML dinámicas con datos provenientes de los controladores que se comunican con éstas. Además se utilizó Javascript para el manejo dinámico de las vistas y la integración con el visor de mapas. Para esta última parte se utilizó una librería para Javascript de ArcGIS, la que permite desplegar mapas, así como mostrar las capas de geometrías ingresadas y desplegar infoWindows[2] con la información requerida en cada una de ellas.

Para el acceso de datos se implementó, como una capa de servicios, un módulo extra que comunica las vistas con los datos geográficos y los datos de mediciones almacenados en la base de datos.

[2] Ventana emergente que muestra contenido (generalmente, texto o imágenes) que aparece sobre el mapa, en una ubicación determinada.

Estos últimos son consultados de manera asíncrona desde el cliente y permiten una interacción más fluida entre el usuario final y la aplicación. En la figura 4.1 se muestra un diagrama de la arquitectura completa del sistema.

Para la publicación de las geometrías y luego poder consumirlas y mostrarlas adecuadamente en la vista se publicaron tres servicios de mapas con la tecnología de ArcGIS Server, a partir de la tabla *Places* que es la que contiene las geometrías. Cada uno de los servicios representa a los tres tipos de geometrías soportado por la aplicación: Punto, Línea y Polígono.

Finalmente se implementó una solución de Inteligencia de negocios usando la herramienta de Microsoft Analysis Services y esta se integró con el resto de la solución.

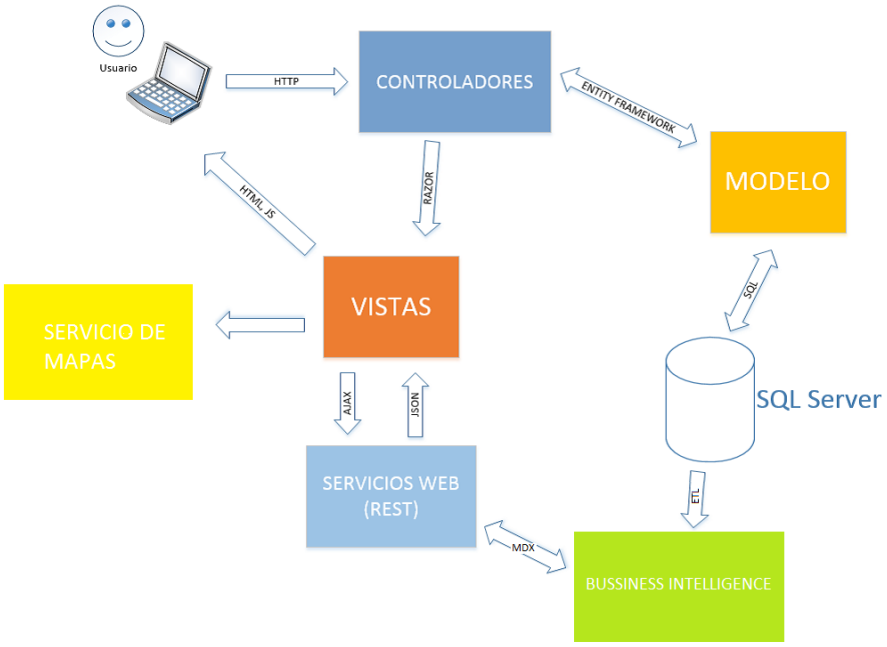

Figura 4.1: Diagrama de arquitectura del sistema.

4.2 Modelo de Datos

El modelo diseñado en este trabajo cubre los requerimientos de usuario más importantes mencionados en el capítulo anterior, al ser una aplicación piloto no se han considerados características necesarias para una puesta en producción de cualquier sistema de información como son seguridad, perfiles de usuario, logs del sistema, etc. Características poco relevantes para los alcances de esta memoria.

Para cubrir los requerimientos se necesitó crear un modelo desde el principio. Desde el inicio se sabe que los valores de las variables de medición son la entidad más importante y alrededor de estas deberían girar todas las componentes y otras entidades del modelo.

Primero se necesitan agrupar estas mediciones ya que muchas de ellas provienen de un mismo dispositivo de medición, los que se encuentran emplazados geográficamente en lugares determinados y están constantemente tomando datos. Estos dispositivos a su vez también deben poder ser agrupados ya que no es un único dispositivo el que mide cierta variable si no que son muchos dispositivos midiendo en distintas zonas, por ejemplo a lo largo de un río. Con lo anterior se pueden obtener dos nuevas entidades: los dispositivos de medición y el conjunto de estos dispositivos. Ejemplo de estos últimos son los dispositivos están geográficamente emplazados es natural que en la entidad que lo represente se almacene este lugar, por lo que la entidad fue nombrada como Lugar.

Por su parte, el conjunto de dispositivos de medición fue nombrado como Proyecto debido a que cuando un usuario quiere cargar nuevas mediciones este es un conjunto totalmente nuevo de datos geográficos y de medición, lo que podrían no tener relación alguna con datos cargados anteriormente por el mismo usuario u otro, por lo que es natural considerar estos conjuntos de datos como distintos proyectos independientes, y de esta forma se logra modularidad en el sistema, lo que es útil para cumplir el requerimiento de software RU17, con el fin de poder hacer el sistema escalable.

Ahora por parte de las variables se necesita guardar diferentes tipos de variables, cada una de estas con su propio nombre, unidad y características. Para esto se creó la entidad Variable que almacene estos datos y que además contenga el tipo de dato, ya que como se vio en los requerimientos este dato podría ser de distinto tipo (fecha, valor numérico o incluso cadena de caracteres).

El valor del dato de cada medición también debe ser almacenado en una entidad definida, como se dijo al principio esta es la entidad más importante: será la tabla más masiva en cuanto a registros por lo que para trabajar con ella se deben idear estrategias más allá del modelo de datos relacional. En la figura 4.2 se muestra un esquema del diagrama entidad-relación para el modelo de datos definido anteriormente, se muestran sólo los atributos más relevantes de algunas entidades:

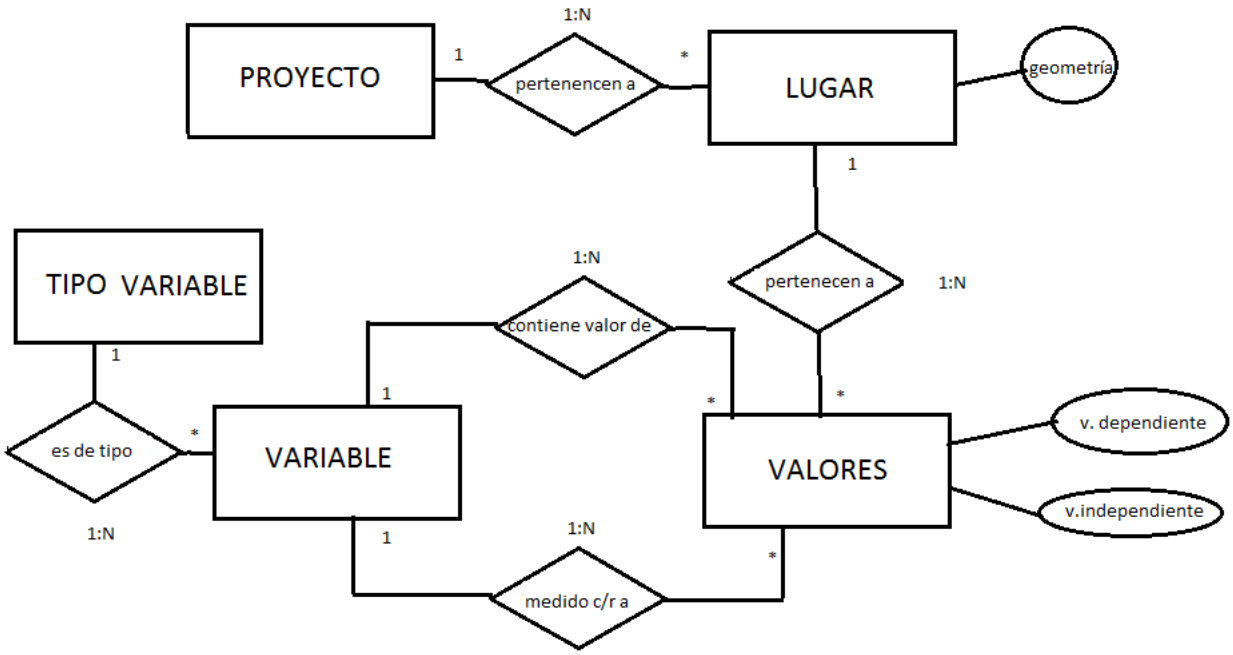

Figura 4.2: Diagrama de las principales entidades y relaciones entre ellas.

Luego de la creación del diagrama Entidad-Relación que modela los actores dentro del proyecto se procedió a crear las tablas en sistema de base de datos relacional, agregando nuevas tablas para datos útiles como son el sistema de referencia en que se representarán las geometrías de cada lugar, una tabla para definir el tipo de geometría que se agregarán a cada proyecto y una tabla con las restricciones de cada variable ingresada al sistema (implementación de bandas de control). En la figura 4.3 se puede ver un diagrama del modelo de datos resultante.

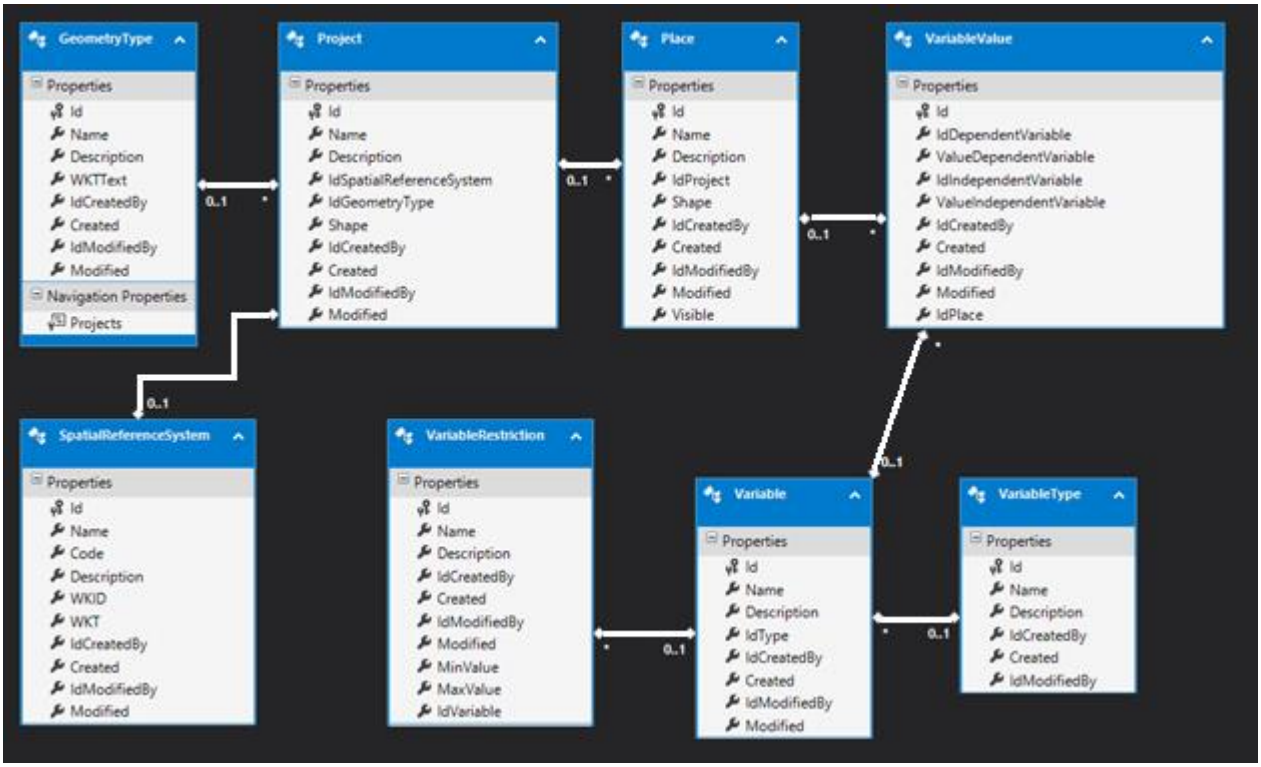

Figura 4.3: Diagrama del modelo de datos

A continuación se procede a describir las tablas del modelo de datos relacional implementado:

- **Projects**: Cuando se quiere ingresar un nuevo conjunto de mediciones al sistema es en esta tabla donde se empieza, creando un nuevo proyecto de medición, el que tendrá relacionado los sectores donde son tomadas las medidas (es decir el lugar geográfico donde se emplaza la estación que mide las variables), así como valores necesarios como el usuario que está creando el proyecto y la fecha de creación.
- **Places**: Tabla donde se almacenan los sectores de medición cargados al sistema mediante un archivo kml. Estos archivos serán vistos en más detalle en las siguientes secciones, pero básicamente contienen los datos geográficos de los lugares donde se encuentran las estaciones de medición, así como un nombre para identificarla. Esta tabla es una de las más importantes dentro del proyecto ya que contiene el campo de tipo geography, el que es usado para guardar los datos geográficos del lugar, que luego deben ser visualizados en el mapa desplegado en el *dashboard* del usuario final.
- **VariableTypes**: Esta tabla cubre el requerimiento de que el tipo de las variables sea dinámico. Indica si la variable es de tipo tiempo (date), numérico (float) o cadena de caracteres (string).
- **Variables**: Tabla usada para guardar las variables que serán ingresadas en cada proyecto. Aquí es donde se decide qué es lo que se medirá: caudal de agua, temperatura, tiempo,

concentración de algún compuesto químico en el aire, respuestas de una encuesta callejera, etc.

- **VariableRestrictions:** Esta tabla se encarga de definir las bandas de control definidas para cada una de las variables, exigidas como requerimientos.
- **VariableValues**: Es la tabla que guarda el dato de cada medición. Está relacionada dos veces con la tabla *Variables* ya que una indica la variable medida y la otra con respecto a qué fue medida. Cabe destacar una característica especial de esta tabla: los valores que guarda de los datos siempre los hacen formato string (nvarchar para SQL Server), por el hecho de que se necesitan guardar fechas, valores numéricos y cadenas de caracteres por igual. Para su posterior manipulación de los datos se usan las tablas *Variables* y *VariablesTypes* para averiguar el real tipo del valor almacenado y hacer la conversión pertinente.

4.3 Modelo Estrella y Data Mart

Con el fin de facilitar el trabajo de generar el cubo OLAP se generó un nuevo modelo de datos, siguiendo una lógica distinta al modelo de datos original que se usó antes. El cubo sigue la misma lógica que el modelo estrella, es decir se compone de una o más *tablas de hechos* con sus correspondientes *tablas de dimensiones*. Tomando esto en cuenta el nuevo modelo que se generó fue un Data Mart con los datos de las tablas involucradas, siguiendo el modelo estrella que se detallará a continuación.

4.3.1 Tabla de Hechos

Los datos a analizar se encuentran en la tabla *VariableValues* del modelo de datos, por lo que este conforma nuestra principal tabla de hechos y fue nombrada *FactVariable*. A este se agregó un atributo por cada dimensión necesaria y luego se procedió a crear las tablas de dimensiones asociadas a cada uno de estos atributos a través de una llave foránea.

4.3.2 Tablas de Dimensiones

Cada tabla de dimensión es una variable por la cual luego se quieren filtrar los valores de medición. Para este fin, cada uno de los tipos de valores de variables independientes fueron separados en tablas de dimensión, obteniendo así tres dimensiones:

- *DimTime*: Representa la dimensión temporal de la tabla de hechos, esta dimensión está separada en jerarquías de año, mes, día, hora, minuto y segundo. A su vez algunos niveles de esta jerarquía fueron agregados en distintos formatos: se tiene el día como número en una semana (1 al 7), como texto (Lunes, Martes, Miércoles, etc.) y como número en un mes (1 al 31), el mes como texto (Enero, Febrero, Marzo, etc.) y como número en un año (1 al 12), además de tener más de una categoría en un valor, como es el tiempo (hora, minuto, segundo) y la fecha (día, mes, año). Toda esta categorización y jerarquización se hizo con el objetivo de dar variadas opciones al usuario final de cómo quiere visualizar sus datos.

- *DimNumber*: Esta dimensión representa el valor numérico de la variable, la categorización que se hizo en esta dimensión para dar opciones de visualización al usuario fue separar el intervalo que generan las bandas de control definidas para estas variables en 5, 10 y 20[3] sub intervalos generando así clases donde caen cada uno de los valores y poder así graficar histogramas u otro tipo de gráficos.
- *DimText*: La dimensión de texto es la que representa los valores de cadenas de caracteres. Esta dimensión no tiene jerarquización alguna ya que no fue usada en la prueba piloto, solo se agregó por completitud.

Cabe destacar que cada registro de la tabla de hechos tiene asociadas las tres dimensiones antes mencionadas, pero solo puede estar vinculada a una de ellas por lo que siempre se cumplirá que dado un registro que tiene una de estas tres dimensiones con valor distinto de NULL entonces las otras dos tendrán asociado el valor NULL para ese registro.
Otras dimensiones que también se consideraron:

- *DimPlace:* Dimensión asociada a la tabla *Places* y que es usada para filtrar las mediciones por el lugar en que fueron medidas. Este filtro es el más importante ya que es el primero que se aplica y que reduce considerablemente la cantidad de registros a consultar.
- *DimVariable*: Dimensión asociada al tipo de variable independiente del dato de medición.
- *DimProject*: Dimensión asociada a *DimPlace*, debido a que la visualización se hace por cada proyecto esta dimensión es útil para filtrar primero por proyectos y disminuir la cantidad de lugares a consultar.
- *DimVariableType*: Dimensión usada para saber de qué tipo es el dato almacenado en la tabla de hechos.
- *DimCompany*: Dimensión asociada a la compañía que pertenece el usuario que cargó los datos al sistema, no se usa en el piloto pero se agregó por completitud.

En la figura 4.4 se muestra un diagrama del modelo estrella resultante.

[3] La cantidad de clases ideal para un histograma depende de la cantidad de datos que se están analizando, en este contexto es inviable calcular las clases cada vez que se ingresen datos nuevos, por lo que esto se define sólo a manera de prueba piloto.

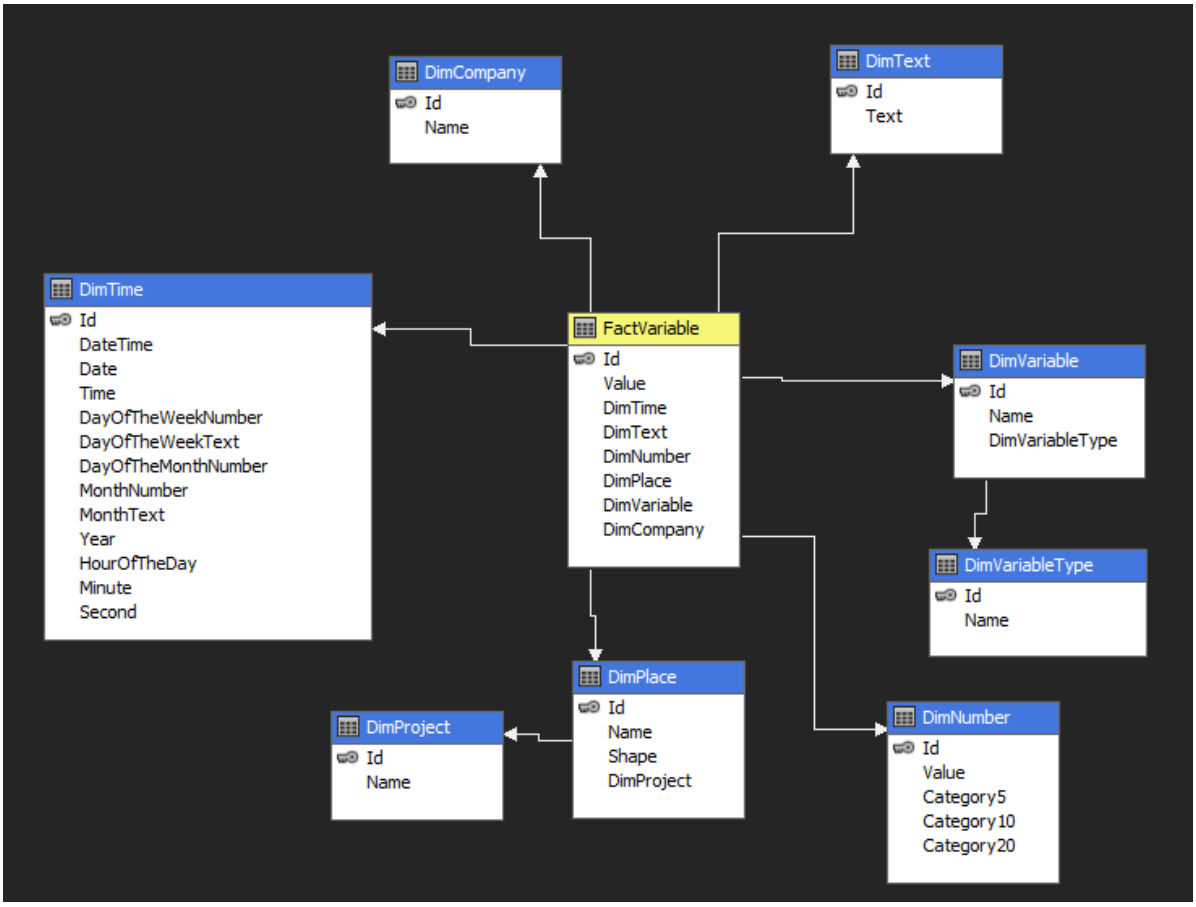

Figura 4.4: Diagrama de modelo estrella. En amarillo la tabla de hechos y en azul sus dimensiones.

Capítulo 5: Implementación de la solución

En este capítulo se detalla el trabajo de implementación de la solución planteada, el desarrollo de todas las componentes que interactúan entre sí. Además se muestran figuras de las vistas de la aplicación con el fin de explicar de mejor manera el funcionamiento de la aplicación web.

5.1 Data Mart

Luego de haber generado el nuevo modelo de datos siguiendo un modelo estrella se empezó a tratar a este como Data Mart definiendo un proceso ETL que lo alimentara a partir de los datos guardados en la base de datos relacional.

Se estudiaron dos opciones para implementar este proceso de ETL: el primero fue hacer inserciones en las tablas del Data Mart desde la aplicación, cada vez que se insertaran, modificaran o eliminaran registros de las tablas asociadas, la segunda opción fue configurando un disparador (*trigger*) en la base de datos, es decir definiendo procedimientos almacenados que cada vez que se realizara una inserción, modificación o eliminación en las tablas asociadas fueran ejecutados y se encargaran de hacer la extracción, transformación y carga al Data Mart de los datos involucrados en el evento disparado.

Finalmente se decidió optar por la opción de los disparadores y procesamientos almacenados por la razón de que era más complejo identificar dentro de la aplicación cada uno de los lugares donde se realizaran cambios a la base de datos para activar estas actualizaciones y por lo tanto esta opción complejizaría innecesariamente el código. Por su lado la opción elegida no tiene este problema ya que funciona a nivel de base de datos, cualquier cambio hecho en ella dispara los procesamientos adecuados para llevar a cabo la ETL de forma adecuada. Cabe destacar que esta solución funciona no solo para cambios que se hagan dentro de la aplicación si no que cualquier cambio hecho se verá reflejado en su ETL correspondiente, lo que es conveniente cuando se hagan cambios directamente en la base de datos vía SQL, sin la aplicación como intermediario.

5.2 Analysis Services

Se creó un proyecto con la herramienta Microsoft Analysis Services a partir del Data Mart ya creado. A partir de este punto si el modelo que se usa para crear el cubo es lo suficientemente bueno en términos de que coinciden las tablas de hecho y las de dimensiones con un modelo estrella el trabajo es mínimo gracias al asistente de creación de la herramienta:

- Se selecciona un *Data Source* entregando datos del servidor de SQL Server donde se encuentran los datos, además de credenciales de conexión. Con esto se genera un string de conexión que es usado por la aplicación para conectarse a los datos.

- Se selecciona las *Data Source Views* que serían las tablas que participarán en la creación del cubo, la tabla de hechos y las dimensiones asociadas.
- En el siguiente paso se crea el cubo, seleccionando las tablas agregadas como *Views* en el paso anterior.
- Finalmente se seleccionan las dimensiones que pertenecerán al cubo y las jerarquías que estas definen en sus atributos.

En términos generales si el modelo que se usa para generar el cubo es uno creado especialmente para ello, con todas las características necesarias la herramienta de Microsoft facilita enormemente el trabajo.

El cubo deber ser procesado cada vez que se agreguen nuevos datos a la fuente desde donde los consume (Data Mart). Debido a esto se configuró una tarea programada en el servidor de base de datos que realiza este procesamiento en intervalos de 5 minutos con el fin de tener siempre los datos más recientes accesibles desde la aplicación.

5.3 Módulos de la solución

La solución implementada fue siempre modularizada lo más posible con el fin de facilitar la mantención y la futura realización de nuevas versiones y mejoras. Bajo esta premisa la solución de separó en cinco módulos interdependientes: El módulo de la capa de datos que ya fue visto en el capítulo anterior, una aplicación Web que sigue la arquitectura Modelo-Vista-Controlador, una capa de servicios REST que implementa una API para consultas desde el cliente en la aplicación web a los datos, un proyecto de Analysis Services que constituye la solución de inteligencia de negocios y finalmente una aplicación que usa el lenguaje de consulta MDX implementando métodos que faciliten la consulta el cubo OLAP al entregar un objeto más simple de manipular y serializable.

5.3.1 Aplicación de consultas al cubo

En este módulo se realizan las consultas al cubo OLAP a través del lenguaje de consulta MDX, utilizando funcionalidades provistas por el paquete de *Microsoft Analysis Services* para el framework ASP.NET. La motivación de implementar este módulo viene dada debido a que este paquete trabaja con objetos llamados *CellSet* que representan resultados multidimensionales retornados al hacer consultas a bases de datos multidimensionales. Estos objetos son complejos de manipular por lo que este módulo se encarga de hacer la transformación a objeto *DataSet* que es más simple de usar y además serializable.

Por lo tanto este módulo consiste en dos partes, la primera es la que consulta extracción de los datos desde cubo, siendo la segunda la que transforma estos datos obtenidos en resultados más manipulables, retornándolos para su uso en la capa de servicios REST. Para este fin se implementó

una representación del cubo como objeto, con las propiedades de su nombre y su string de conexión, además un método que realiza consultas MDX sobre el cubo y otro que transforma el resultado obtenido en objeto serializable.

5.3.2 Capa de servicios REST

En esta aplicación se implementó un método que es consultado desde las vistas de la aplicación web, el que recibe como parámetros la medida que se quiere consultar, las dimensiones por las que se quiere "cortar" el cubo y filtros. Este servicio es consultado a momento de hacer click en alguna geometría desplegada en el mapa.

Generalmente la medida que se consulta es el valor de la variable de medición y la dimensión es el mes del año como número y el filtro sería por el Id del lugar que se está consultando.

5.3.3 Aplicación Web

La aplicación web fue implementada siguiendo la arquitectura Modelo-Vista-Controlador. Del modelo ya se ha hablado en el anterior capítulo, ahora se revisarán los controladores más importantes con sus respectivas vistas.

Home

Es el controlador que se encarga de la pantalla de inicio de la aplicación, donde muestras accesos a la creación de proyectos, de variables e ingreso al *dashboard* interactivo.

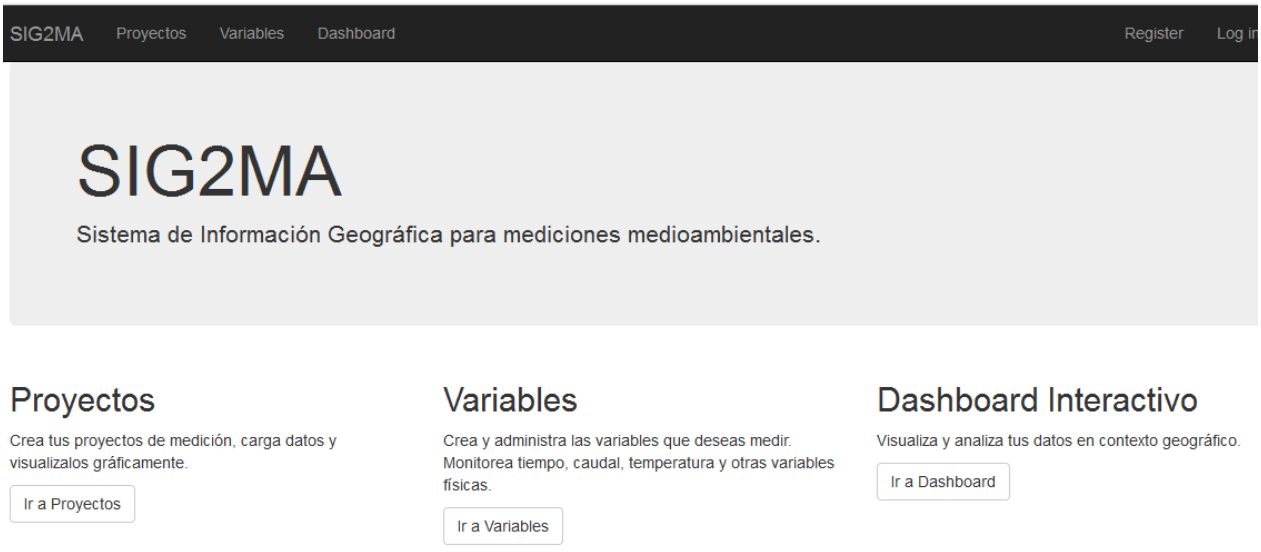

Figura 5.1: Vista de la pantalla de inicio de la aplicación.

Projects

El controlador de proyectos es el que se encarga de la creación, consulta, edición y eliminación de proyectos de medición. Al crear un nuevo proyecto se deben seleccionar dos cosas importantes:

una es el sistema de referencia en que se ingresarán las geometrías que representan los lugares de medición, el otro es el tipo de geometría que se quiere agregar a ese proyecto. El sistema de referencia es uno de los atributos más importantes ya que de este depende que las geometrías se desplieguen de manera correcta en el contexto geográfico del mapa, si el sistema de referencia no coincide con el del mapa base que se muestra se debe hacer una conversión (más sobre sistemas de referencia y proyección en el capítulo de Anexos).

Figura 5.2: Lista de proyectos ingresados al sistema

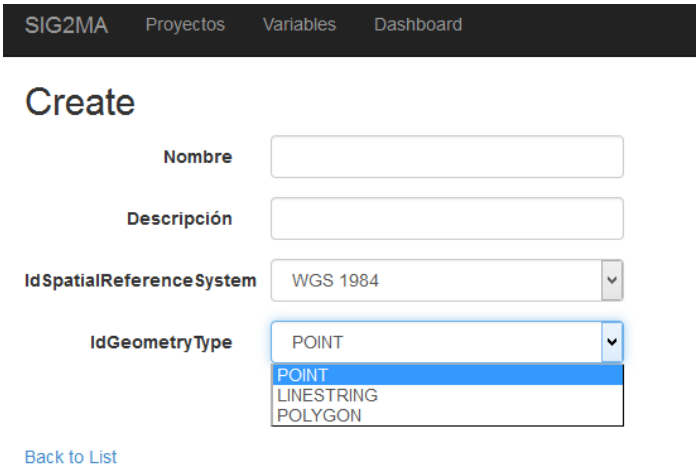

Figura 5.3: Formulario de creación de nuevos proyectos.

Figura 5.4: Detalles de un proyecto

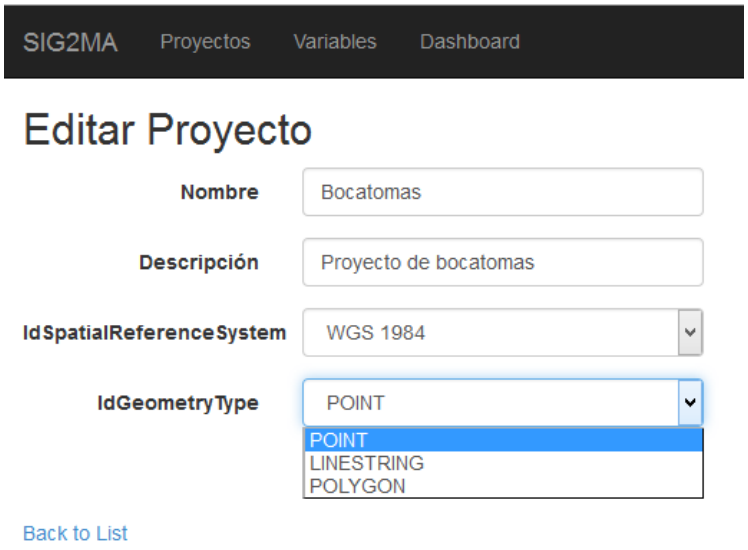

Figura 5.5: Formulario de edición de un proyecto.

Además, en el controlador de proyectos se implementó la funcionalidad de cargar lugares al sistema asociados a un proyecto específico. Para esto se creó un módulo que recibiera un archivo en formato kml, lo procesara y a partir de este generara los registros de geometría que se guardarían en la base de datos.

Para lograr lo anterior el autor se tuvo que familiarizar con dos estándares de OGC: por un lado el formato kml, entender los esquemas XSD que definen estos archivos en formato xml para luego poder parsearlos y extraer la información, y por otro lado transformar esta información obtenida desde el archivo kml a un formato reconocible por el objeto que representa el registro en la base de datos, en el caso de SQL Server este es DbGeometry. Estos objetos se pueden crear a partir de la representación alfanumérica de la geometría conocida como Well-Known Text o WKT, las que se generan a partir de una gramática.

Figura 5.6: Archivo KML con las geometrías a ingresar en el sistema.

Por lo tanto, el proceso de cargar los registros de lugares obtenidos a partir de un archivo KML a la tabla de la base de datos *Places* se dividió en dos pasos:

1. Leer el archivo kml como archivo xml. Debido a que su estructura es conocida de antemano se puede parsear y extraer un arreglo de figuras con arreglo de coordenadas representado todos sus vértices.
2. Usar la lista de vértices obtenidas en el paso anterior para generar un string acorde a la gramática de WKT, usando este string y el sistema de referencia asociado al proyecto se pueden crear las geometrías y almacenarlas en la base de datos.

Figura 5.7: En rojo el botón para cargar lugares a un proyecto a partir de archivo KML.

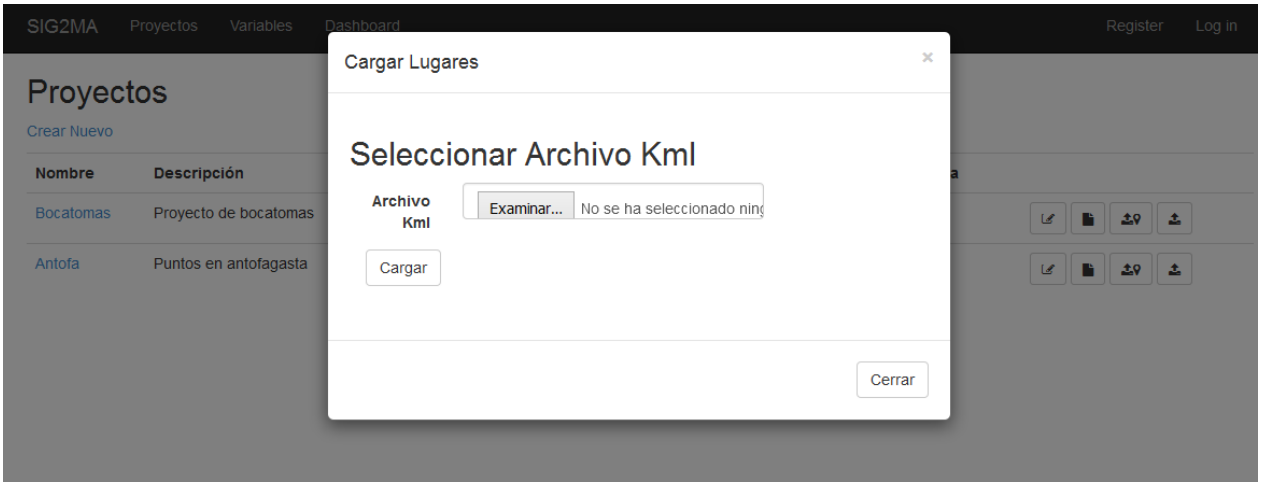

Figura 5.8: Al hacer click en el botón se despliega un modal con el formulario para cargar las geometrías.

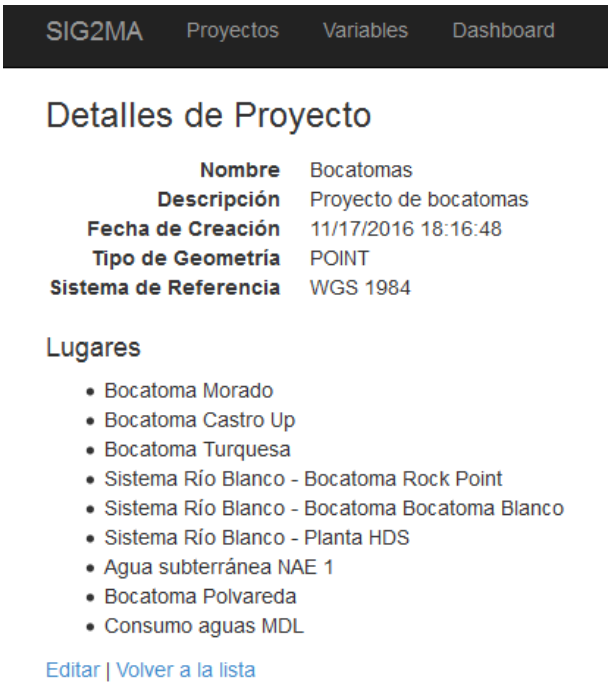

Figura 5.9: Luego de cargar el archivo KML se mostrarán las geometrías cargadas en los detalles del proyecto.

Variables

Este controlador es el que se encarga de la creación, edición, consulta y eliminación de las variables en el sistema. Al crear una variable se debe elegir qué tipo de valores almacenará, así como definir sus restricciones (banda de control).

Figura 5.10: Lista de las variables ingresadas al sistema.

Figura 5.11: Formulario de creación de nuevas variables.

LoadData

Controlador creado con el fin de tener separada la carga de los datos de medición. Este controlador se encarga de dos cosas: Recibe un archivo xls con los datos a cargar al sistema, volcando estos en la base de datos, además genera un archivo xls template con el objetivo de darle al usuario una plantilla donde poner sus datos y el sistema pueda reconocerlos sin confusiones y ni posibles errores.

Figura 5.12: En rojo el botón para cargar datos a un proyecto en particular.

Figura 5.13: Formulario para cargar los datos. En rojo el botón para descargar plantilla que se rellena con datos.

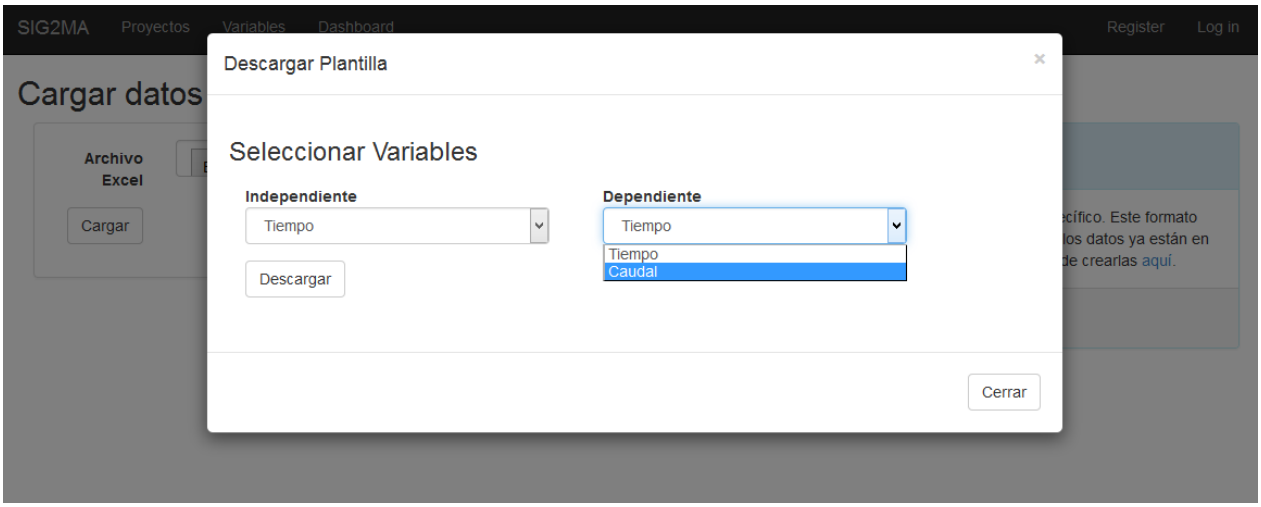

Figura 5.14: Se abre un modal donde se seleccionan las variables a incluir en la plantilla.

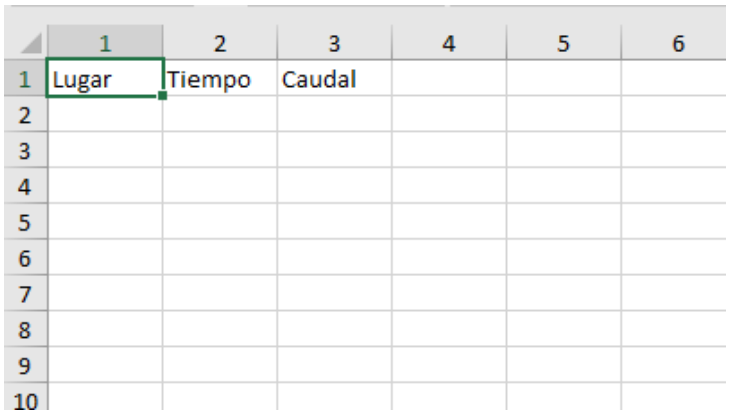

Figura 5.15: Plantilla descargada. Hoja de Excel simple pero entrega patrón a seguir y es eficiente.

Dashboard

Es el controlador que se encarga de mostrar el *dashboard* interactivo. La vista de este controlador trabaja con JavaScript: Por un lado usa la API de ArcGIS para desplegar el mapa y las geometrías, además de usar la API de Google Charts para mostrar los datos consultados en forma de gráficos.

Las geometrías que se muestran aquí son consultadas a un servicio de mapas publicado en ArcGIS Server a partir de la tabla *Places*. Primero se generaron tres vistas en la base de datos, una por cada tipo de geometría, y luego estas vistas se usaron para publicarlas como tres capas independientes, las que se cargarían en el *dashboard* usando la API de ArcGIS.

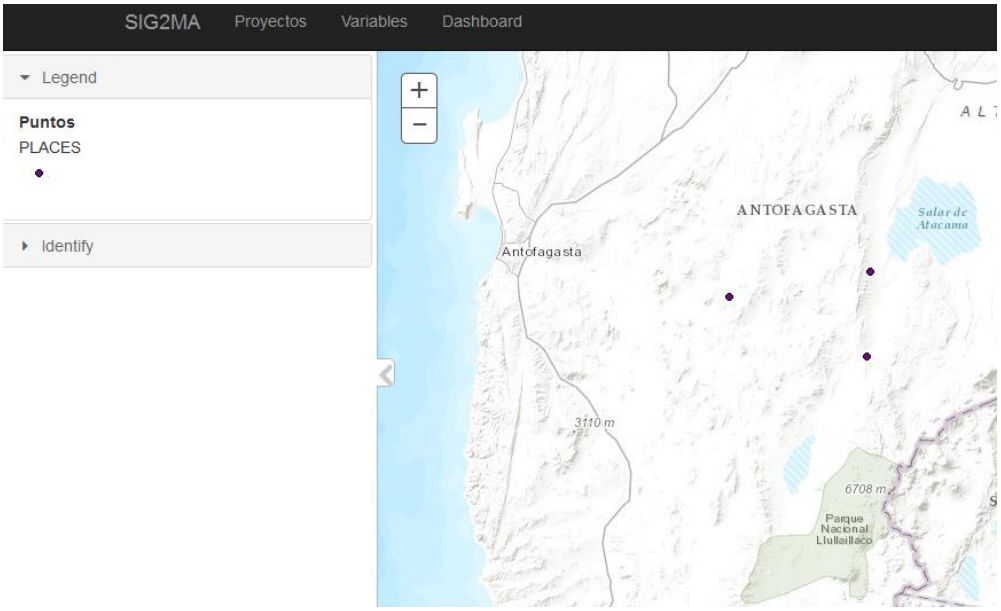

Figura 5.16: Dashboard con polígonos (puntos) desplegados en mapa.

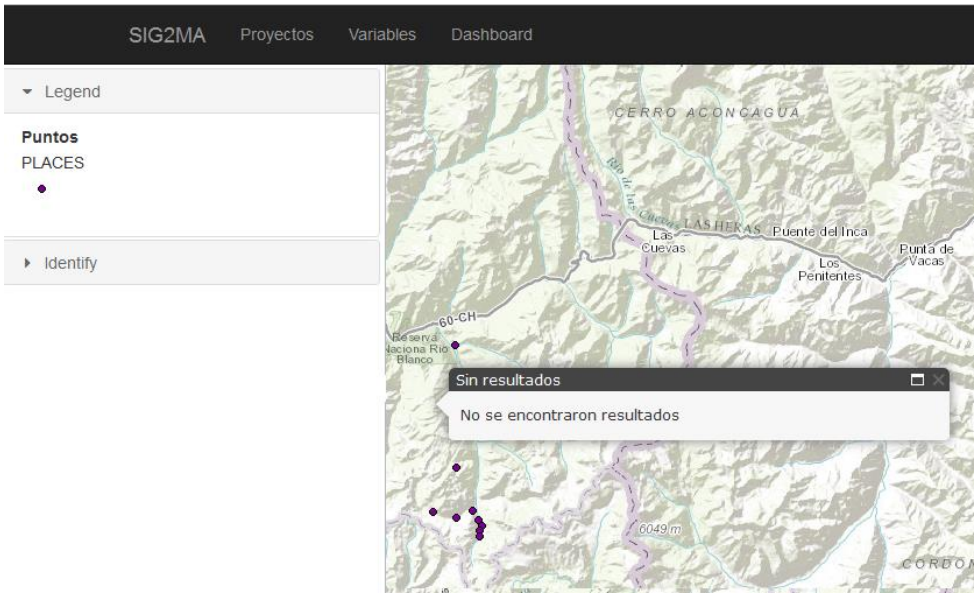

Figura 5.17: Consulta por datos a un lugar donde no hay mediciones, no se encuentran resultados.

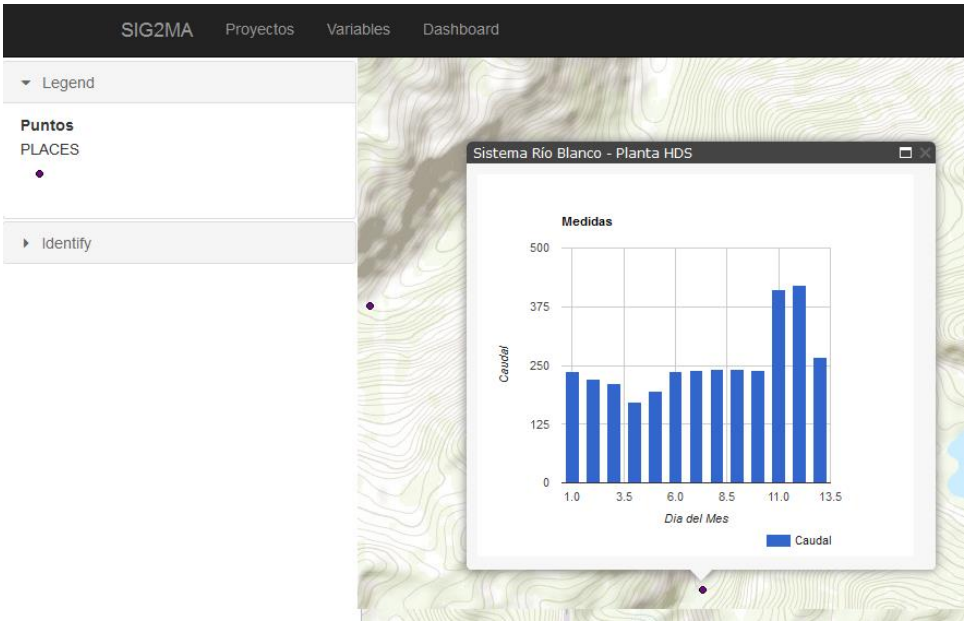

Figura 5.18: Se hace click en punto desplegado en mapa y se muestran resultados de sus medidas ingresadas.

Capítulo 6: Validación de la solución

Con el fin de obtener feedback de parte de potenciales usuarios y además validar que cada uno de los requerimientos de usuario fueron satisfechos se sometió al sistema un plan de prueba, el que consistió en seleccionar una cantidad de usuarios pertenecientes a distintas áreas vinculadas al tema que se trata, entregarles un set de instrucciones a seguir junto con el documento de requerimientos de usuario, además de la libertad de manipular el sistema como quisieran durante un tiempo de 30 minutos.

Los usuarios que realizaron estas pruebas fueron cinco, dos del área de desarrollo de software, dos del área de geografía y un usuario común no perteneciente a ninguna de las dos áreas anteriormente mencionadas.

6.1 Plan de pruebas

A continuación se especifica el plan de pruebas, consistente en el set de instrucciones entregado a los usuarios en forma de flujo de usuario, el cuestionario que debieron responder cada uno de estos y los resultados obtenidos.

6.1.1 Flujo de usuario

El set de instrucciones dado a los usuarios para probar la aplicación se resume en el siguiente flujo de usuario:

1. El usuario ingresa al sistema y crea un nuevo Proyecto de Medición.

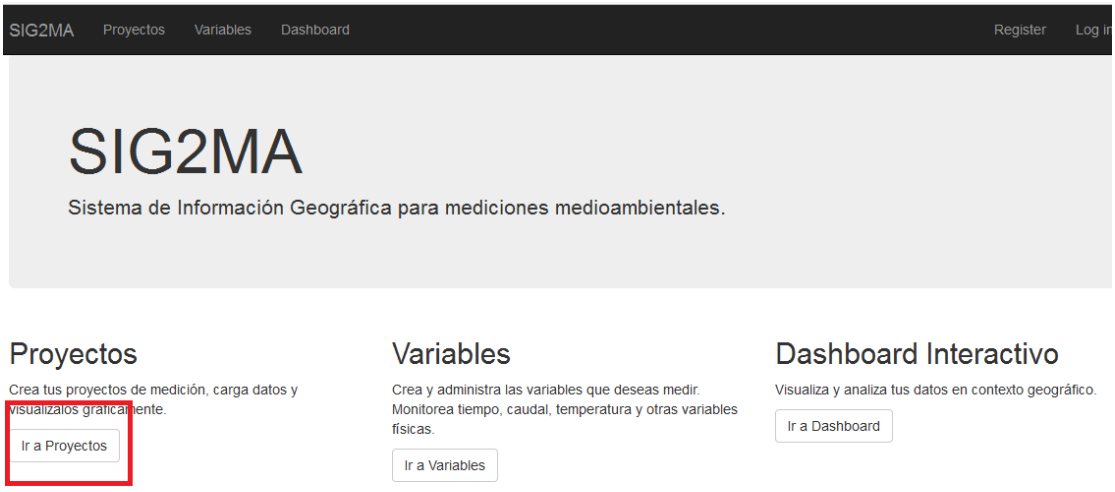

Figura 6.1: Inicio de la aplicación. En rojo el acceso a crear proyectos.

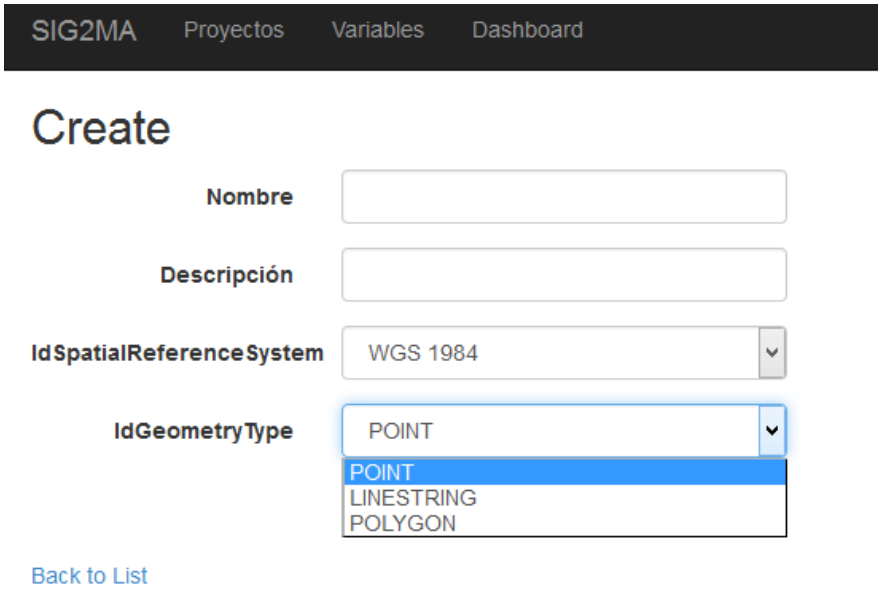

Figura 6.2: Formulario de creación de Proyectos.

2. En su Proyecto de Medición debe ingresar las Variables de Medición que utilizará (Nombre, Unidad de medida, Bandas de Control, etc.).
3. Luego de tener las Variables de Medición, se necesita establecer las Variables Independientes. Con respecto a éstas se tomarán las mediciones. Para cada Variable Independiente se ingresa el Nombre y la Unidad de medida.

Figura 6.3: Inicio de la aplicación. En rojo el acceso a crear variables.

Figura 6.4: Formulario de creación de Variables.

4. Después de tener la estructura del Proyecto de Medición, se procede a ingresar datos. Esta etapa se divide en dos fases:
    a. *Ingreso de datos geoespaciales*: Aquí se ingresan los datos espaciales de las estaciones de medición, como por ejemplo la posición en el mapa y la forma geométrica. Todo esto viene en un archivo KML.
    b. *Ingreso de datos de medición*: Para hacer un ingreso uniforme de datos de medición, el sistema provee una plantilla en formato Excel (.xls) que es generada dependiendo de la estructura del proyecto de medición creado (variables de medición y variables independientes). Dicha planilla viene lista para ser llenada con los datos y luego cargada al sistema.

Figura 6.5: Lista de proyectos del sistema. En rojo el botón para cargar lugares.

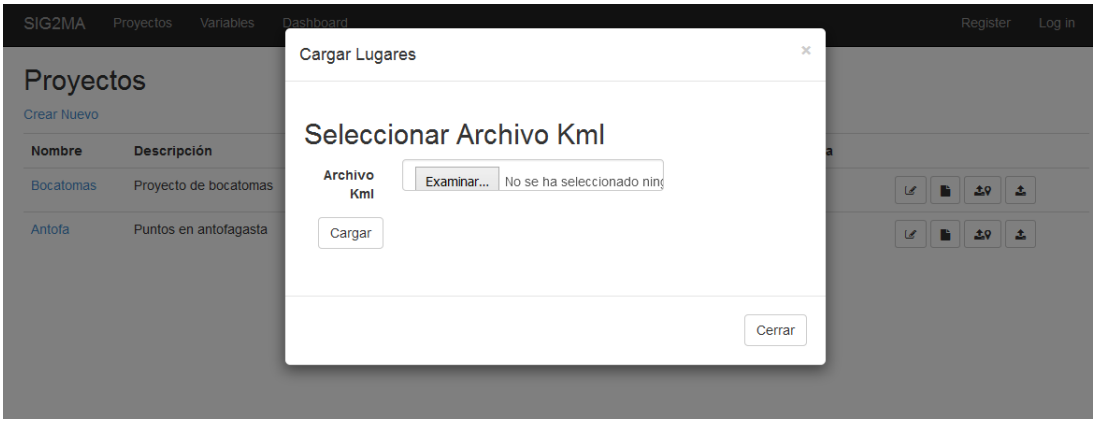

Figura 6.6: Formulario para carga de lugares vía archivo kml.

Figura 5.7: Lista de proyectos del sistema. En rojo el botón para cargar datos de medición.

Figura 6.8: Formulario para carga de datos de medición vía archivo xls.

5. Finalmente el usuario puede dirigirse al visor de mapas, cargar sus datos y visualizar en contexto geográfico sus datos geoespaciales, así como gráficamente las mediciones asociadas a esos datos.

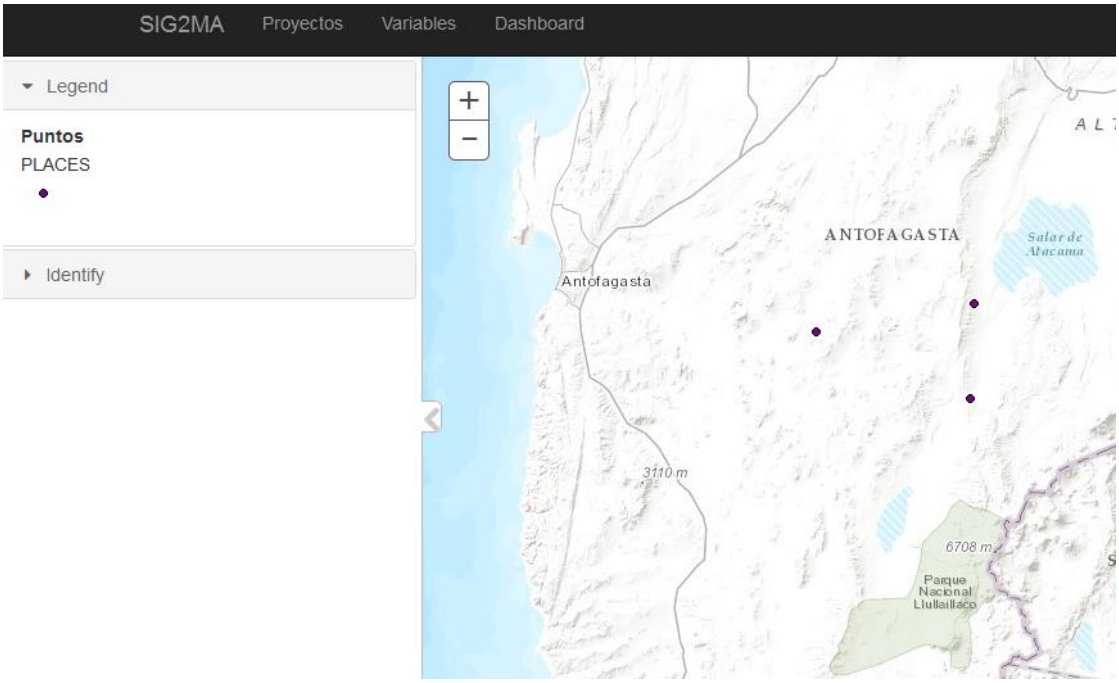

Figura 6.9: Vista de dashboard con datos de geometrías cargados.

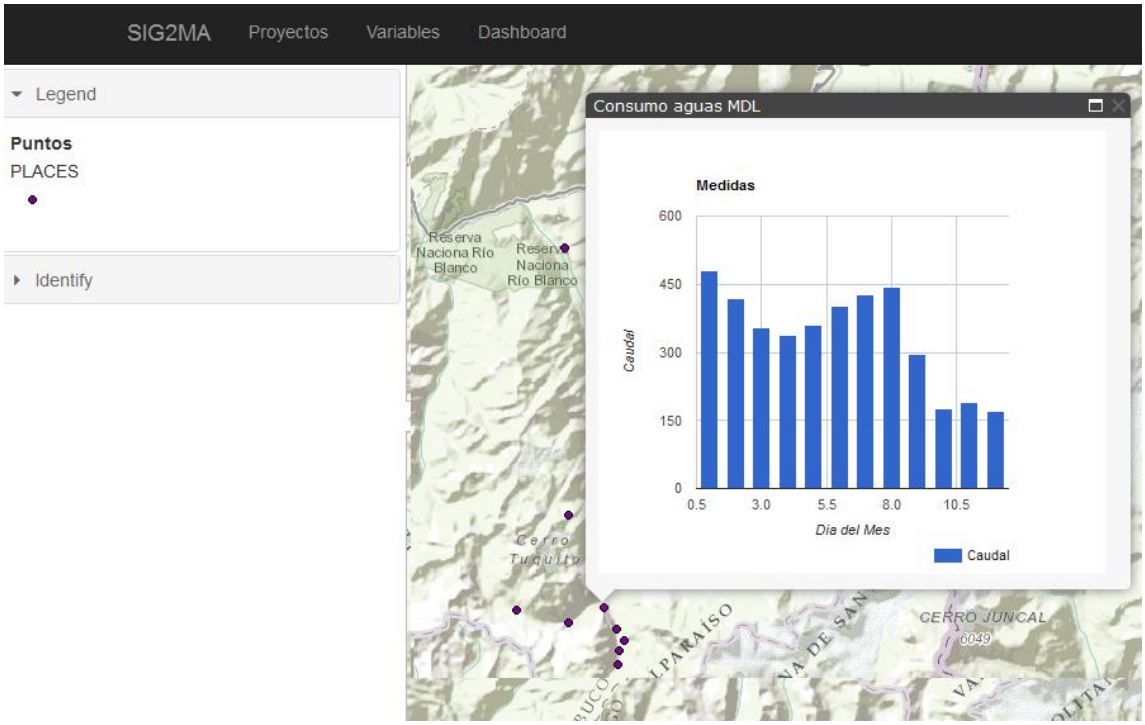

Figura 6.10: Vista en dashboard de gráficos de datos de medición.

En la figura 6.11 se presenta un esquema del flujo seguido por los usuarios.

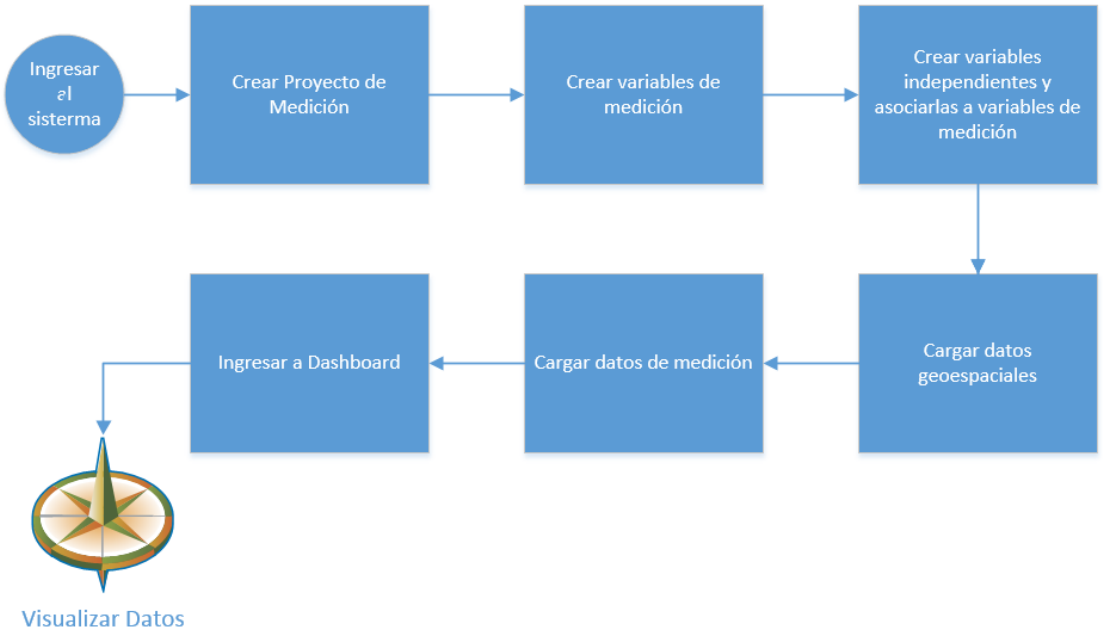

Figura 6.11: Flujo de usuario para prueba de validación.

6.1.2 Cuestionario a responder por los usuarios

A cada usuario se le entregó una tabla que debía rellenar de acuerdo a la percepción que tuvo de la aplicación después de someterse a la experiencia del plan de pruebas. Se les pidió marcar con una X en la opción que considere más adecuada:

N°	Ítem	Muy en Desacuerdo	En Desacuerdo	Indeciso	De acuerdo	Muy de acuerdo
1	Las tareas asignadas fueron fáciles de entender y realizar (general)					
2	No se entiende lo que significan los elementos de la interfaz de usuario (general)					
3	Es fácil de recordar cómo realizar una cierta actividad usando la aplicación (general)					
4	Los mensajes que entrega la aplicación son difíciles de entender (general)					
5	Ante un error en el ingreso de datos, el sistema le indica al usuario cómo corregirlo (general)					

6	No es fácil saber cómo llegar a la funcionalidad que necesito para realizar una cierta tarea.					

Tabla 5.1: Cuestionario entregado a los usuarios participantes del plan de pruebas.

Luego de completar esta tabla se le indicó al usuario escribir comentarios, sugerencias o errores encontrados con el fin de mejorar el software.

6.1.3 Resultados

A continuación se muestran los resultados obtenidos de los cinco usuarios sometidos a las pruebas.

N°	Ítem	Muy en Desacuerdo	En Desacuerdo	Indeciso	De acuerdo	Muy de acuerdo
1	Las tareas asignadas fueron fáciles de entender y realizar (general)				XX	XXX
2	No se entiende lo que significan los elementos de la interfaz de usuario (general)	XXX	X	X		
3	Es fácil de recordar cómo realizar una cierta actividad usando la aplicación (general)				X	XXXX
4	Los mensajes que entrega la aplicación son difíciles de entender (general)	XXXX	X			
5	Ante un error en el ingreso de datos, el sistema le indica al usuario cómo corregirlo (general)	XXXXX				
6	No es fácil saber cómo llegar a la funcionalidad que necesito para realizar una cierta tarea.	XXX	XX			

Los resultados de esta prueba se analizarán en el siguiente capítulo de conclusiones.

Capítulo 7: Conclusiones y trabajo a futuro

Los trabajos interdisciplinarios son sin duda un gran aporte en cuanto a experiencia intelectual, laboral y humana. De la misma forma significan un gran desafío para quienes optan por tomarlos, sobre todo si desde un principio no se tiene previsto que el trabajo a realizar cubre tantas aristas distintas y más aún que algunas se salen completamente del dominio de conocimiento adquirido durante una carrera de 6-7 años. Esto es sin duda una experiencia enriquecedora.

En cuanto a objetivos se puede decir que se cumplió el objetivo general, se logró sistematizar la compilación de los de los datos generados por las estaciones de monitoreo, además se implementó la herramienta para que el usurario pudiera gestionar sus mediciones y visualizarlas de forma agregada a través de la aplicación Web. Uno de los objetivos específicos era validar la aplicación con potenciales usuarios reales de esta, la que por temas de coordinación y disponibilidad horaria no se pudo llevar a cabo. Aun así se pudo llevar a cabo un plan de pruebas con otro tipo de usuarios, lo que de todas maneras sirve para obtener feedback y poner a prueba los límites del piloto.

Hubo requisitos de software que de un principio se plantearon de una forma pero con el avanzar del proyecto se fueron estudiando mejores opciones para cumplir los requisitos, cambiando pero con el mismo objetivo de cumplir con su requisito de usuario asociado. Un ejemplo claro de esto es el requisito RS9: *Implementar capa de servicios REST que comunique la aplicación cliente con las geometrías almacenadas en la base de datos, entregándolas en formato acorde para su despliegue,* finalmente este requisito se cubrió con la publicación de la tabla con geometrías en un servicio de mapas que luego la aplicación consumía vía JavaScript, lo que resultó ser mejor ya que se le delegó el trabajo de desplegar cada geometría a la API de ArcGIS que está diseñada para esto.

Con respecto a la validación con usuarios que se hizo se puede ver que al seguir instrucciones el uso de la aplicación se hacía más simple y luego de explicar la lógica que siguen los proyectos, variables y otras entidades se podía entender mejor el problema planteado y como la aplicación lo soluciona. Por el contrario si no se explicaba la lógica del modelo era poco intuitivo para los usuarios saber los pasos a seguir para cumplir el objetivo de la prueba.

También se pudo ver en las pruebas de validación que los usuarios notaron la falta de mensajes de error al ingresar datos erróneos, datos faltantes, archivos corruptos, etc. y el sistema no indicaba cómo solucionar los problemas ocasionados por estos errores. Si bien la aplicación logró sistematizar el proceso de compilación de los datos, cumpliendo el objetivo general de este trabajo de memoria el resultado obtenido es una versión piloto del sistema, por lo que se necesitarán nuevas funcionalidades en las nuevas versiones. Algunos de los trabajos que quedan pendientes para próximas versiones:

- Considerar la opción de hacer reingeniería sobre el modelo original e integrar el modelo estrella en este, no considerarlos como entidades separadas, esto debido a la duplicación de

datos que se genera al ejecutar procesos de ETL cada vez que se agreguen nuevos registros, por lo que en caso de poder integrar ambos modelos los procesos de ETL ya no serían necesarios y simplificarían la implementación y mantención de la aplicación.

- Agregar al sistema perfiles de usuario. En el modelo se consideraron entidades como Usuario (tabla *Users*) y Compañía (tabla *Companies*) con el objetivo de mostrar que el sistema puede ser multi-usuario e incluso se puede considerar ofrecer como servicio a distintas compañías que puedan administrar sus propios datos, otorgando permisos a sus usuarios manejando una jerarquía de perfiles o roles.
- Extender la cantidad de formatos que el sistema acepta para cargar geometrías. Estudiar la posible compatibilidad con formatos no estándares pero populares (como son los archivos shapefile de ArcGIS[4]) y la carga directa desde servicios de mapas.
- Agregar métodos de logging de acciones y seguridad a la aplicación.
- Implementar generación de reportes de las mediciones medioambientales

Una de las funcionalidades más útiles del sistema a implementar en el futuro sería esta última: la generación de reportes de los datos medioambientales, esto debido a que uno de los potenciales y principales clientes de la aplicación serían las grandes empresas que presentan proyectos al Ministerio del Medio Ambiente con el fin de reportar el impacto medio ambiental que pueden llegar a causar estos proyectos en ciertos lugares, por lo que el potencial de utilidad de la herramienta es muy grande, midiendo concentraciones de compuestos en el aire o el agua, o midiendo el caudal de ríos cuyos cauces son modificados artificialmente, los que ayudarían al estudio presentado con el fin de obtener una Resolución de Calificación Ambiental aprobada, la que exige un monitoreo constante de los resultados presentados en el proyecto, por lo que la aplicación seguiría siendo útil.

Finalmente se puede concluir que los proyectos ambiciosos no nacen de la noche a la mañana, hay un sin número de aristas a considerar antes de sacar adelante un producto apto para su comercialización, y hay oportunidades donde una idea que en un principio se ve con futuro puede que luego de aplicar la ingeniería necesaria y aterrizar la idea ámbitos más realistas esta puede llegar a ser una pérdida en todo sentido si no se trabaja de forma adecuada, e incluso considerando este último aspecto se puede llegar a la conclusión que simplemente hay ideas que no son buenas.

[4] Documentación: https://doc.arcgis.com/en/arcgis-online/reference/shapefiles.htm

8. Bibliografía

[1] Larson, Brian. Delivering Business Intelligence with Microsoft SQL Server 2012, 3rd Edition, McGraw-Hill Education, 2012.

[2] Open Geospatial Consortium (OGC). Standards: http://www.opengeospatial.org/docs/is. Último acceso: 26 de Diciembre de 2016.

[3] Tomlison, Roger. Thinking About GIS: Geographic Information System Planning for Managers, 5th Edition, ESRI Press, 2013.

[4] KML Standards: http://www.opengeospatial.org/standards/kml. Último acceso: 26 de Diciembre de 2016.

[5] KML Reference: https://developers.google.com/kml/documentation/kmlreference. Último acceso: 26 de Diciembre de 2016.

[6] J. Herring, OpenGIS Implementation Specification for Geographic Information – Simple Feature Access, 2006.

9. Anexo

9.1 Conceptos básicos de geografía

9.1.1 Geoide, elipsoide, datum

No es correcto decir que la tierra es una esfera o elipse perfecta; se define el geoide de la tierra como el conjunto de puntos sobre esta con la misma fuerza de gravedad. Con esta definición se hace más exacto el modelamiento de la tierra.

Definición 9.1: *Sea* $g: \mathbb{R}^3 \to \mathbb{R}$ *la función que a cada punto en la superficie terrestre le asigna le valor de la gravedad medida en ese punto. Se define el geoide terrestre como el conjunto:*

$$G = \{p \in \mathbb{R}^3 : g(p) = c\}$$

Donde c *es una constante.*

En términos generales, se establecen los modelos de geoides a partir del campo gravitacional y tomando los puntos más cercanos a la superficie del nivel del mar.

La Geodesia, ciencia que se ocupa de fijar la forma de la tierra, se han ocupado de diseñar diferentes técnicas y herramientas para lograr modelar de la manera más exacta posible la superficie de la Tierra, y poder así establecer un sistema de medición de posiciones sobre ella. A esto le sigue la necesidad de representar la superficie de un modelo de la tierra en tres dimensiones, en un plano. Lo primero que se hace para representar la superficie terrestre en un plano, es modelarla como un elipsoide.

Definición 9.2: *Un elipsoide es la superficie definida por la ecuación:*

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1$$

Los parámetros a*,* b *y* c *son denominados los semiejes de* x*,* y *y* z *respectivamente.*

La superficie del *geoide* es entonces aproximada con un elipsoide. Dados los semiejes de éste, y una inclinación, un elipsoide logra aproximarse mejor al geoide en la medida en que existe la mayor cantidad de puntos coincidentes entre la superficie definida por éste y por el geoide.

Definición 9.3: *Dado el geoide* $G \subset \mathbb{R}^3$ *y un elipsoide definido por:*

$$E = \left\{(x, y, z) \in \mathbb{R}^3 : \frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1\right\}$$

Se define el conjunto de puntos datum como:

$$D = G \cap E$$

La determinación del *datum* (y por lo tanto, infiriendo de la Definición 9.3, del elipsoide; en la Figura 8.1 se puede apreciar un llamado punto datum) permite establecer una referencia para un sistema de coordenadas con el cual medir la posición en el modelo de la Tierra. Sin embargo, dada la particular forma del geoide, los datum son sólo de alcance regional, lo mismo que los elipsoides de referencia: dependiendo de la zona geográfica modelarán con mejor o peor precisión la superficie de tal zona.

Figura 9.1: Punto datum

Elipsoide y datum conforman lo que se llama un Sistema de Referencia, a partir del cual se posicionan los objetos en la Tierra fijando el sistema de coordenadas (llamadas coordenadas geodésicas). Un Sistema de Referencia clásico está entonces compuesto por:

- Un elipsoide de referencia
- Un datum geodésico
- Un origen de latitudes y longitudes (que corresponden a la inclinación del elipsoide).

Uno de los más populares es el *World Geodetic System* del año 1984 (WGS 84), definido por el Departamento de Defensa de Estados Unidos, y utilizado por la mayoría de los GPS.

9.1.2 Sistema de proyección

Una vez definido el elipsoide que modela el geoide terrestre, surge la necesidad de representar la superficie de este elipsoide en un plano. Para esto debe definirse un método que mapee los puntos desde una superficie a otra, conservando, en la medida de lo posible, algunas propiedades básicas como distancias, ángulos, áreas, forma de las superficies o direcciones. Para esto se definen las proyecciones.

Definición 9.4: *Dado un elipsoide* $E \subset \mathbb{R}^3$ *, una proyección geográfica es una función* $f: E \to \mathbb{R}^2$ *que mapea los puntos del elipsoide a un plano cartesiano de dos dimensiones.*

La clasificación más común de las proyecciones es la forma en que se proyectan los sectores del elipsoide en un plano. La figura 9.2 ilustra a grandes rasgos la forma en que se realiza la proyección en el plano para cada tipo.

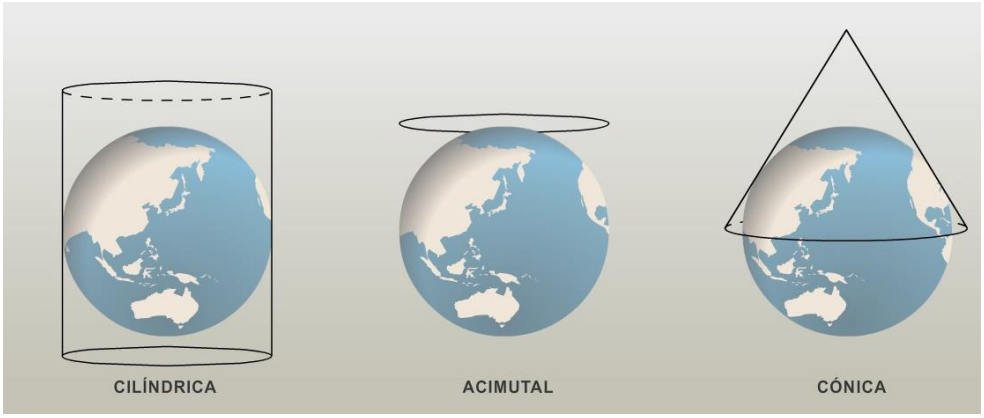

Figura 9.2: Esquema de proyecciones de la tierra.

Hay que considerar que una proyección será mejor que otra, del mismo modo que los elipsoides, sólo a nivel regional. Por ejemplo, si consideramos una proyección cónica tomada desde un polo, y la comparamos con una cilíndrica que haga tangencia en la Línea del Ecuador, podemos decir que:

1. La proyección cónica sólo tiene sentido en un hemisferio del elipsoide, mientras que la cilíndrica sirve para mapear el elipsoide completo.
2. La proyección cónica es más precisa que la cilíndrica en el polo.
3. La proyección cilíndrica es más precisa en el Ecuador, y va perdiendo precisión a medida que se aleja hacia los polos.

La más utilizada dentro de las proyecciones globales (aquellas que mapean el elipsoide completo) es la Proyección de *Mercator*. Esta es una proyección cilíndrica y conforme (esto es, que conserva los ángulos) formalizada por Gerardus Mercator en el siglo XVI, y base para el sistema de coordenadas UTM (Universal Tranverse Mercator).

9.2 Gramática BNF que define Well-Known Text

```
<well-known text representation> ::=
    <point text representation> |
    <curve text representation> |
    <surface text representation> |
    <collection text representation>

<point text representation> ::= POINT [ <z m> ] <point text>

<curve text representation> ::=
    <linestring text representation> |
    <circularstring text representation> |
    <compoundcurve text representation>
```

```
<linestring text representation> ::=
    LINESTRING [ <z m> ] <linestring text body>

<circularstring text representation> ::=
    CIRCULARSTRING [ <z m> ] <circularstring text>

<compoundcurve text representation> ::=
    COMPOUNDCURVE [ <z m> ] <compoundcurve text>

<surface text representation> ::=
    <curvepolygon text representation>

<curvepolygon text representation> ::=
    CURVEPOLYGON [ <z m> ] <curvepolygon text body> |
    <polygon text representation> |
    <triangle text representation>

<polygon text representation> ::=
    POLYGON [ <z m> ] <polygon text body>

<triangle text representation> ::=
    TRIANGLE [ <z m> ] <triangle text body>

<collection text representation> ::=
    <multipoint text representation> |
    <multicurve text representation> |
    <multisurface text representation> |
    <geometrycollection text representation>

<multipoint text representation> ::=
    MULTIPOINT [ <z m> ] <multipoint text>

<multicurve text representation> ::=
    MULTICURVE [ <z m> ] <multicurve text> |
    <multilinestring text representation>

<multilinestring text representation> ::=
    MULTILINESTRING [ <z m> ] <multilinestring text>

<multisurface text representation> ::=
    MULTISURFACE [ <z m> ] <multisurface text> |
    <multipolygon text representation> |
    <polyhedralsurface text representation> |
    <tin text representation>

<multipolygon text representation> ::=
    MULTIPOLYGON [ <z m> ] <multipolygon text>

<polyhedralsurface text representation> ::=
    POLYHEDRALSURFACE [ <z m> ] <polyhedralsurface text>

<tin text representation> ::=
    TIN [ <z m> ] <tin text>

<geometrycollection text representation> ::=
    GEOMETRYCOLLECTION [ <z m> ] <geometrycollection text>
```

```
<linestring text body> ::=
    <linestring text>

<curvepolygon text body> ::=
    <curvepolygon text>

<polygon text body> ::=
    <polygon text>

<triangle text body> ::=
    <triangle text>

<point text> ::=
    <empty set> |
    <left paren> <point> <right paren>

<point> ::= <x> <y> [ <z> ] [ <m> ]

<x> ::= <number>
<y> ::= <number>
<z> ::= <number>
<m> ::= <number>

<linestring text> ::=
    <empty set> |
    <left paren> <point> { <comma> <point> }... <right paren>

<circularstring text> ::=
    <empty set> |
    <left paren> <point> { <comma> <point> }... <right paren>

<compoundcurve text> ::=
    <empty set> |
    <left paren> <single curve text> { <comma> <single curve text> }... <right
paren>

<single curve text> ::=
    <linestring text body> |
    <circularstring text representation>

<curve text> ::=
    <linestring text body> |
    <circularstring text representation> |
    <compoundcurve text representation>

<ring text> ::=
    <linestring text body> |
    <circularstring text representation> |
    <compoundcurve text representation>

<surface text> ::=
    CURVEPOLYGON <curvepolygon text body> |
    <polygon text body>

<curvepolygon text> ::=
    <empty set> |
    <left paren> <ring text> { <comma> <ring text> }... <right paren>
```

```
<polygon text> ::=
    <empty set> |
    <left paren> <linestring text> { <comma> <linestring text> }... <right
paren>

<triangle text> ::=
    <empty set> |
    <left paren> <linestring text> <right paren>

<multipoint text> ::=
    <empty set> |
    <left paren> <point text> { <comma> <point text > }... <right paren>

<multicurve text> ::=
    <empty set> |
    <left paren> <curve text> { <comma> <curve text> }... <right paren>

<multilinestring text> ::=
    <empty set> |
    <left paren> <linestring text body> { <comma> <linestring text body> }...
<right paren>

<multisurface text> ::=
    <empty set> |
    <left paren> <surface text> { <comma> <surface text> }... <right paren>

<multipolygon text> ::=
    <empty set> |
    <left paren> <polygon text body> { <comma> <polygon text body> }... <right
paren>

<polyhedralsurface text> ::=
    <empty set> |
    <left paren> <polygon text body> { <comma> <polygon text body> }... <right
paren>

<tin text> ::=
    <empty set> |
    <left paren> <triangle text body> { <comma> <triangle text body> }...
<right paren>

<geometrycollection text> ::=
    <empty set> |
    <left paren> <well-known text representation> { <comma> <well-known text
representation> }... <right paren>

<empty set> ::= EMPTY

<z m> ::= ZM | Z | M
<left paren> ::= (
<right paren> ::= )
```