Líder en Servicios y Soluciones informáticas globales

Universitat Autònoma de Barcelona
Departament de Geografia

DISEÑO, DESARROLLO E IMPLEMENTACIÓN DE UNA APLICACIÓN WEB DE INVENTARIO PARA LA GESTIÓN DE DIFERENTES ÁMBITOS DE ESTUDIO (PARQUES, JARDINES, PLAYAS Y MOBILIARIO URBANO).

Proyecto final de Máster en Tecnologías de las Información Geográfica, 12ª edición.

Organizador: Departamento de Geografía (LIGIT)

Empresa colaboradora: SEYS Semiconductores y Sistemas, S.A.

Autor: Lara de Gregorio

Tutor LIGIT: Ignacio Ferrero

Tutor SEYS: Andrés Torregrosa Mejías

12mtig 2010
Professionals per a la Societat de la Informació

RESUMEN

La aplicación se ha desarrollado durante un periodo di practica trimestral en la empresa Seys Semiconductores y Sistemas S.A. ubicada en Barcelona en colaboración con la Universidad Autónoma de Barcelona. Este convenio ha permitido la realización del proyecto final del Máster en Tecnologías de la Información Geográfica, 12ª edición, organizado por el Departamento de Geografía y llevado a cabo por el Laboratorio de Información Geográfica y Teledetección (LIGIT).

El objetivo del proyecto es desarrollar una herramienta de gestión web de inventario de fácil uso adaptable a más de un ámbito de estudio (Ej. parques, jardines, playas, mobiliario urbano) que permita la identificación de los elementos puntuales, la consulta e actualización periódica de los datos. El proyecto no tiene un cliente real se propone crear una aplicación web a nivel de investigación interna a la empresa Seys que pueda ser utilizada para los futuros pedidos de los clientes de la empresa.

La primera fase ha sido un análisis de requerimientos hecha a través de una reunión con la empresa que ha permitido definir los objetivos específicos que se pretendían conseguir con la aplicación.

El desarrollo del proyecto incluyo dos fases, la primera de creación del sistema informativo y la segunda de desarrollo de la aplicación web.

El diseño lógico de la base de datos se ha realizado con el software Power Design® 12.5 y se ha implementado su plataforma Oracle® Database 11g. Una vez completada la base de datos la fase siguiente ha sido el diseño de la aplicación definiendo todas las funcionalidades de cada una de las herramientas da implementar y también el aspecto visual que se quería dar a la aplicación. La tecnología SIG en uso en la empresa Seys es Autodesk® MapGuide en este proyecto se ha utilizado la última versión MapGuide Enterprise 11 y la programación de la aplicación ha sido en ASP.NET su servidor web Microsoft® Windows® Server 2003. Como entorno de desarrollo se ha usado Microsoft® Visual Studio® 2008.

El resultado ha sido una aplicación web intuitiva, eficiente y atractiva. La funcionalidad ha sido probada sobre un conjunto de datos experimentales relativos a un parque de prueba, cada elemento puntual del parque puede ser identificado en un formulario con sus atributos y los atributos pueden ser modificados y también añadidos. Hay un segundo formulario activable desde la barra de herramienta que permite hacer consultas a la base de datos y las consultas devuelven una tabla con todos los atributos de cada elemento resultado de la consulta. Haciendo click en la tabla resultado el programa permitirá también visualizar en el mapa la posición del elemento descrito.

Las dificultades encontradas en la estructura interna del software MapGuide han sido superadas y se pueden afirmar que el proyecto ha tenido un éxito positivo, dejando satisfechas las partes interesadas.

ÍNDICE

1.INTRODUCCIÓN

1.1. ¿Qué es una base de datos?

Todas las empresas requieren almacenar información. Desde siempre lo han hecho. La información puede ser de todo tipo. Cada elemento informativo (nombre, dirección, sueldo, etc.) es lo que se conoce como dato.

Las soluciones utilizadas por las empresas para almacenar los datos son diversas. Antes de la aparición de la informática se almacenaban en ficheros con cajones, carpetas y fichas. Tras la aparición de la informática estos datos se almacenan en archivos digitales dentro de las unidades de almacenamiento del ordenador (a veces en archivos binarios, o en hojas de cálculo).

Además las empresas requieren utilizar aplicaciones informáticas para realizar tareas propias de la empresa a fin de mecanizar a las mismas. Estas aplicaciones requieren manejar los datos de la empresa. En los inicios de la era informática, cada programa almacenaba y utilizaba sus propios datos de forma caótica. La ventaja de este sistema (la única ventaja), es que los procesos eran independientes por lo que la modificación de uno no afectaba al resto. Pero tiene grandes inconvenientes:

- € coste de almacenamiento elevado;
- € datos redundantes;
- € probabilidad alta de inconsistencia en los datos;
- € difícil modificación en los datos y problemas de inconsistencia al realizar esas modificaciones.

Lógicamente la solución a este problema es hacer que todas las aplicaciones utilicen los mismos datos. Esto provoca que los datos deban estar mucho más protegidos y controlados. Además los datos forman una estructura física y funcional que es lo que se conoce como *base de datos*.

De esta forma una *base de datos* es una serie de datos relacionados que forman una estructura lógica, es decir una estructura reconocible desde un programa informático.

Esa estructura no sólo contiene los datos en sí, sino la forma en la que se relacionan. Las bases de datos empiezan a aparecer en los años 60 y triunfan en los años setenta y Ochenta [1].

1.2. Servidores de base de datos y servidores web

Los servidores de bases de datos surgen con motivo de la necesidad de las empresas de manejar grandes y complejos volúmenes de datos, al tiempo que requieren compartir la

información con un conjunto de clientes (que pueden ser tanto aplicaciones como usuarios) de una manera segura. Avanzando un grado más en las capas de servicios que debe proporcionar un servidor de datos, nos encontramos con las herramientas que proporciona tanto al usuario administrador como al cliente consumidor de los datos: los sistemas de gestión de bases de datos (SGBD). Estos son software muy específicos desarrollados principalmente para servir de interfaz entre las bases de datos y los usuarios y las aplicaciones. Estas herramientas deben proporcionarle un entorno amigable y sencillo de manejar, que le permita orientarse a su trabajo y no preocuparse con detalles de más bajo nivel, al tiempo que le permite realizar sus tareas de la manera más rápida y simplificada posible. Un SGBD debe proporcionar también servicios de forma global y en la medida de lo posible, independientemente de la plataforma. Internet se ha convertido en la mayor plataforma de comunicaciones jamás vista. Esto hace que las empresas tiendan a presentar su información a través de la Web en forma de contenidos, que después los clientes consultarán para establecer relaciones con dichas empresas[2].

Puesto que todo tiende a unificarse con Internet, los servidores de datos también deben proporcionar servicios de datos a la red. Los servicios disponibles incorporan generación y alimentación de páginas Web a partir de consultas prediseñadas en la base de datos por esto que se necesita un Servidor web.

El Servidor web se ejecuta en un ordenador manteniéndose a la espera de peticiones por parte de un cliente (un navegador web) y que responde a estas peticiones adecuadamente, mediante una página web que se exhibirá en el navegador o mostrando el respectivo mensaje si se detectó algún error. Instalar un servidor web en nuestro PC nos permitirá, entre otras cosas, poder montar nuestra propia página web sin necesidad de contratar hosting, probar nuestros desarrollos vía local, acceder a los archivos de nuestro equipo desde un PC remoto[3].

1.3. IDE (entorno de desarrollo informático)

Un entorno de desarrollo informático (en inglés *Integrated Development Environment* - IDE) es un programa informático compuesto por un conjunto de herramientas de programación, puede dedicarse en exclusiva a un solo lenguaje de programación o bien, poder utilizarse para varios. Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Los IDEs pueden ser aplicaciones por sí solos o pueden ser parte de aplicaciones existentes[3].

1.4. Autodesk MapGuide

1.4.1. Características básicas y requisitos de instalación

MapGuide es una aplicación sistemas de información geográfica (GIS) centrada en red. El GIS es construido desde un principio para la arquitectura de Internet y de intranets corporativas, es el avance tecnológico más importante dentro del sector. Los usuarios de GIS tienen una elevada facilidad para acceder a la información dado que los sistemas utilizan este entorno de red distribuido. Con estos sistemas, cada persona en una organización tendrá total capacidad para visionar, realizar "*queries*" y obtener informes en un contexto geográfico espacial. Hasta el 1998 los usuarios debían de hacerlo con programas GIS que permitían un acceso a la red. La distribución de datos con estos sistemas es lenta y comporta altos costes de desarrollo e implementación. Un nuevo concepto de GIS centrado en la red (*network centric*) permitió a las organizaciones compartir esta información con un coste muy reducido y con eficacia. Autodesk MapGuide ha sido el líder en innovación en este nuevo mercado de rápido crecimiento. Es un programa diseñado desde el primer momento para la arquitectura de la Web y como resultado ofrece unas mejores prestaciones, facilidad de uso y un coste reducido (Joe Astroth, 1998)[4].

MapGuide está formado por cuatro componentes independientes que deben instalarse en el siguiente orden:

- MapGuide Server
- MapGuide Web Extensions (para el desarrollo de aplicaciones)
- Autodesk MapGuide Studio (para la creación de mapas)
- MapGuide Viewer

Requisitos de instalación:

- Tanto MapGuide Server como MapGuide Web Extensions se ejecutan tanto en Linux como en Windows.
- MapGuide Web Extensions se ejecuta en IIS y Apache (Windows) y Apache (Windows y Linux).
- Autodesk MapGuide Studio se ejecuta en Windows.
- Los visores de MapGuide .Hay dos tipologías del Visor de MapGuide: el visor AJAX (o "cliente espectador") no necesita una descarga. Si se ejecuta en Windows trabaja con Internet Explorer, Safari o Firefox y el visor descargable (Autodesk DWF espectador) se basa en un Microsoft ActiveX Control y tiene soporte completo para el formato DWF. Funciona con el navegador Microsoft Internet Explorer solamente.
- MapGuide Web Extensions admite el desarrollo de aplicaciones en PHP, ASP y JSP.

1.4.2. Presentación general de los componentes

Autodesk MapGuide Studio y MapGuide Viewer son aplicaciones cliente de MapGuide Server. Las solicitudes procedentes de estos clientes, así como de cualquier otra aplicación cliente desarrollada utilizando la API, son dirigidas a Web Extensions a través del protocolo HTTP. El componente WebAgent de Web Extensions procesa las solicitudes y las envía a MapGuide Server. Al iniciar Autodesk MapGuide Studio, debe introducir la dirección IP del servidor Web que aloja Web Extensions. Por ejemplo, http://localhost/mapguide2011. Al recibir una solicitud, MapGuide Server accede a los recursos almacenados en el repositorio de recursos, crea el mapa y lo modeliza como una imagen estática para el visor AJAX, y lo devuelve a Web Extensions, que a su vez lo devuelve al cliente. Autodesk MapGuide Studio tiene integrado MapGuide Viewer. El visor muestra los datos modelizados en la pantalla y administra las capas y otras propiedades de los datos que devuelve el servidor[6].

1.5. Tecnología ASP.NET

El desarrollo de aplicaciones para el entorno Internet ha cambiado mucho desde su aparición hace algunos años. El mundo de la informática avanza rápidamente, y las necesidades básicas de un sitio Web han cambiado. Al principio, el desarrollo de páginas Web se limitaba prácticamente al uso del lenguaje de marcado HTML (lenguaje de marcado de hipertexto), un HTML sencillo basado en la especificación 3.2, sin capas ni estilos, con lo que se obtenían páginas estáticas cuya labor únicamente era suministrar información personal del autor o de la empresa (de forma textual o a través de imágenes), pero, sin proporcionar ninguna interacción con el usuario. Actualmente, un sitio Web suele soportar la recogida de datos a través de formularios, inclusiones de archivos (subir ficheros al servidor, como imágenes o documentos), envío de correos electrónicos, acceso a bases de datos, re direccionamiento de acuerdo al perfil del usuario, etc. En definitiva, hoy día, las aplicaciones Web se diseñan con el propósito de suministrar al usuario una serie de servicios más complejos.

ASP3 es la versión previa a la tecnología ASP.NET, y por tanto, también la solución inicial de Microsoft a la programación Web. Sus siglas se corresponden con *Active Server Pages* (Páginas Activas de Servidor) en su versión 3, y funcionaban bajo las versiones del servidor Web de Microsoft, IIS (*Internet Information Server*, Servidor de Información de Internet) en sus versiones 3, 4 y 5. Este sistema se basaba en la ejecución de una serie de lenguajes de script (principalmente, VBScript y Jscript) embebidos en páginas HTML. La extensión de estas páginas es .asp (son las comúnmente llamadas páginas ASP, y que aún hoy siguen utilizándose en la programación de multitud de sitios Web).

Su proceso de des rrollo fue relativamente rápido, hoy está integrado con el modelo COM (Modelo de Objetos Componentes) de Windows y se encuentra muy extendido (ha sido uno de los más utilizados en los últimos tiempos).

A partir de la versión 5 de su servidor Web, IIS, Microsoft da soporte a su nueva solución de programación Web, ASP.NET, (aunque existe compatibilidad con el código ASP anterior) ASP.NET mejora en los lenguajes y herramientas de desarrollo, asume el modelo de programación orientada a objetos, mejora el rendimiento general del sistema. A diferencia de las páginas ASP tradicionales, las páginas ASP.NET se compilan antes de ejecutarse. La primera vez que alguien accede a una página ASP.NET, esta se compila y se crea un fichero ejecutable que se almacena en una caché del servidor web, un *assembly* si utilizamos la terminología propia de la plataforma .NET. De esta forma, las siguientes ocasiones en las que se solicite la página, se podrá usar directamente el ejecutable. Al no tener que volver a compilar la página ASP.NET, la ejecución de esta será más eficiente que la de una página ASP convencional[7].

2. MARCO DEL TRABAJO

2.1. Presentación

Este trabajo ha sido realizado en un periodo trimestral (setiembre - diciembre) en el departamento de desarrollo de la empresa Seys Semiconductores y Sistemas S.A. ubicada en Barcelona en colaboración con la Universidad Autónoma de Barcelona. Este convenio ha permitido la realización del proyecto final del Máster en Tecnologías de la Información Geográfica, 12ª edición, organizado por el Departamento de Geografía, y llevado a cabo por el Laboratorio de Información Geográfica y Teledetección (LIGIT).

La empresa Seys Semiconductores y Sistemas S.A. ha evolucionado en paralelo con las mejores compañías del sector de las Tecnologías de la Información y hoy se ocupa de desarrollar aplicaciones GIS para distintos tipo de actividades, desde personalización de software estándar y medianas soluciones a medida hasta complejos proyectos integrales bajo definición de necesidades de sus clientes (universidades, ayuntamientos, empresas privadas etc.).

2.2. Fases del proyecto

La ejecución de un proyecto de programación web implica la definición de diferentes tareas tanto de carácter organizativo, así como de diseño y desarrollo. Las fases del proyecto se pueden resumir en Figura 2.1.

La fase de definición es la fase que vamos a profundizar en este capítulo es una fase que permite definir los requisitos que la aplicación debería tener una vez completada y los objetivos específicos que se quieren alcanzar (las funcionalidades de la aplicación). A de más se definen las tecnologías que se quiere utilizar y se procede a la instalación de un entorno de trabajo. La análisis de requerimientos se hace a través de una reunión con el cliente, en este caso no hay un cliente real porqué es una aplicación a nivel de investigación pero podemos considerar como tale la empresa Seys que nos ha propuesto el trabajo. La reunión hubo lugar con ambos los tutores. La propuesta para la definición de la plataforma tecnológica ha sido prevalentemente por parte de la empresa que necesitaba una aplicación compatible con la tecnología en uso en la misma para permitir el utilizo futuro del proyecto por parte de su equipo de desarrollo.

La fase de diseño y de implementación serán explicadas con mayor detalle en el Capítulo 3 dedicado a la metodología agrupándolas en dos apartados uno de sistemas en el que se hablará del diseño y de la implementación de la base de datos y uno dedicado al diseño y a la implementación de la aplicación web.

Figura 2.1 Fases del proyecto

Los resultados se mostrarán en el Capitulo 4 mediante capturas de pantalla del producto final y se analizará toda la operatividad de la aplicación entrando detalladamente en la fase de análisis de resultados incluyendo el control de errores.

En este proyecto no tendremos la última fase que es la instalación porque como ya dicho al principio no tenemos un cliente real pero es importante recordar que en esta fase el desarrollador debería instalar el programa en los ordenadores del cliente y siempre es una fase crucial porque la mayoría de las veces se producen errores y incidencias debidas al diferente entorno informático por esto que consideramos de fundamental importancia la primera fase que nos permite elegir la mejor tecnología requerida por un determinado trabajo.

2.3. Plan de trabajo

Para dar al lector una idea de cuánto tiempo se ha dedicado a las diferentes fases del proyecto en este párrafo se explicará brevemente como ha sido divido el periodo de práctica en estrecha relación con el plan de trabajo. Cada periodo ha sido definido para poder seguir el trabajo y tener siempre bajo control el desarrollo en las distintas partes del proyecto. El esquema planeado se muestra en Figura 2.2.

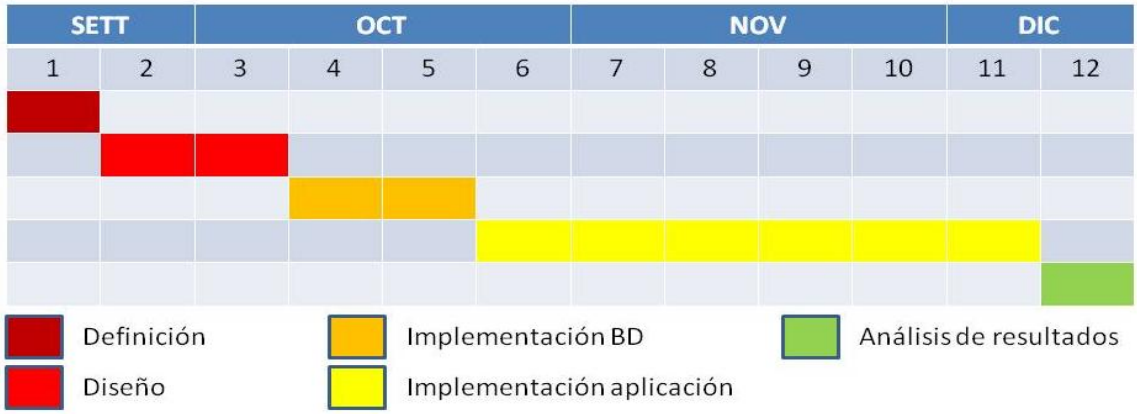


Figura 2.2 Plan de trabajo

2.4. Objetivos

El objetivo fundamental que es desarrollar una herramienta de gestión web de inventario de fácil uso adaptable a más de un ámbito de estudio (Ej. parques, jardines, playas, mobiliario urbano) que permita la identificación de los elementos puntuales, la consulta e actualización periódica de los datos.

Basado en este objetivo fundamental, la aplicación estará dotada de las siguientes funcionalidades:

1) Reunir en una base de datos toda la información alfanumérica del ámbito de estudio a partir de un diseño lógico más genérico posible así que pueda ser adaptable a los requerimientos de diferentes clientes.
2) Fácil uso que permita actualización y gestión de incidencia por parte de personal no calificado.
3) Identificación de los atributos de los elementos puntuales del ámbito de estudio por medio de un formulario dinámico activable haciendo click directamente en el mapa (solo los elementos puntuales serán seleccionables).
4) Actualización de la base de datos a partir de la modifica de las informaciones por parte del usuario en la interfaz web.
5) Posibilidad de añadir atributos a un elemento directamente en el formulario web.
6) Consulta de la base de datos desde un prepuesto formulario activable haciendo click en un botón añadido a la barra de herramientas del visor web.
7) Visualización en una tabla de los resultados de la consulta.
8) Posicionamiento en el mapa de los elementos resultados de la consulta haciendo click en la tabla resultado del formulario de consulta (el elemento resultará evidenciado con un cuadrado de diferente color).

2.5. Casos de Uso

Los casos de usos son una sucesión de transacciones en un sistema cuyo objetivo es proporcionar un servicio a un usuario del sistema[8]. En otras palabras, un caso de uso es una secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema[3]. Los diagramas de casos de uso sirven para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas. Una relación es una conexión entre los elementos del modelo.

Los diagramas de casos de uso se utilizan para ilustrar los requerimientos del sistema y esquematizan las relaciones existentes entre actores y las respuestas del sistema frente a diferentes estímulos que se generan en su entorno. En todo diagrama de casos de uso aparecen actores, entendiendo como tales las entidades externas (personas, dispositivos, subsistemas, etc.) que interactúan con el sistema e interpretan un rol determinado[8]. Los casos de uso permiten, por lo tanto, ahorrar tiempo y recursos, pues solventan a priori problemas que en el producto final no serían admisibles.

En el caso que nos compete en el proyecto, existen tres actores: el usuario final, que se encarga de manejar el producto, el servidor de BBDD, que proporciona parte de los resultados y el servidor web que almacena informaciones en forma de páginas web y a través del protocolo HTTP las entrega a peticiones de los clientes (en un navegador web) en formato HTML.

Se presentan en este párrafo los casos de uso para las principales funcionalidades que se planteó conseguir con este proyecto.

Caso de uso: Identificación en el mapa de elementos puntuales

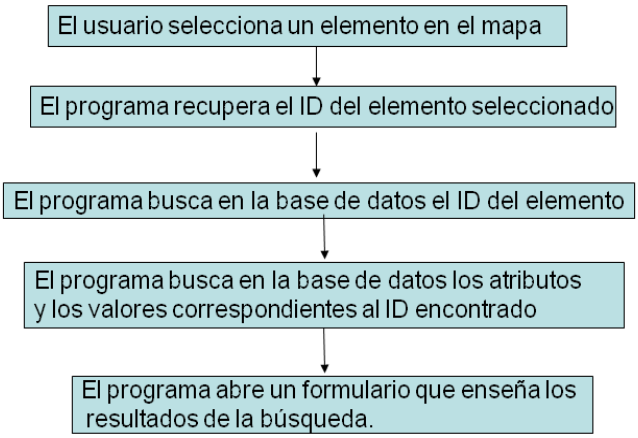
Sumario:
El usuario desea visualizar un formulario con los atributos de cada elemento que selecciona en el plano.
Se buscará el ID del elemento en la base de datos y luego sus atributos y los valores.
El caso finaliza cuando será posible visualizar correctamente el formulario del elemento.

Descripción:
Un usuario con acceso al programa selecciona un elemento en el plan.
El programa buscará el ID del elemento, sus atributos y los valores correspondientes en la base de datos, solo se mostrará si el elemento seleccionado es un punto (en el mapa será posible seccionar solo puntos).
El programa devolverá una pantalla nueva donde se observaran para cada elemento seleccionado por el usuario los atributos con sus valores.

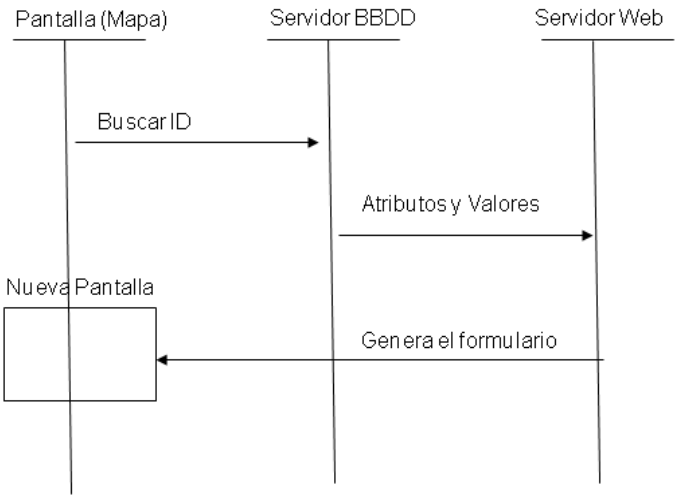
Objetivo:
Abrir una nueva pantalla donde se visualiza el formulario de identificación de un elemento seleccionado.

Actores:
Cualquier persona autorizada con acceso al programa, servidor de BBDD, servidor web.

Diagrama:



Esquema:



Caso de uso: Modifica valores en el formulario de identificación

Sumario:
El usuario desea modificar uno o más valores el formulario de identificación.
El usuario modificará el formulario y una vez seleccionado el botón Guardar las informaciones vendrán guardadas en la base de datos. El caso finaliza cuando la aplicación guarda los nuevos valores en la base de datos.

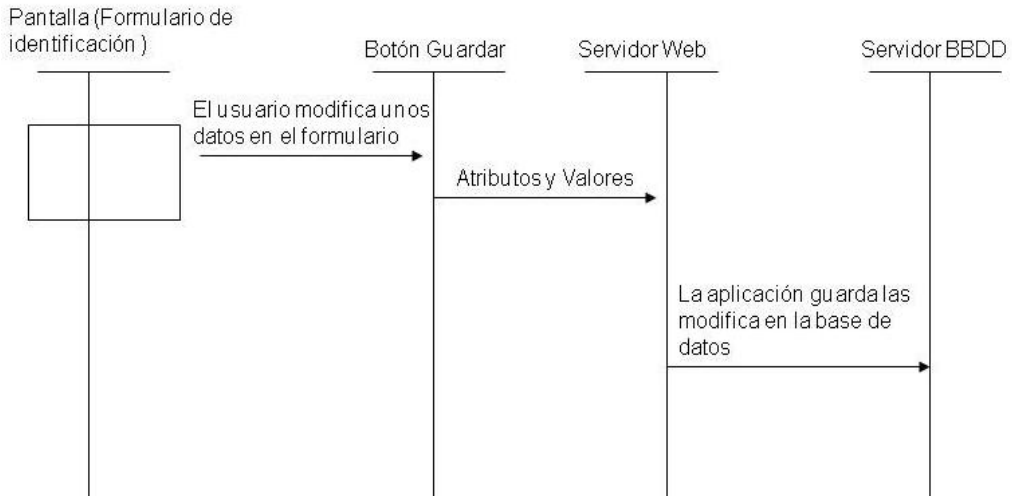
Descripción:
Un usuario con acceso al programa selecciona un elemento en el plan y visualiza su formulario de identificación.
El usuario modificará algunos datos en el formulario y seleccionará el botón Guardar.
La aplicación recuperará los datos modificados y los guardará en la base datos.

Objetivo:
Guardar en la base datos las modifica hecha desde la interfaz web.

Actores:
Cualquier persona autorizada con acceso al programa, servidor de BBDD, servidor web.

Diagrama:

Esquema:



Caso de uso: Visualización en el mapa de los elementos resultado de una consulta

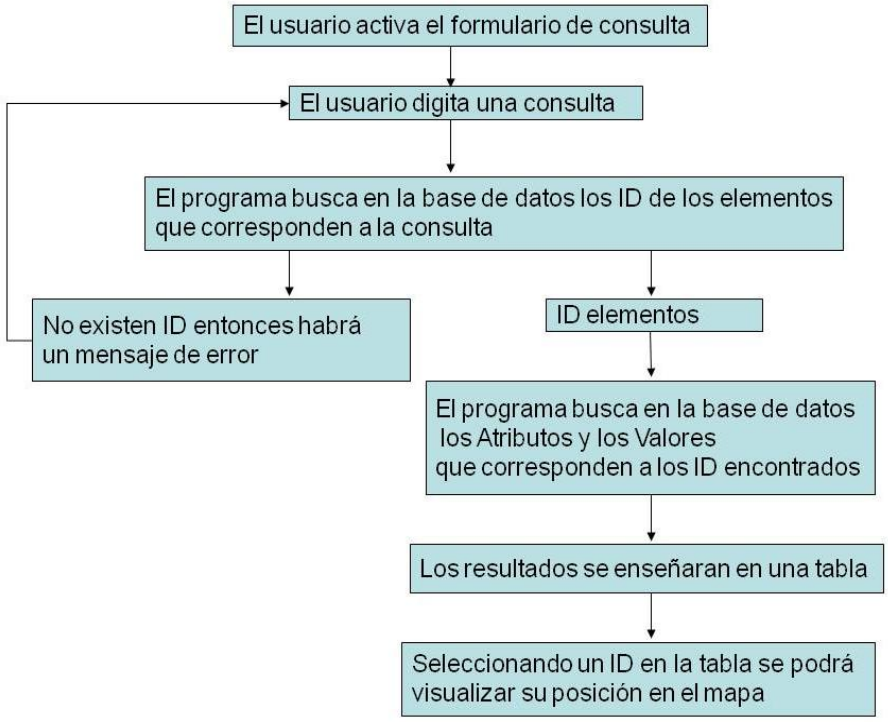
Sumario:
El usuario desea hacer una búsqueda y visualizar en el mapa el resultado.
Se activará un formulario de consulta que devolverá uno o más ID del elemento. El resultado se visualizará en una tabla.
El caso finaliza cuando el usuario haciendo click en la tabla podrá visualizar la posición del elemento en el mapa.

Descripción:
Un usuario con acceso al programa selecciona en la barra de herramienta del visor el botón que abre el formulario de consulta.
Se muestra una pantalla nueva que contiene el formulario de consulta donde el usuario podrá seleccionar las características de los elementos de interés.
El programa buscará en la base de datos y devolverá el ID del elemento, sus atributos y los valores correspondientes, los resultados se muestrearan en una tabla en la misma pantalla.
Si la búsqueda no devuelve algún valor habrá un mensaje de error.
El usuario seleccionando los elementos en la tabla resultado visualizará los elementos gráficos evidenciados con un color diferente.

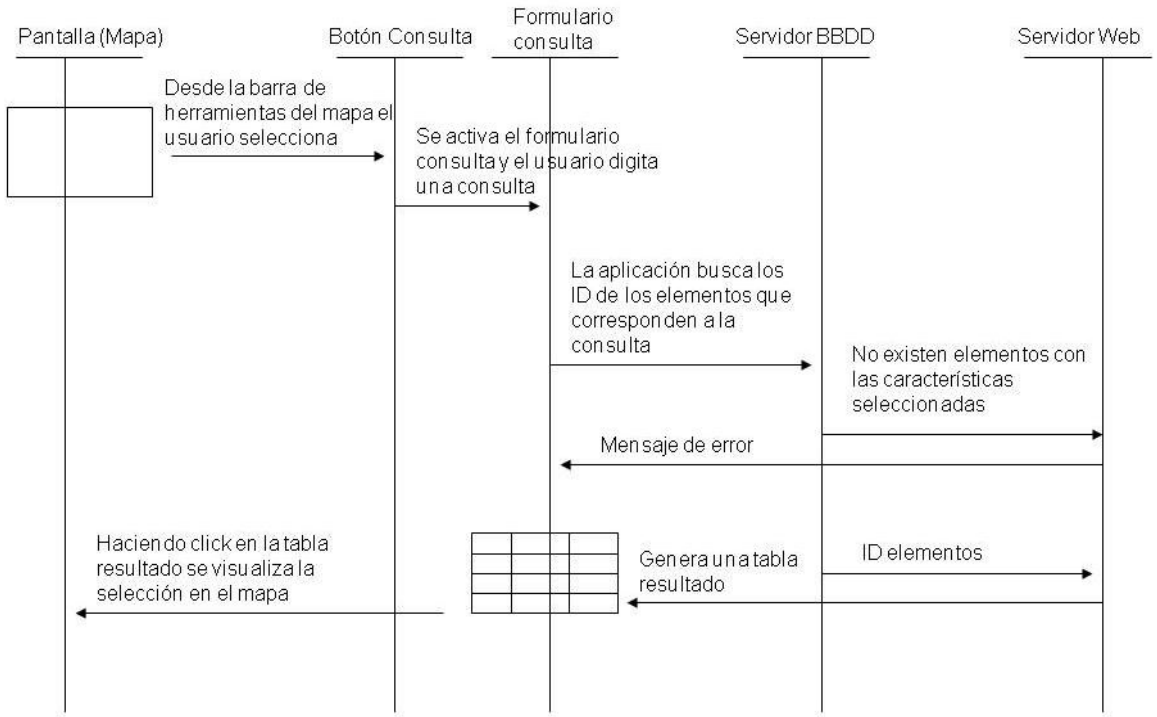
Objetivo:
Averiguar que la búsqueda devuelva resultado, representar el mismo en una tabla y visualizar los elementos en el mapa.

Actores:
Cualquier persona autorizada con acceso al programa, servidor de BBDD, servidor web.

Diagrama:



Esquema:



Caso de uso: Añadir atributos a un elemento en el formulario de identificación

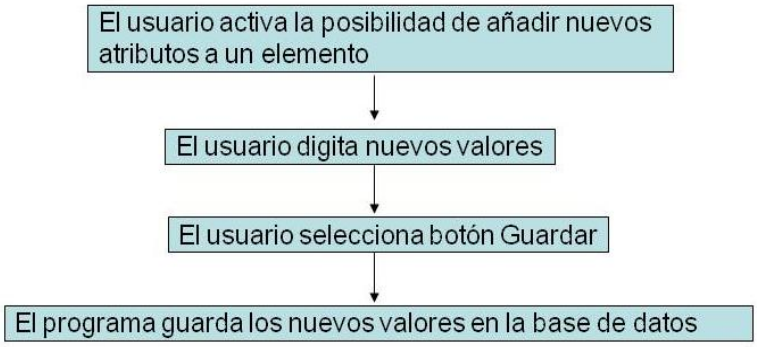
Sumario:
El usuario desea añadir un atributo a un elemento en el formulario de identificación.
El usuario activará en el mismo formulario un panel invisible seleccionando el botón Añadir Atributo.
Una vez seleccionado el botón Guardar las informaciones vendrán guardadas en la base de datos.
El caso finaliza cuando la aplicación guarda los nuevos valores en la base de datos.

Descripción:
Un usuario con acceso al programa selecciona un botón que activará un panel que permite añadir atributos a un elemento.
El usuario insertará nuevos datos en el formulario y seleccionará el botón Guardar.
La aplicación guardará los nuevos datos en la base de datos.

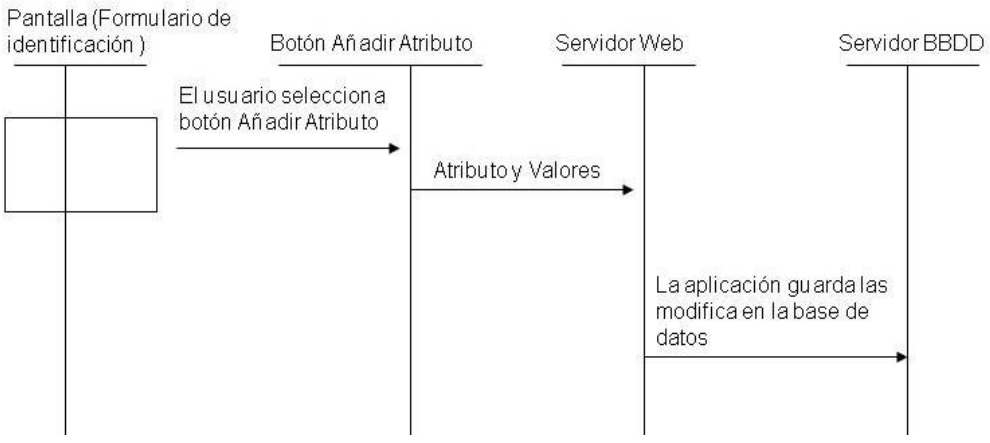
Objetivo:
Guardar en la base datos nuevos valores insertados desde la interfaz web.

Actores:
Cualquier persona autorizada con acceso al programa, servidor de BBDD, servidor web.

Diagrama:



Esquema:



2.6. Plataforma tecnológica

2.6.1. Requerimientos tecnológicos

Los requerimientos tecnológicos para el desarrollo de esta aplicación se pueden resumir en los siguientes:

- un servidor de bases de datos, así como una herramienta para su gestión (SGBD);
- un servidor Web;
- un entorno de desarrollo de la aplicación - IDE (*Integrated Development Environment*).

Ver Capitulo 1 para una detallada explicación de cada componente.

2.6.2. Tecnologías utilizadas en el proyecto

El diseño lógico de la base de datos se ha realizado con el software Power Design® 12.5 y se ha implementado su plataforma Oracle® Database 11g. Se ha elegido Oracle 11g porque es una plataforma integral de base de datos (SGBD) que permite análisis bien integradas, calidad de datos, garantiza integridad de la información y se ejecuta en una infraestructura *grid* de bajo costo y confiable. Como interfaz entre las bases de datos y el usuarios se utilizó Toad® for Oracle, un software que permite manejar de manera clara, sencilla y ordenada un conjunto de datos.

La tecnología SIG en uso en la empresa Seys es Autodesk® MapGuide, una plataforma de software para distribuir datos espaciales en Internet o en una intranet. Existen dos versiones de MapGuide: MapGuide Open Source y MapGuide Enterprise. En este proyecto se ha utilizado la última versión MapGuide Enterprise 11. El Visor de MapGuide proporciona un medio para ver mapas en un navegador web. Después haber cargado los datos espaciales en el internet o en la intranet de una empresa, los usuarios pueden utilizar el visor para mostrar los datos e interactuar con ellos. En el proyecto se ha utilizado el visor AJAX (ver Capitulo 1.4) el cual permite también personalizar la apariencia del Visor de MapGuide y seleccionar los comandos de barra de herramientas que desea utilizar con una plantilla llamada de “diseño web”. Por supuesto, también permite agregar características y funciones programadas a través de un código proprio del administrador.

La programación de la aplicación ha sido en ASP.NET (ver Capitulo 1.5) su servidor web Microsoft® Windows® Server 2003 IIS6 64 Bits.

Como entorno de desarrollo (IDE) se ha usado Microsoft® Visual Studio 2008 que permite manejar varios lenguajes de programación de manera unificada y simplificada. Los lenguajes de programación que se han utilizado para la programar la aplicación son Visual Basic y Java Script y para el dibujo de los formularios HTML. En Figura 2.3 están resumidas las tecnologías utilizadas en el proyecto.

Tecnología	Software
Servidor Web	Windows Server 2003
Sistema de Gestión de Base de Datos	Oracle 11g
Administrador de Base de Datos	Toad for Oracle
Diseño de la Base de Datos	PowerDesigner 12.5
Desarrollo cliente web	Visual Studio 2008
GIS	MapGuide Enterprise 2011

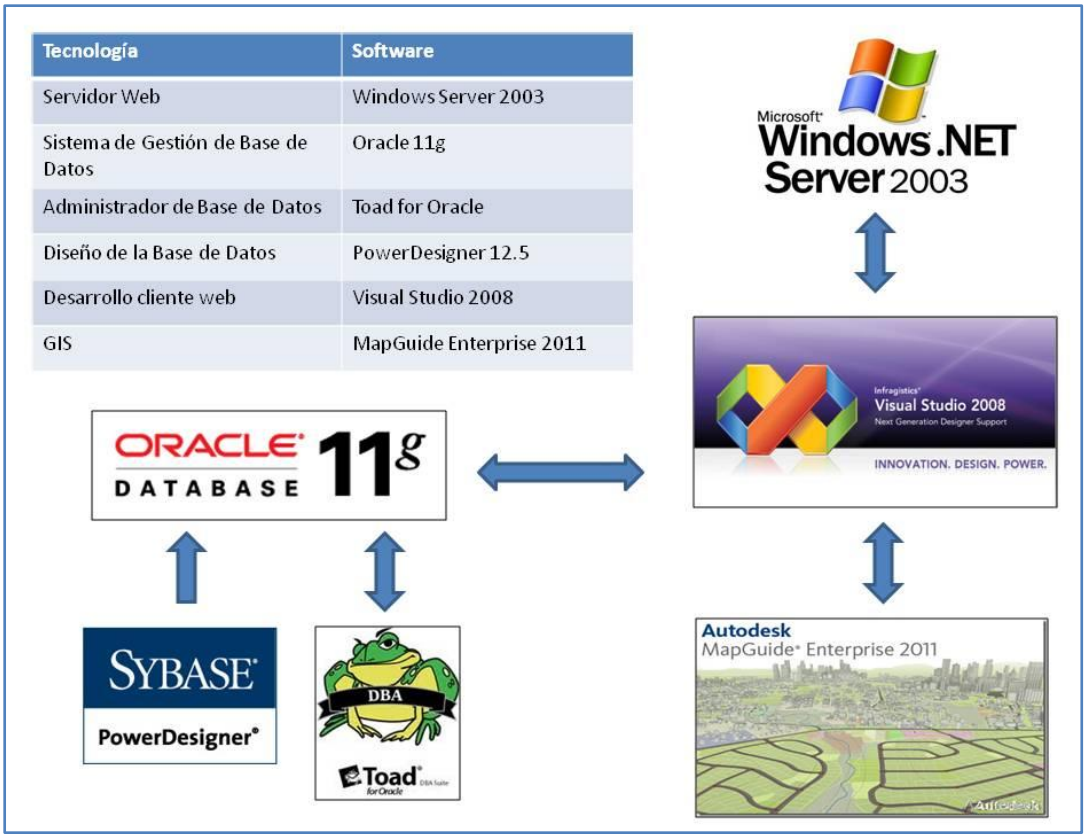


Figura 2.3 Tecnologías utilizadas en el proyecto.

3. METODOLOGÍA

3.1. Presentación

El proyecto asignado es integral sin ningún antecedente concreto entonces antes de empezar el desarrollo de la aplicación a través de la programación se ha creado la base de datos por lo tanto este capítulo dedicado a la metodología está dividido en dos apartados uno de sistemas en el que se hablará del diseño y de la implementación de la base de datos y uno dedicado al diseño y a la implementación de la aplicación web.

3.2. Creación de la base de datos

3.2.1. Diseño de la base de datos, diagrama conceptual y lógico

El diseño conceptual es un esquema de la base de datos (BD) que permite definir los objetos reales en términos de entidad con sus propios atributos y definir las relaciones cuantitativas que hay entre las entidades (ej. 0-n, n-n, 1-1).

El diseño lógico es un esquema de la estructura formal de la BD. Se definen para cada entidad las reglas que hay que cumplir para relacionarlas entre sí, es decir se definen claves foráneas y primarias. Una clave primaria (o Primary Key PK) a un campo o a una combinación de campos que identifica de forma única cada fila de una tabla. Una clave primaria comprende de esta manera una columna o un conjunto de columnas. No puede haber dos filas en una tabla que tengan la misma clave primaria. Una clave foránea (o Foreign Key FK) es una limitación referencial entre dos tablas. La clave foránea identifica una columna o grupo de columnas en una tabla (tabla hija o referendo) que se refiere a una columna o grupo de columnas en otra tabla (tabla maestra o referenciada). Las columnas en la tabla referendo deben ser la clave primaria u otra clave candidata en la tabla referenciada. Los valores en una fila de las columnas referendo deben existir solo en una fila en la tabla referenciada. Así, una fila en la tabla referendo no puede contener valores que no existen en la tabla referenciada. De esta forma, las referencias pueden ser creadas para vincular o relacionar información. Esto es una parte esencial de la normalización de base de datos.

Para este proyecto se pensó un diagrama conceptual lo más genérico posible así que pueda ser adaptable a los requerimientos de diferentes clientes. En Figura 3.1 se observan cuatro *Ámbitos* che identifican el área de estudio, el caso típico es Provincia, Municipio, Distrito y el cuarto ámbito es el área que vamos a inventariar (ej. parque, jardín, playa, mobiliario urbano de una ciudad). El *Elemento* es el objeto específico identificado con un índice numérico (ej. Elemento 23), cada objeto pertenecerá a una *Clase* (ej. Clase árbol),

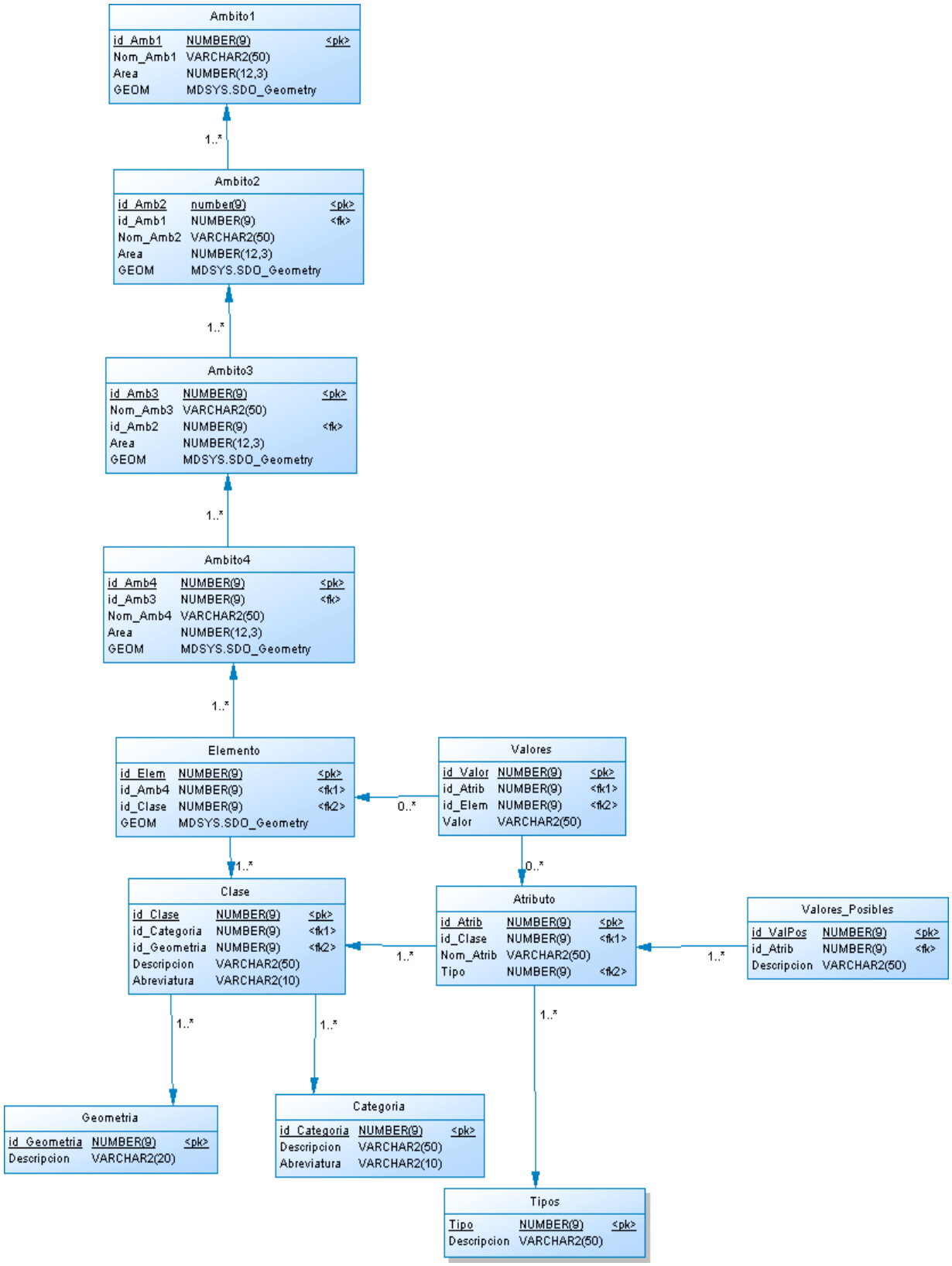


Figura 3.1 Diagrama conceptual y lógico de la base de datos

cada clase tendrá una *Geometría* (ej. punto) y pertenecerá a una *Categoría* (ej. vegetación). Las clases tienen una serie de *Atributos* (ej. Clase árbol: especie, diámetro, altura, fecha de poda etc.) y cada atributo tendrá *Valores* que se almacenaran en una nueva tabla que tendrá un enlace directo a la tabla de elementos. Se han añadido al final dos tablas una es la de *Tipos* que permite identificar de que tipo es el valor (numero, fecha, imagen, lista, lista desplegable) y la otra es *Valores Posibles* que almacena los posibles valores de un atributo (solo para atributos en el que el tipo es una lista o una lista desplegable).
Para pasar de diagrama conceptual a lógico (ver Figura 3.1) se han definido claves primarias y foráneas y también se han añadido a las entidades que tienen geometría el atributo GEOM que almacena las coordenadas x,y que permiten definir donde está el punto en el espacio. Se ha definido este atributo de tipo MDSYS.SDO_Geometry que es un tipo de variable suportada da Oracle que sirve para almacenar coordenadas.

3.2.2. Implementación de la base de datos

El primer paso es la creación de *tablespace*, usuario e esquema de la base de datos vacía en Oracle 11g. Para crear el *tablespace* hay que utilizar el usuario System definiendo nombre y anchura del mismo usando la siguiente sentencia SQL:

```
CREATE TABLESPACE "INVENTARIO_TBL"
LOGGING
DATAFILE 'E:\oradata\11g\INVENTARIO\INVENTARIO_01.ora' SIZE 1000M REUSE
AUTOEXTEND
ON NEXT  100M MAXSIZE  10000M EXTENT MANAGEMENT LOCAL;
COMMIT;
```

En la misma manera se define también un segundo *tablespace temp* (temporáneo) donde el programa trabajará antes del COMMIT con la misma anchura del precedente:

```
CREATE TABLESPACE " TEMP"
LOGGING
DATAFILE 'E:\oradata\11g\INVENTARIO\INVENTARIO_01.ora' SIZE 1000M REUSE
AUTOEXTEND
ON NEXT  100M MAXSIZE  10000M EXTENT MANAGEMENT LOCAL;
COMMIT;
```

Una vez creados los *tablepaces* se usa el siguiente script SQL para crear el *usuario* con sus permisos:

```
CREATE USER INV IDENTIFIED BY INV DEFAULT TABLESPACE "INVENTARIO_TBL"
TEMPORARY TABLESPACE "TEMP";
GRANT CONNECT,RESOURCE TO INVENTARIO;
COMMIT;
```

Lo script para generar automáticamente la estructura de la base de datos se ha generado en PowerDesign 12.5 a partir del diagrama lógico en Figura 3.1 y se ha ejecutado en la

consola del Toad for Oracle o también se pudiera haber ejecutado directamente en la consola de Oracle. Lo script está compuesto de una secuencia de CREATE TABLE + ALTERTABLE, el CREATE TABLE crea las tablas vacías y el ALTERTABLE sirve para crear las relaciones entre tablas.

```
ALTER TABLE INVENTARIO.NOMBRETABLA ADD
CONSTRAINT NOMBRETABLA_R01
 FOREIGN KEY (NOMBRE)
 REFERENCES INVENTARIO.NOMBRETABLARELACIONADA (NOMBRE)
 ENABLE
 VALIDATE
```

3.2.3. Carga de datos

La carga de datos ha sido manual directamente en la base de datos. Se han cargado uno datos alfanuméricos de prueba y como datos cartográfico un dibujo de AutoCad formato DWG.

3.3. Creación de la aplicación

3.3.1. Desarrollo de la aplicación con MapGuide

El diagrama de Figura 3.2 muestra el proceso de desarrollo de una aplicación basada en Web mediante MapGuide. En el diagrama, los rectángulos representan tareas, las formas ovales representan entidades creadas o utilizadas por las tareas y las flechas indican el flujo de los datos[6]. El proceso de desarrollo se puede descomponer en seis fases (aunque también exista una fase de planificación y preparación que no está incluida en el diagrama que llamaremos fase 0):

1) Cargar datos basados en archivos, configurar conexiones con bases de datos externas y extender los datos de elemento mediante la unión.
2) Crear capas que hagan referencias a los datos y les apliquen temas y estilos.
3) Crear mapas mediante la combinación de capas.
4) Colocar el mapa en Internet o en una intranet mediante el uso de capas para generar páginas Web.
5) Desarrollar la aplicación Web utilizando las API de MapGuide para añadir funciones.
6) Probar la aplicación Web.

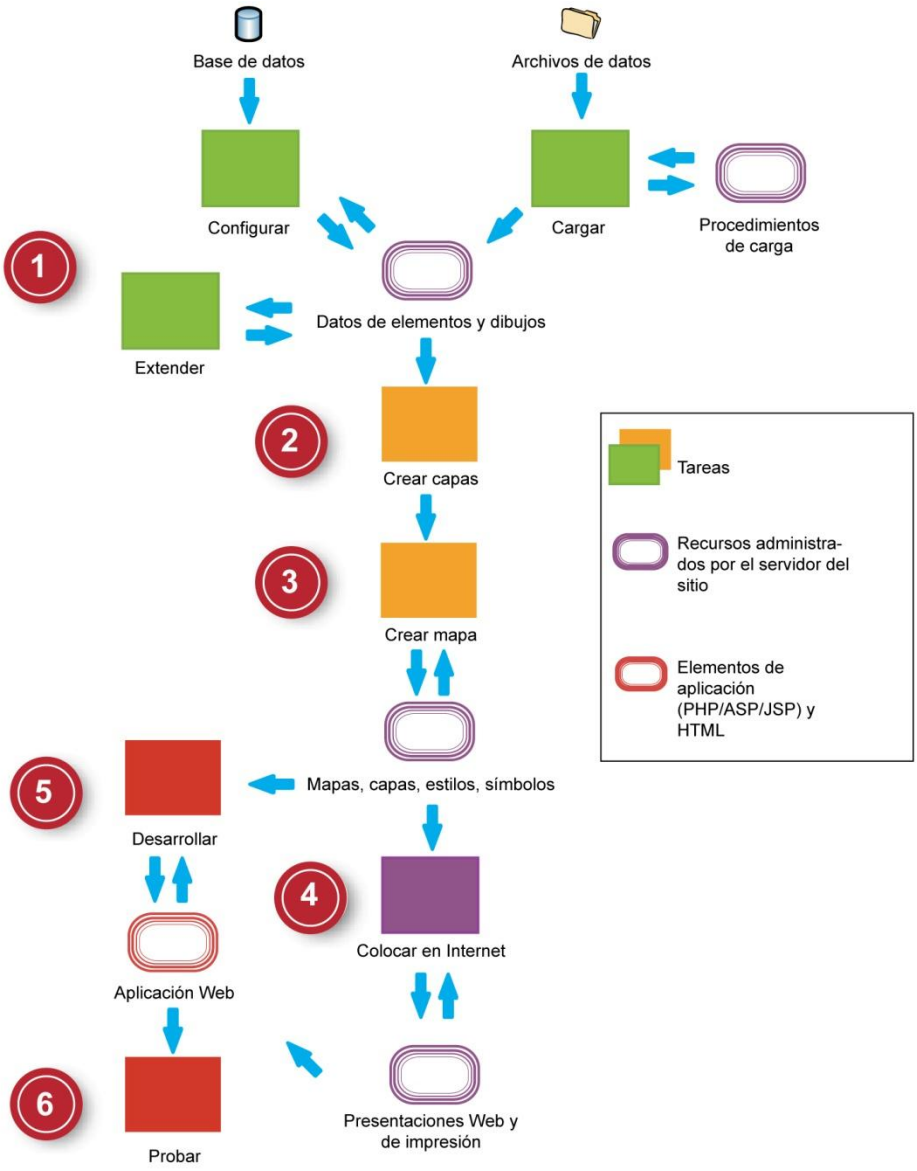


Figura 3.2 Desarrollo aplicación en MapGuide[6]

Sigue un análisis detallada de las diferentes fases de desarrollo por medio de diagramas de flujos. Los números de las fases de estos diagramas de flujo corresponden a los pasos numerados en Figura 3.2.

3.3.2. Fase 0: Planificación

La Figura 3.3 muestra el diagrama con los pasos que se deberían seguir en la fase de planificación.

Antes de programar una aplicación es muy útil preparar un diseño funcional y grafico basado en los objetivos específicos que la aplicación debería tener para satisfacer el cliente (ver Capitulo 2). Se deben contemplar todos los aspectos fundamentales:

- descripción de toda la funcionalidad de cada una de las herramientas a implementar;

- definición de la aplicación en cuanto a distribución de los diferentes elementos y como queremos que sean;
- diseño del aspecto visual que le queremos dar a nuestra aplicación.

Existen varios software que permiten hacer esto incluso se puede utilizar simplemente PowerPoint, en el caso del proyecto se ha utilizado Balsamiq® Mockups, un software de dibujo disponible en la empresa.

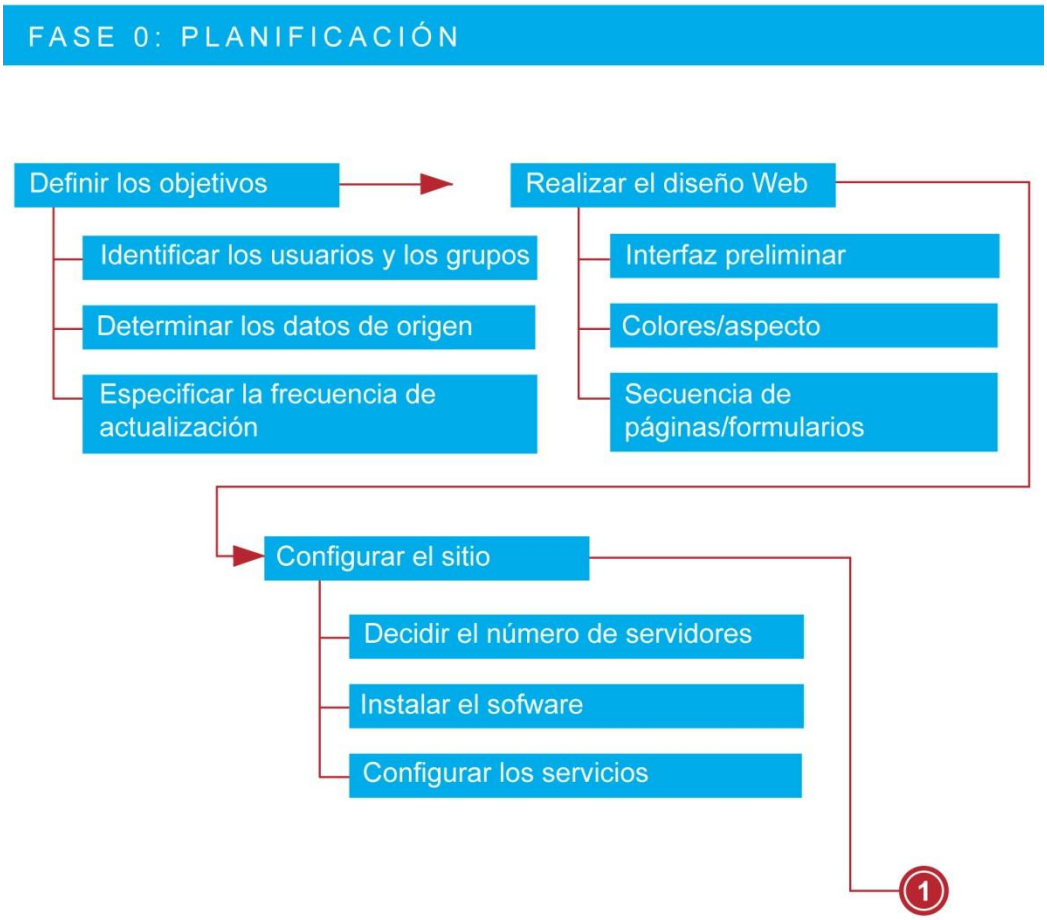


Figura 3.3 Fase de planificación

Para el dibujo del aplicación Inventario desde el principio se planteó la presencia de una página principal del visor con titulo, barra de herramientas MapGuide y leyenda a la izquierda (ver Figura 3.4). MapGuide proporciona por defecto una barra de herramientas (Figura 3.5) con las principales funcionalidades que necesita un visor (en el orden imprimir, medir, buffer, zoomin, zoomout, select, pan) a estas se pensó añadir dos formulario uno activable simplemente seleccionando un elemento en el mapa (formulario de identificación) y uno activable seleccionando un nuevo botón añadido manualmente a la barra de herramienta MapGuide (formulario de consulta), el botón evidenciado en Figura 3.5 con el círculo rojo es el botón añadido manualmente. Por defecto MapGuide proporciona también

una barra de zoom que se puede usar en alternativa a otros instrumentos de zoom y puede ser posicionada in cualquier parte del mapa.

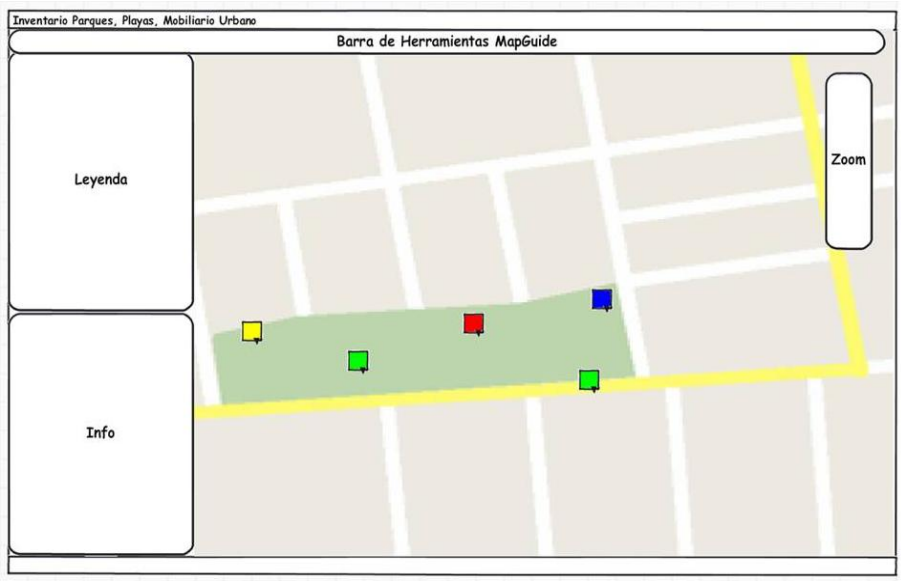


Figura 3.4 Diseño del visor

Figura 3.5 Barra de herramientas MapGuide

La Figura 3.6 muestra el formulario de identificación, hay un titulo que incluye los diferentes ámbitos para situar el entorno. Cada atributo tiene valores de diferentes tipos así que habrá que definir si son listas desplegables, cuados de texto, números o fechas a través de un formulario dinámico. Siempre habrá un atributo imagen y la posibilidad de guardar modificas. Para permitir de añadir atributos al mismo formulario se ha añadido un botón “Añadir Atributo” que activa un panel que permitirá escribir los valores da guardar en la base de datos.

La Figura 3.7 muestra el dibujo del formulario de consulta el cual presenta una lista que permitirá elegir la clase de elementos que el usuario quiere buscar en base de datos y una serie de listas desplegables para definir de manera más especifica la consulta. Se ha intentado de crear una interfaz de usuario muy simple de utilizar así que ha sido incorporado en la parte de abajo un cuadro de texto que va escribiendo la consulta mientras el usuario la define. Será luego el programa a trasformar esta consulta en lenguaje SQL que se pueda ejecutar en la base de datos.

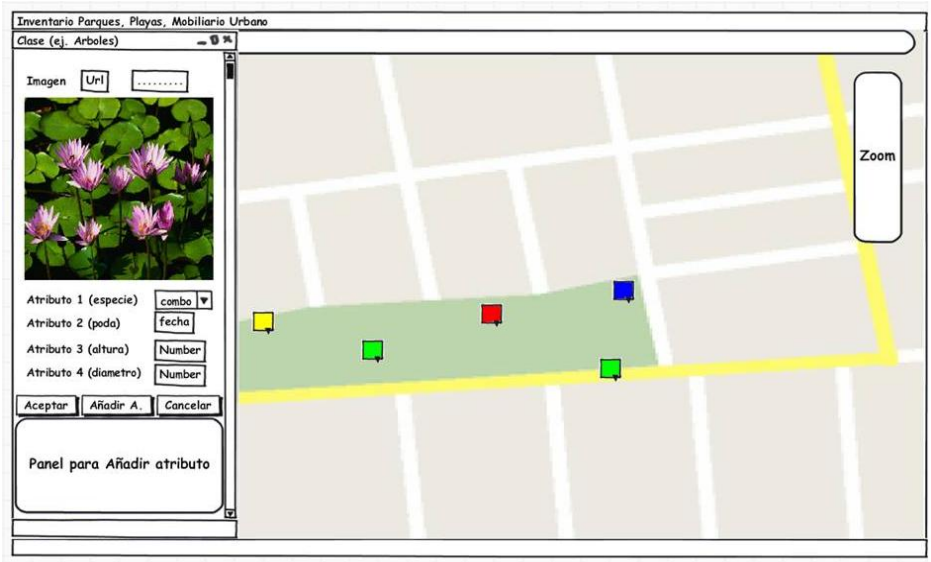


Figura 3.6 Formulario de identificación con panel para añadir atributo

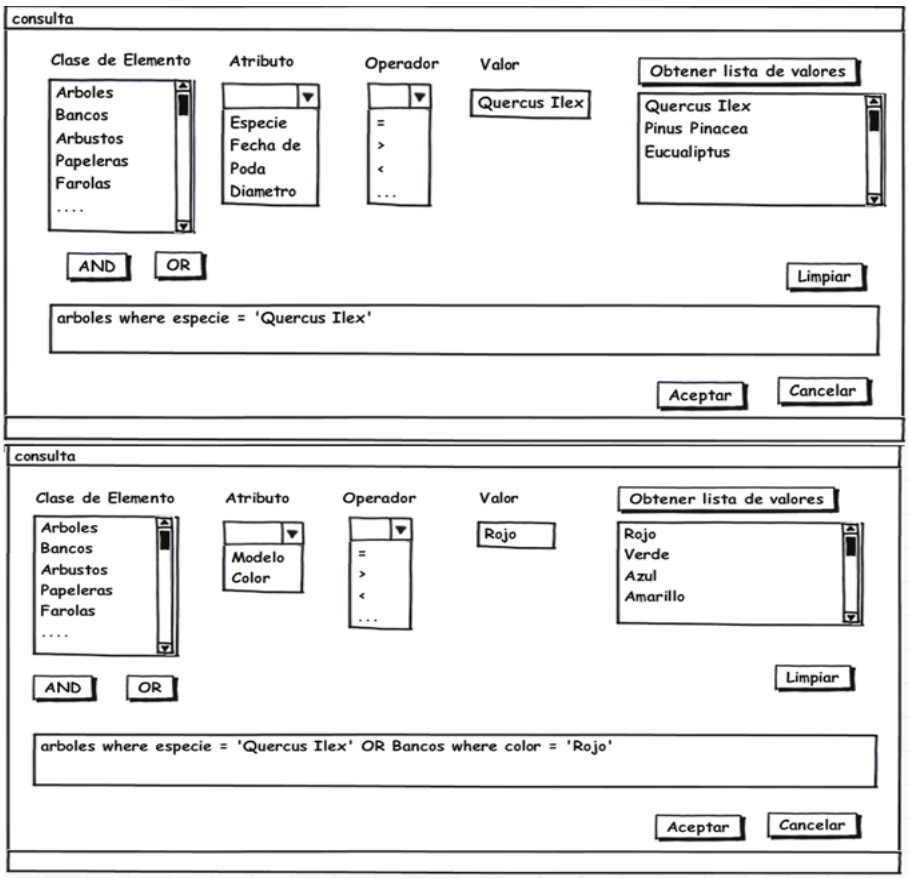


Figura 3.7 Formulario de consulta

La fase de planificación termina con la instalación del software y la configuración de los servicios.

3.3.3. Fase 1: Cargar y configurar

Antes de poder crear mapas con MapGuide, los datos de origen deben estar disponibles. Hay tres formas de lograr que los datos de origen estén disponibles (ver Figura 3.8):

- Cargar los datos basados en archivos, como DWG, SHP o SDF en el sitio MapGuide. También se puede cargar archivos ráster.
- Conectar con archivos SDF, SHP y ráster que no estén cargados en el repositorio de recursos de MapGuide.
- Configurar conexiones con bases de datos, como Oracle, ArcSDE o MySQL.

En este proyecto se ha trabajado con capas DWG para los datos cartográficos y configurando una conexión a la base de datos creada en Oracle 11 para almacenar los datos alfanuméricos.

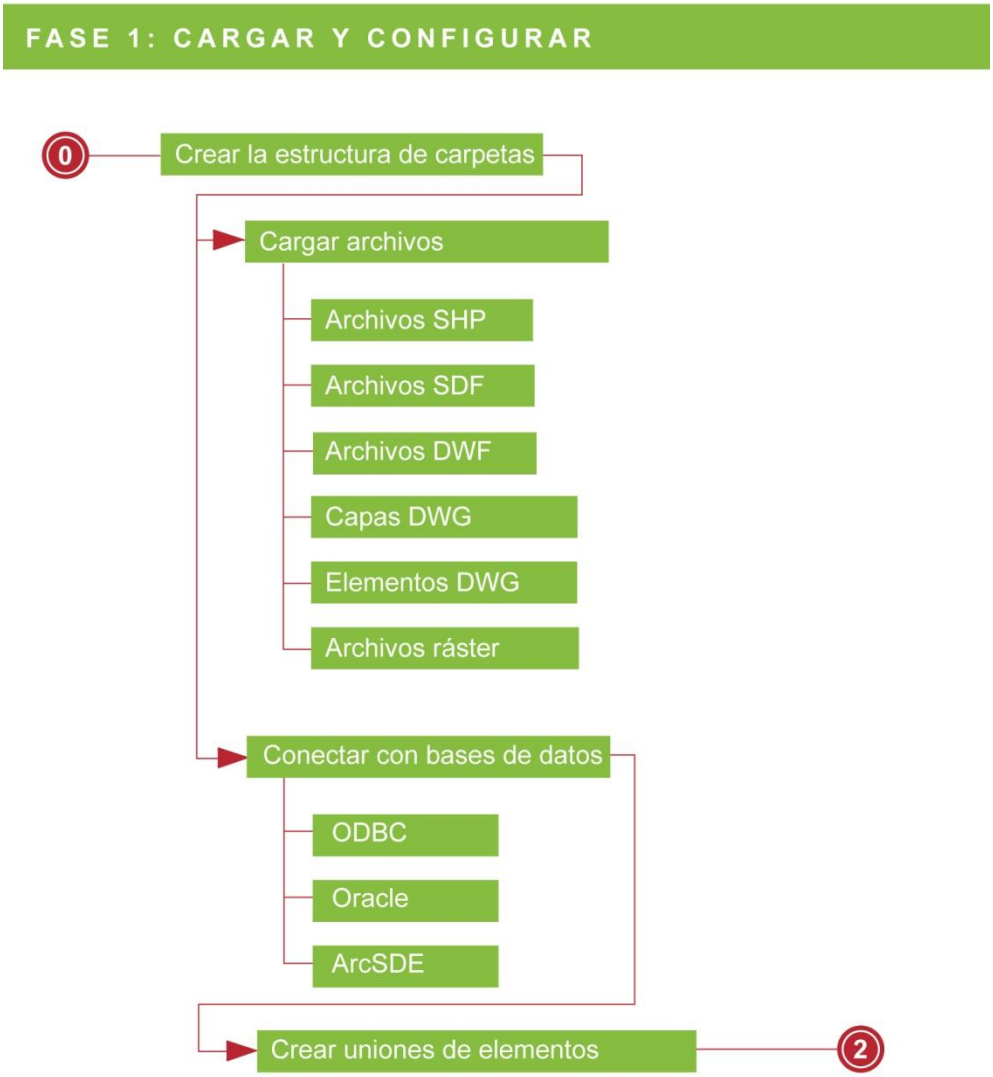


Figura 3.8 Cargar y configurar

3.3.4. Fase 2-3: Crear capas y mapas

Es esta la fase en la que se crean capas a partir de los datos cargados y mapas agregando las capas, el diagrama en Figura 3.9 muestra la secuencia de operaciones que se incluyen en esta fase. En el proyecto no se ha dado particular importancia a esta fase para concentrarse más en el desarrollo de las funcionalidades de la aplicación a nivel de código. Se recuerda que se ha trabajado con datos de pruebas.

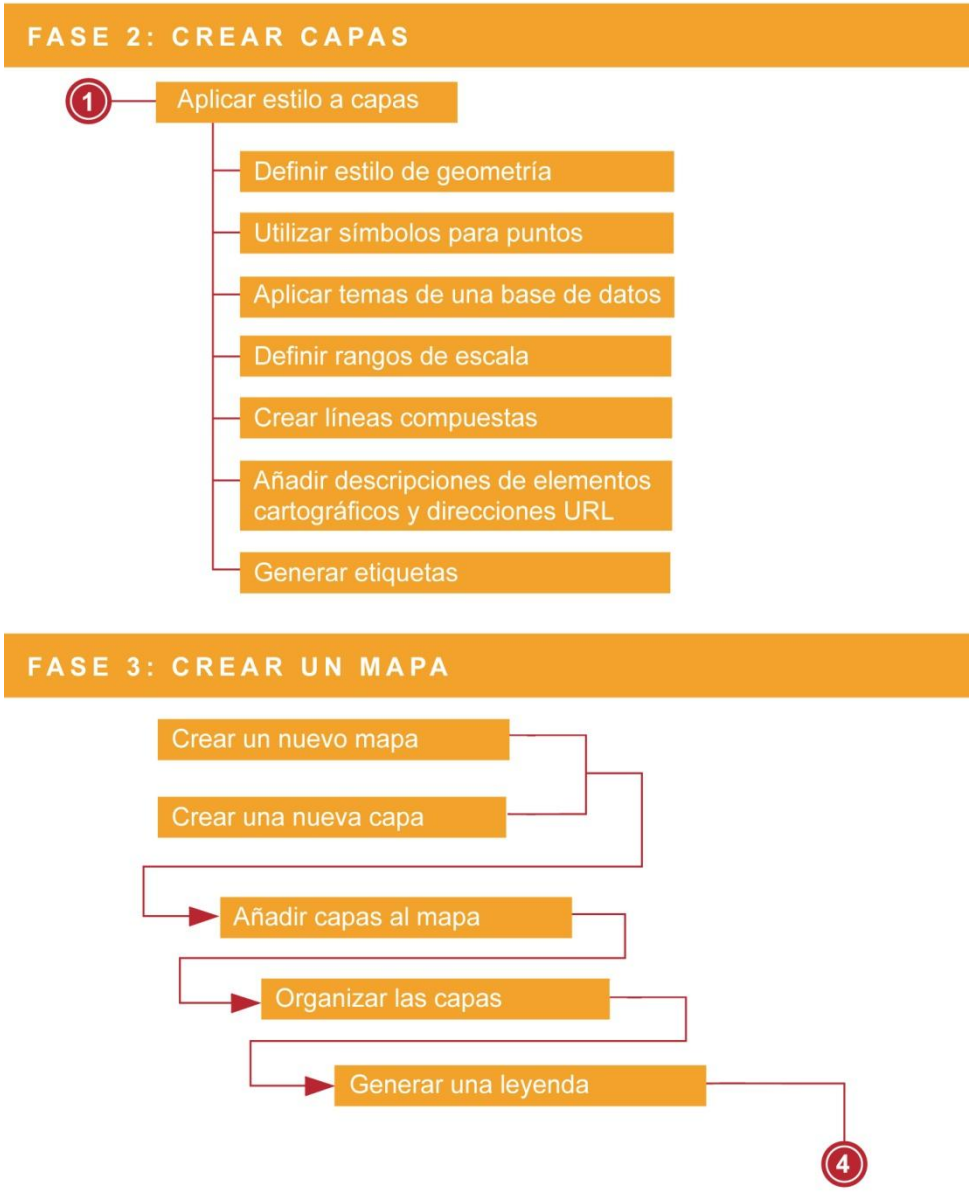


Figura 3.9 Crear capas y mapas

3.3.5. Fase 4: Colocar en Internet

Una vez se haya definido las capas y creado el mapa es necesario definir un *Layout* di presentación en la página web. MapGuide permite personalizar el visor directamente desde

el programa. Se pueden añadir o quitar botones en la barra de herramientas, decidir dónde poner la barra de zoom y crear o quitar espacios (*div*) laterales respecto al mapa para insertar la leyenda o nuevos formularios. MapGuide también tiene una galería de *Layout* pre configurados que se pueden utilizar tal cuales. En este proyecto se ha preferido configurar un *Layout* nuevo que se enseñará directamente en el capítulo dedicado a la presentación de los resultados (ver Capitulo 4). Una vez configurado el *Layout* será posible visualizarlo en un explorador digitando la URL del *localhost*. La Figura 3.10 muestra el diagrama para la fase 4.

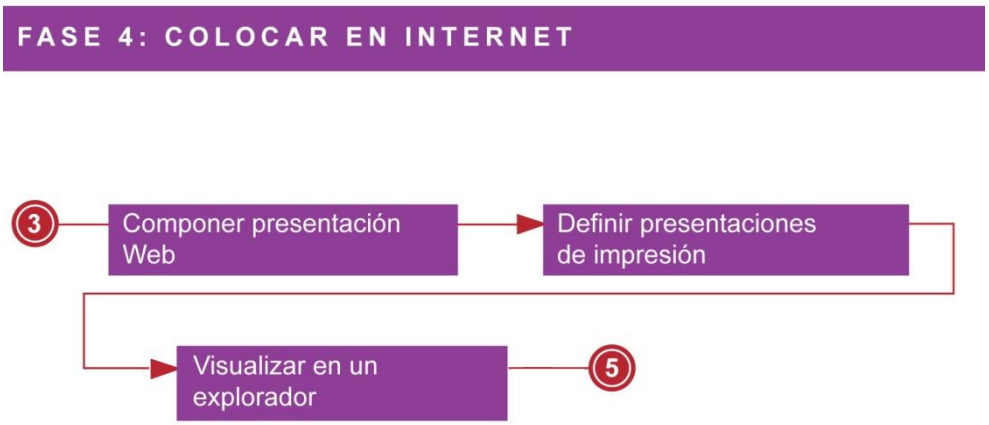


Figura 3.10 Colocar en internet

3.3.6. Fase 5-6: Desarrollar y probar

En este párrafo se profundizaran las fases 5-6 (ver Figura 3.11).

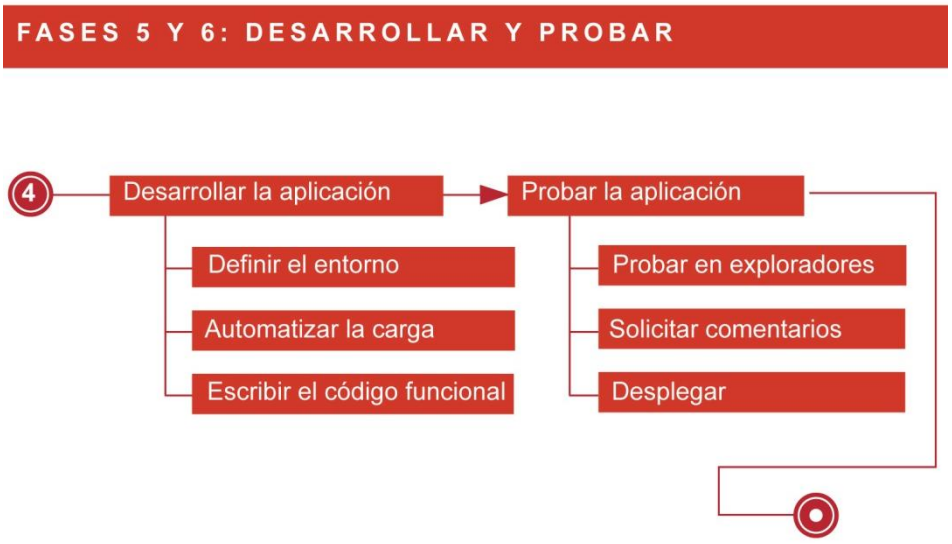


Figura 3.11 Desarrollar y probar.

Para desarrollar la aplicación se ha usado el entorno de desarrollo Microsoft® Visual Studio 2008. Los lenguajes de programación que se han utilizado para programar la aplicación son Visual Basic y Java Script y para el dibujo de los formularios HTML.

Para mantener independientes la interfaz de usuario y la lógica asociada a la aplicación, la implementación de las páginas, ASP.NET la divide en dos ficheros. En un fichero con extensión .aspx en el cual se especifica el aspecto de nuestra interfaz, utilizando tanto etiquetas HTML estándar como etiquetas específicas para hacer referencia a los controles ASP.NET que se desea incluir en la página web. En un segundo fichero, que será un fichero de código con extensión .vb si se utiliza en lenguaje *visualbasic*, se implementará la lógica de la aplicación. Visual Studio se ocupará de enlazar el código escrito por el programador con los componentes de la interfaz[7].

La Figura 3.12 muestra la división en carpetas en el explorador de soluciones de Visual Studio, las paginas nombradas Default.aspx, Global.aspx y las carpetas App.cod, App.data, Bin vienen por defecto utilizando ASP.NET, son carpetas que contienen código de configuraciones extra que el programador puede se modificar o dejar tal cuales. En el detalle:

- El fichero Default.aspx es la página que se sirve cuando los usuarios navegan a su sitio sin especificar una página determinada. Se puede configurar la aplicación para que la página Default.aspx se solicite automáticamente pero no es obligatorio. Se puede usar también como página de inicio para el sitio, o se puede escribir código en la página para redirigir a los usuarios a otras páginas[5].
- El archivo Global.asax, a veces llamado el archivo de aplicación ASP.NET, está en el directorio raíz de la aplicación. Aunque Visual Studio. NET lo inserta automáticamente en todos los proyectos nuevos, en realidad es un archivo opcional, se puede utilizar para aplicar la seguridad de aplicaciones, así como otras tareas. Si el archivo Global.asax está configurado cualquier solicitud HTTP directa (a través una URL) viene rechazada automáticamente, por lo tanto los usuarios no pueden descargar o ver su contenido. El marco (framework) de páginas ASP.NET reconoce automáticamente todos los cambios que se realizan en el archivo Global.asax[5].
- La carpeta App_code contiene el código fuente para las clases comunes y los objetos business (por ejemplo CS y vb files) si el programador desea puede modificarlo como se fuera una parte su aplicacion. En un proyecto Web ASP.NET siempre compilará el código en la carpeta App_Code automaticamente en la solicitud inicial de la aplicación[5].
- La carpeta App_data contiene los archivos de datos de la aplicación incluyendo los archivos .mdf (*database file*), los archivos XML y otros archivos de almacén de datos. La carpeta App_Data es utilizada por ASP.NET para almacenar base de datos locales de una aplicación[5].

- La carpeta Bin contiene *compiled assemblies* (archivos .dll) para controles, componentes u otro código que desea hacer referencia en su aplicación. Cualquier clase representada por el código en la carpeta Bin se hace referencia automáticamente en la aplicación[5].

Las otras carpetas que no se han nombrado son las creadas por mano del autor y dependen de cada proyecto, en este proyecto hay:

- El fichero web.config con el cual se configura el e enlace al servidor Web y a la base de datos. La automatización de la carga (Figura 3.11) se debe a este fichero que permite modificare las impostaciones básicas sin haber físicamente acceso a los servidores.
- El Title.aspx, el Form_Elemento.aspx y el Form_Consulta.aspx son los formulario creados para la aplicación, son los formularios en los cual se desarrollan las funcionalidades de la aplicación. Cada uno incluye una página .vb que contiene el código fuente.
- La carpeta IMAGEN contiene todas las imágenes .jpeg y .png que se quieren enseñar en la página web, en la base de datos solo se han guardado las URLs de direccionamiento a esta carpeta.

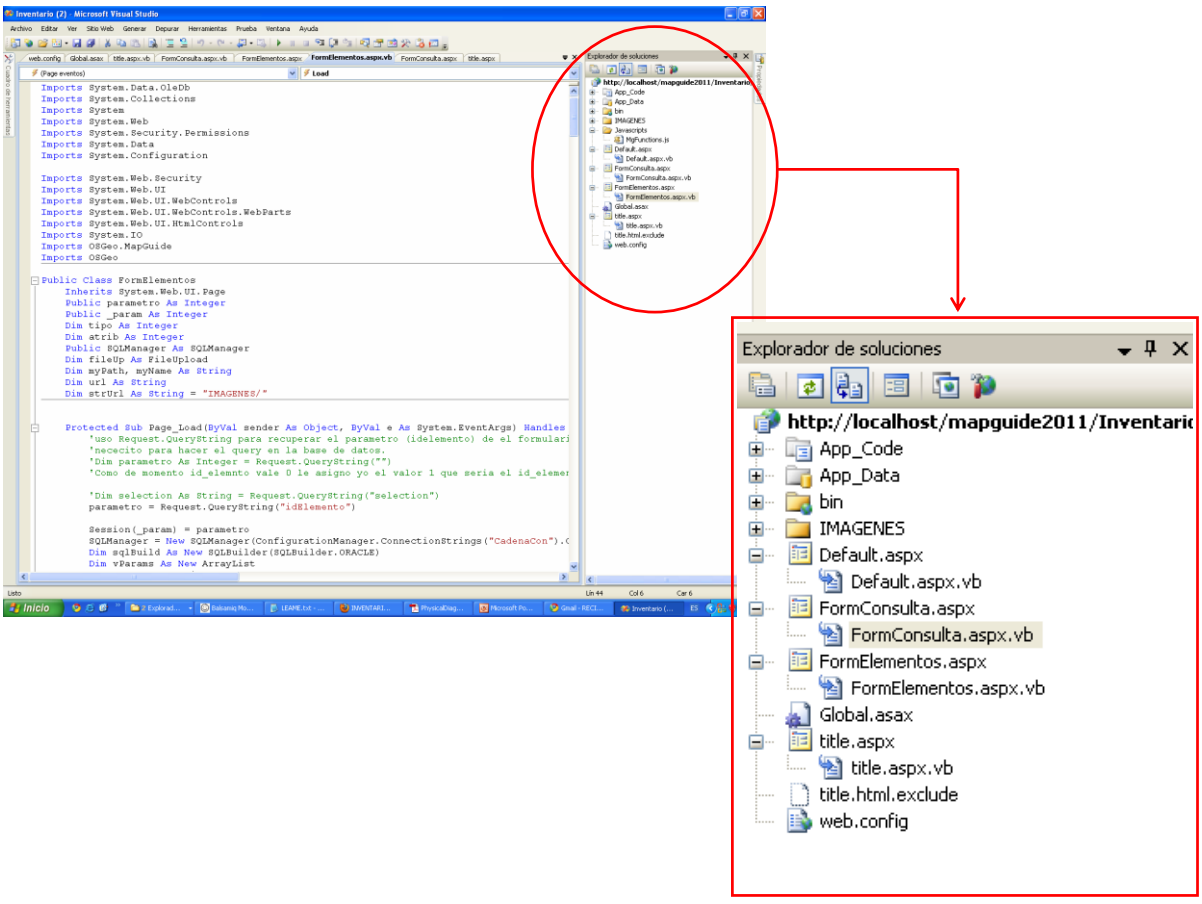


Figura 3.12 Explorador de soluciones Visual Studio 2008

Definido el esquema de trabajo se pasa a escribir el código para desarrollar las funcionalidades. Este proceso implica probar la aplicación mientras se programa directamente ejecutando el código en el explorador en la manera de solucionar cada error y problemática usando el interfaz de usuario que proporciona Visual Studio 2008. Al fin de tener un código claro y ordenado se han incluido comentarios y agrupados los procedimientos en regiones.

3.3.7. Explicación del código

En este párrafo se explicaran y motivaran algunas partes del código fuente de la aplicación desarrolladas por el autor.

Una aplicación web, generalmente, está formada por varios formularios web, en el caso de este proyecto hay tres: Title.aspx, Form_Elemento.aspx y el Form_Consulta.aspx. Cada uno de esos formularios se implementa como una páginas ASP.NET. En la plataforma .NET, las páginas ASP.NET se construyen creando clases derivadas de la clase System.Web.UI.Page. Dicha clase proporciona la base sobre la que se construirán las páginas ASP.NET, que se implementan como subclases de System.Web.UI en las que se incluyen las funcionalidades requeridas por nuestras aplicaciones. Las primeras líneas de código que se escriben en la página .vb de un nuevo formulario web sirven para declarar esta clase, claramente siempre será posible añadir nuevas pero las que son fundamentales son:

```
Imports System
Imports System.Web.UI
```

A esta se han añadido unas clases que se necesitaban para trabajar con MapGuide y con Oracle 11:

```
Imports OSGeo.MapGuide
Imports System.Data.OleDb
Imports SEYSClases
```

SEYSClases es una clase desarrollada en Seys específica para trabajar con las base de datos, permite crear de manera ágil sentencias SQL para buscar, modificar y añadir datos en la base de datos, en términos técnicos construye sentencias de SELECT, ALTER TABLE y INSERT INTO.

El modelo de programación de los formularios web con ASP.NET está basado en el uso de controles y eventos así que es necesario declarar también las bibliotecas de clases que permiten utilizar dichos controles:

```
Imports System.Web.UI.WebControls
Imports System.Web.UI.HtmlControls
```

Estas clases proporcionan una amplia gama de controles predefinidos como botones, cajas de texto, listas o tablas. Dentro de las páginas ASP.NET, los controles se indican en el fichero .aspx utilizando etiquetas de la forma <asp:.../>. La sintaxis general de una etiqueta ASP.NET es de la siguiente forma:

```
<asp: control id="identificador" runat="server"/>
```

Utilizando Visual Studio .NET como entorno de desarrollo los controles no se colocan en coordenadas fijas (a diferencia de la forma habitual de trabajar con formularios Windows) entonces para mover, desplazar y posicionar controles se ha utilizado una tabla html <Table></Table> donde cada celda, definida por medio de declaración de <tr></tr> y <td></td>, contiene un control o un espacio vacío y para que el contenido de las celdas aparezca centrado se han utilizado las opciones de alineamiento del código html (allign y vertical allign), en este manera se han podido construir formularios complexos como por ejemplo el Form_Consulta.aspx que contiene muchos controles alineados y ordenados.

Una vez que un control se ha posicionando en el formulario web, lo normal es que se modifique su identificador (propiedad ID) y el texto que muestra en el formulario (propiedad Text), estas modificas se pueden hacer directamente usando la ventana de propiedad de Visual Studio o se pueden definir usando el código, más adelante se verá que no siempre se puede utilizar la ventana de propiedad.

De todo lo que genera automáticamente el Visual Studio al insertar un control en la pagina .aspx, el detalle más significativo es la inclusión del atributo runat="server" en todos aquellos componentes de la interfaz a los que podemos asociarles manejadores de eventos que implementen la lógica de la aplicación como respuesta a las acciones del usuario. Dichos manejadores de eventos se ejecutarán siempre en el servidor y serán los encargados de que la aplicación sea algo más que páginas estáticas en las que se muestra información. Todos los controles en una página ASP.NET deben estar dentro de una etiqueta <form> con el atributo runat="server". Además, ASP.NET requiere que todos los elementos HTML estén correctamente anidados y cerrados (como sucede en XML). Los controles utilizados en el proyecto se resumen en Tabla 3.1.

A parte los controladores típicos que se pueden fácilmente individuar en los formularios del proyecto se ha utilizado el PlaceHolder, un objeto que se utiliza cuando se desea asociar a un grupo de controles de una página un comportamiento dinámico. El control PlaceHolder no produce ningún resultado visible y sólo se utiliza como contenedor para otros controles en la página Web. Se puede utilizar la colección Control.Controls para añadir, insertar o quitar un control en el control de marcador de posición.

Tabla 3.1 Controles utilizados en el proyecto

Control	Descripción
Button	Botón estándar
DropDownList	Lista desplegable
Grid View	Tabla
Image	Imagen
ImageButton	Botón dibujado con una imagen
Label	Etiqueta de texto estático
ListBox	Lista
Panel	Contenedor en el que se pueden colocar otros controles
PlaceHolder	Reserva espacio para controles añadidos dinámicamente
TextBox	Caja de edición

La asignación de las propiedades a los controles añadidos no se podrá hacer usando la ventana de propiedad sino exclusivamente por código. Lo único que se declara en la pagina .aspx es el `PlaceHolder`. El en proyecto se ha asignado al `PlaceHolder` el nombre "Container", para añadir un control (per ejemplo un Label) el codido será:

```
Dim Lbl As New Label
Container.Controls.Add(Lbl)
```

Las propiedades del Label se definirán por código, por ejemplo para asignar un texto y un color de fondo el código será:

```
Lbl.Text = "Especie"
Lbl.BackColor = Drawing.Color.LightGray
```

En el proyecto se ha utilizado el `PlaceHolder` en el Form_Elemento.aspx para gestionar el formulario de identificación de manera dinámica dado que cada elemento puede tener un número variable de atributos y cada atributo puede requerir diferentes controles (cajas de texto, listas o imágenes) dependiendo de cómo están almacenados los valores la base de datos. En el Form_Consulta.aspx se ha utilizado para enseñar dinámicamente los `GridView` resultado de la consulta hecha por el usuario. Ha sido necesario utilizar un `PlaceHolder` dado que la consulta puede devolver elementos que pertenecen a distintas clases y también los `GridView` pueden tener un número variable de fila y de columnas dependiendo de cuantos elementos se han encontrado de la misma clase y de cuantos atributos tiene cada elemento.

Una vez que se haya construido la interfaz grafica de un formulario lo primero que hay que programar es la respuesta del formulario al evento Page_Load que se produce cuando el usuario accede a la página ASP.NET desde su navegador. Al solicitar una página ASP.NET desde un cliente, en el servidor se dispara el evento Page_Load asociado a la

página antes de generar ninguna salida. Es en el manejador asociado a este evento donde se realizan las tareas de inicialización de la página. Dichas tareas incluyen en los formularios del proyecto el establecimiento de valores por defecto y el rellenado de las listas de valores que han de mostrarse al usuario.

El modo de interacción característico de las interfaces web introduce algunas limitaciones. Estas limitaciones se deben a que cada acción que el usuario realiza en su navegador se traduce en una solicitud independiente al servidor web. El hecho de que cada solicitud recibida por el servidor sea independiente de las anteriores ocasiona que, al desarrollar aplicaciones web, hay que ser conscientes de cuándo se produce cada solicitud y de cómo se pueden enlazar solicitudes diferentes realizadas por un mismo usuario. El evento Page_Load se dispara cada vez que el usuario accede a la página. Para realizar una tarea sólo la primera vez que un usuario concreto accede a la página, la única posibilidad es la de utilizar la propiedad `IsPostBack`. Esta propiedad posee el valor *false* cuando el cliente visualiza por primera vez la página ASP.NET, mientras que toma el valor *true* cuando no es la primera vez que la página ha de ejecutarse para ser mostrada. Esto sucede cuando el usuario realiza alguna acción, como pulsar un botón del formulario web, que tiene como consecuencia volver a generar la página para presentar datos nuevos o actualizados en la interfaz de usuario.

En los formularios del proyecto muchas veces se ha necesitado mostrar los datos en el mismo momento en que el usuario selecciona un elemento de una lista desplegable (`DropDownList`). De esta forma, la respuesta inmediata de la aplicación facilita que el usuario perciba el efecto de las acciones que realiza. En el caso de la lista desplegable, hay que implementar la respuesta del control al evento `SelectedIndexChanged`, pero esto no es suficiente para que la aplicación responda de forma inmediata a un cambio en el elemento seleccionado de la lista desplegable. Para que la página se actualice en el mismo momento en que el usuario selecciona un elemento de la lista, será necesario establecer a `true` la propiedad `AutoPostBack` del control de tipo `DropDownList`. Dicha propiedad es disponible para todos los controles más utilizados, en el proyecto ha sido utilizada prevalentemente para `DropDownList, ListBox y TextBox.`

Como se ha dicho antes cada vez que un servidor Web ejecuta una solicitud HTTP para una página se produce una solicitud independiente El servidor no tiene ningún conocimiento de valores de las variables que se utilizan durante las solicitudes anteriores y la aplicación en este manera pierde informaciones. Para evitar de tener que volver a escribir el código se han utilizado variables de sesión. Las variables de sesión se almacenan en un objeto `SessionStateItemCollection` que se expone a través de

la propiedad `HttpContext.Session`. En una página ASP.NET, las variables de sesión actual se exponen a través de la propiedad `Session` del objeto Page. Las variables de sesión se declaran como *Public* en el evento Page_Load y se crean en el código en cualquier procedimiento asignándoles el valor de la variable que se desea guardar en las siguientes sesiones. Sigue un ejemplo extracto del código fuente del Form_Elemento.aspx, en este caso se quería guardar el valor del id_Elemento recuperado en el Page_Load para utilizarlo en otros procedimientos:

```
Protected Sub Page_Load (ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load

Dim parametro As Integer
Public _param As Integer

parametro = Request.QueryString("idElemento")
Session(_param) = parametro

End Sub
```

En el siguiente capítulo se exponen los resultados donde también se podrán comprobar lo que se ha obtenido por medio de los objetos explicados en este párrafo.

4. RESULTADOS

4.1. Presentación de la aplicación

Figura 4.1 muestra como aparece la aplicación, la barra de herramienta y de zoom son las típicas de MapGuide como también el aspecto de la leyenda. La barra de estado es posicionada y rellenada directamente por el programa con escala, coordenadas y informaciones sobre los elementos seleccionados.

Se han cargado cuatro ámbitos y seis elementos de prueba para verificar la funcionalidad de la aplicación (tres árboles, una farola, un banco y una papelera).

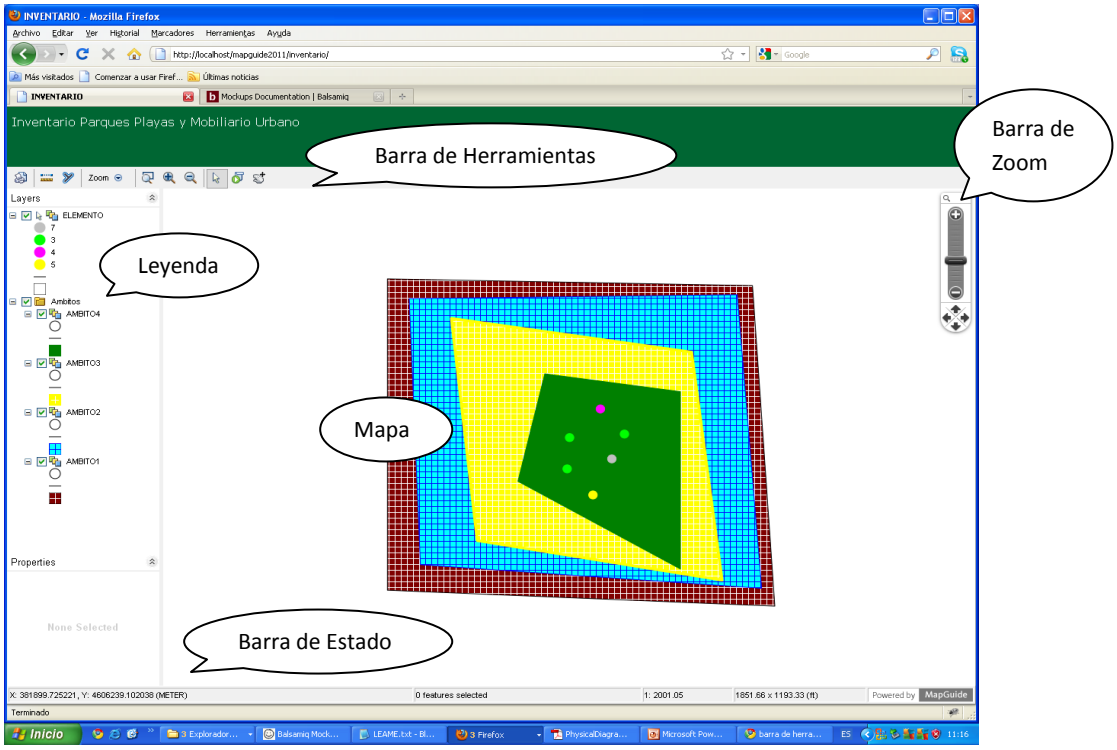


Figura 4.1 Aspecto visor

La Figura 4.2 muestra el funcionamiento del formulario de identificación. Seleccionando un elemento puntual en el mapa se abre una nueva página web que enseña un formulario dinámico (ver explicación del código, Capitulo 3.3.7), diferente para cada elemento. El programa reconoce cuantos atributos hay y crea un formulario adecuado al formato con el cual se han guardado los valores en base de datos: lista, lista desplegable, cuadro de texto, fecha, numero (ver detalle en Figura 4.3). Los elementos lineales y poligonales no son seleccionables.

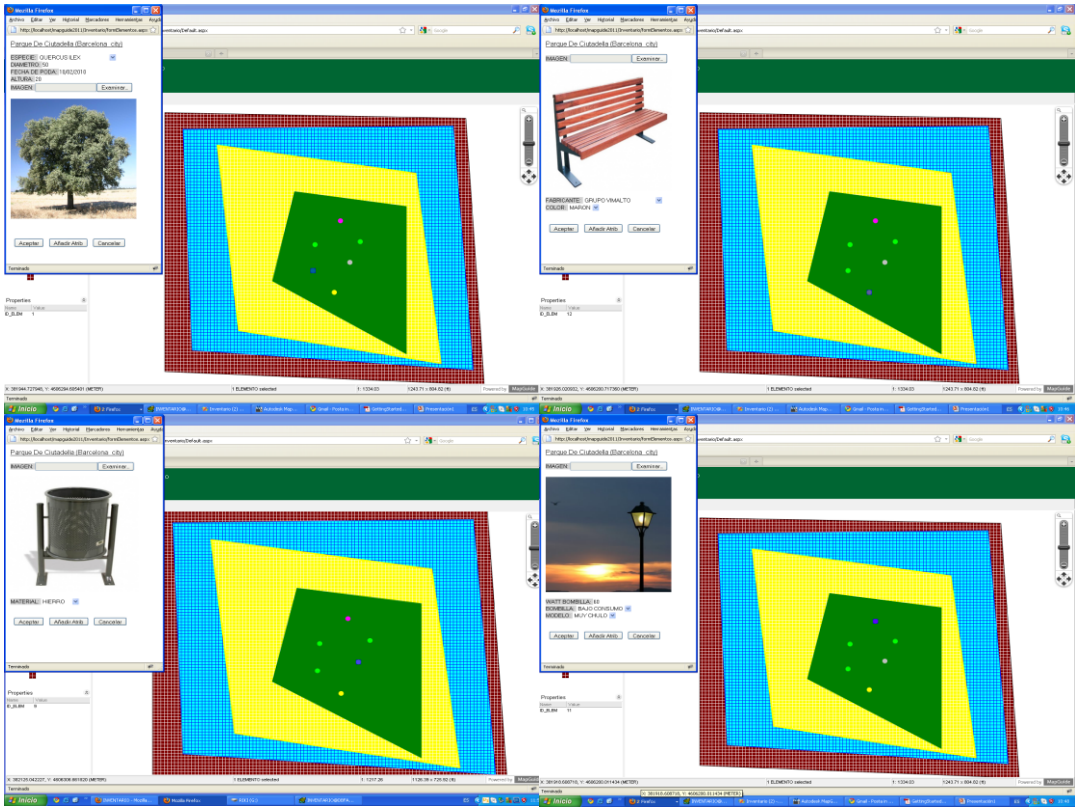


Figura 4.2 Formulario de identificación

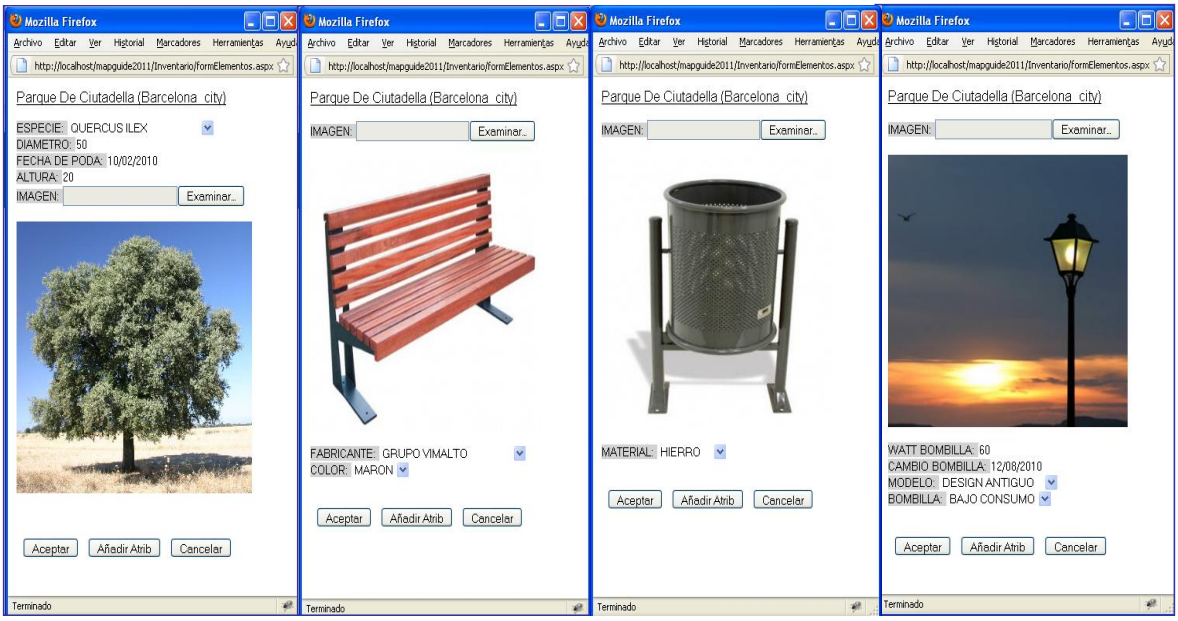


Figura 4.3 Detalle, formulario dinámico

La modifica de los valores puede ser enseguida directamente en el mismo formulario, en este caso será necesario editar los valores seleccionando una de las opciones si el atributo tiene los valores agrupados un una lista desplegable o digitando un nuevo valor en el caso se trate de un cuadro de texto, fecha, imagen o numero. Una vez modificado el valor habrá que apretar el botón Aceptar (ver Figura 4.4) y automáticamente el valor será guardado en base de datos.

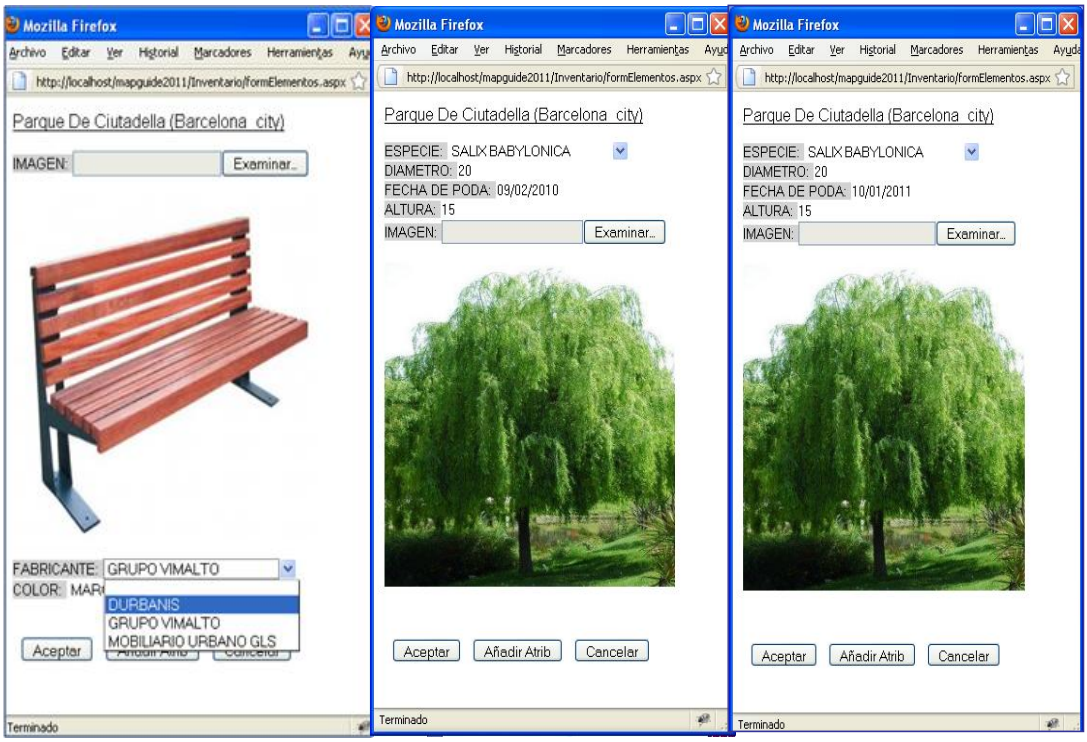


Figura 4.4 Modifica valores en el formulario, modifica de una lista desplegable para el elemento "banco" y modifica de un cuadro de texto (fecha de poda) para el elemento árbol.

La funcionalidad "Añadir Atributo" es dada seleccionando el botón central del formulario de identificación ("Añadir Atributo"), al apretar este botón se activa un panel que permite insertar los valores. El panel preguntará al usuario primero de que tipo es el nuevo atributo, segundo pedirá de escribir el nombre del atributo y al final dependiendo del tipo que el usuario ha elegido habrá dos procedimientos. En el caso el usuario elija cuadro de texto, fecha, numero o imagen simplemente deberá escribir en el campo "Valor" el nuevo dato que desea insertar, en este caso será desactivado el botón "+" y el campo "Lista de Valores". En la base de datos el valor será guardado directamente en la tabla "Valores". En el caso el usuario elija lista o lista desplegable se activará la posibilidad de añadir los valores a una lista che vendrá guardada en base de datos en la tabla "Valores_Posibles".
La Figura 4.5 muestra un ejemplo en el cual se añade un atributo al elemento "Papelera"; el nuevo atributo es de tipo "numero". En Figura 4.6 se ve que el usuario simplemente inserta dos datos: el nombre del nuevo atributo ("Diametro") y el valor ("40").

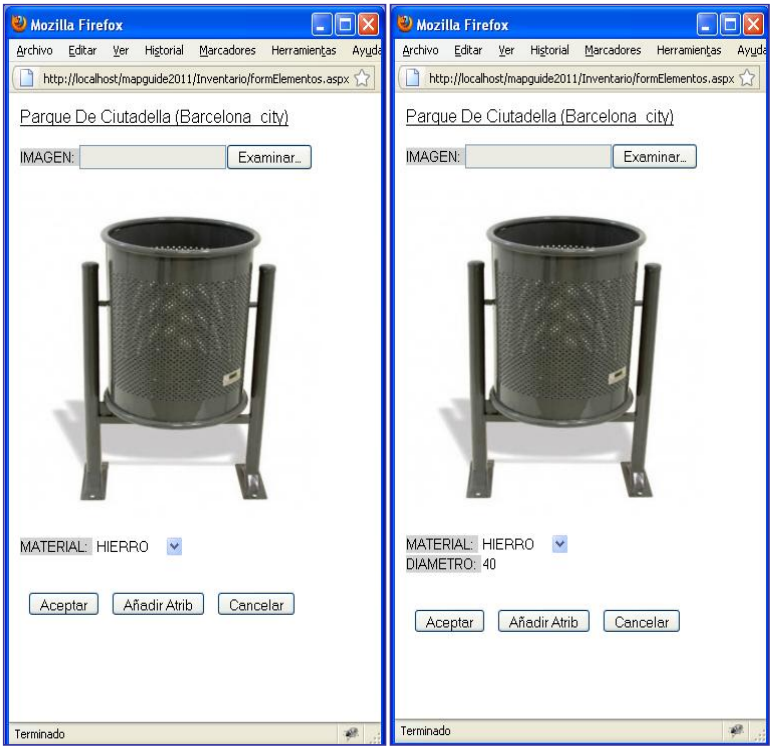


Figura 4.5 Añadir atributo (ejemplo 1), formulario inicial a la derecha y resultado final a izquierda.

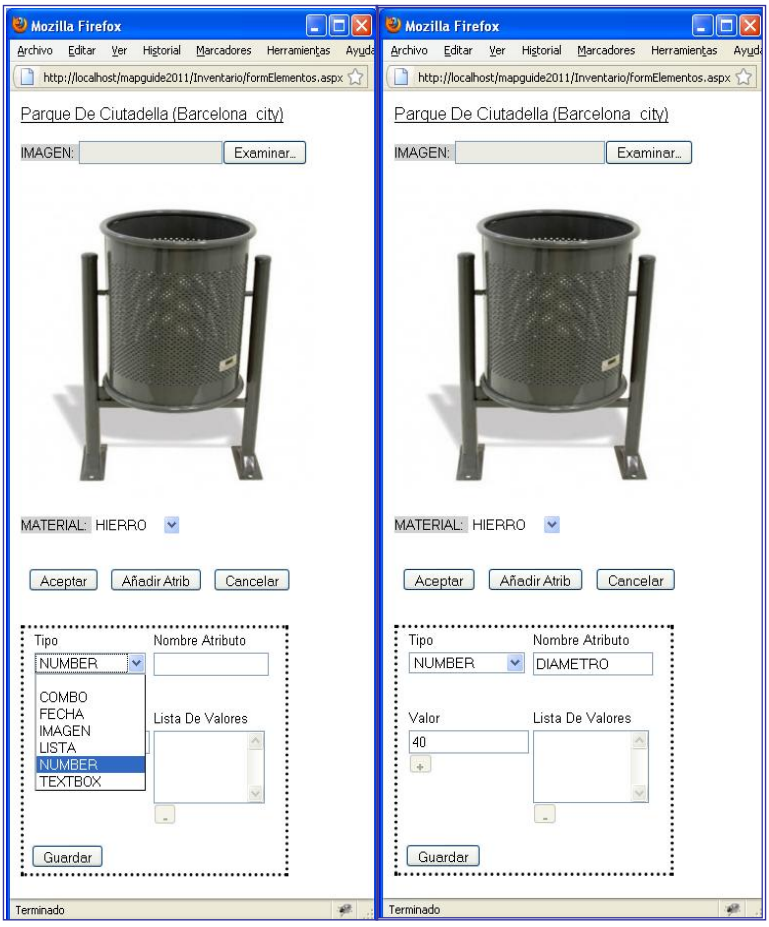


Figura 4.6 Añadir atributo (ejemplo 1), panel para insertar los valores

La Figura 4.7 muestra un segundo ejemplo en el cual se añade un atributo al elemento “Banco”, en este caso el nuevo atributo es de tipo “combo” y entonces el usuario una vez terminado de escribir el primer dato apretando el botón “+” lo añade al campo “Lista de Valores“. Cuando la lista está completa seleccionando el botón “Guardar” se guardaran los datos en la base de datos. El resultado final se muestra en Figura 4.8.

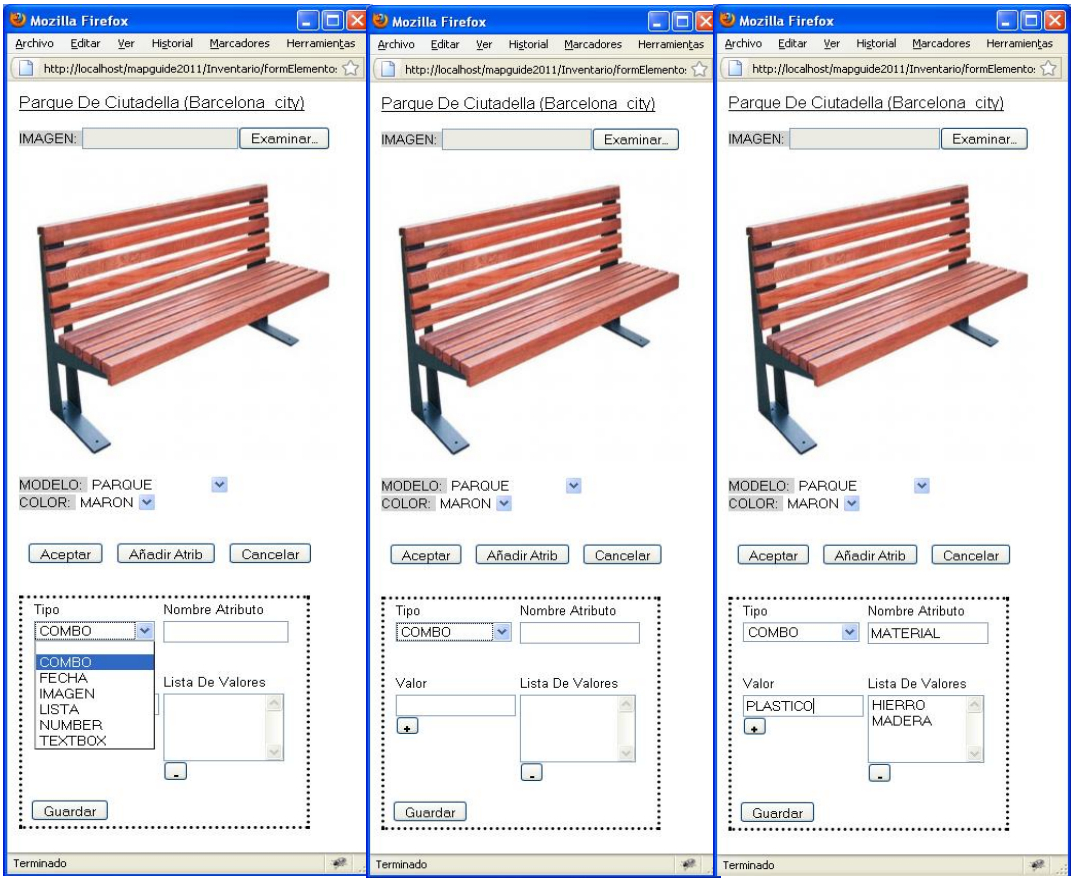


Figura 4.7 Añadir atributo (ejemplo 2), panel para insertar los valores

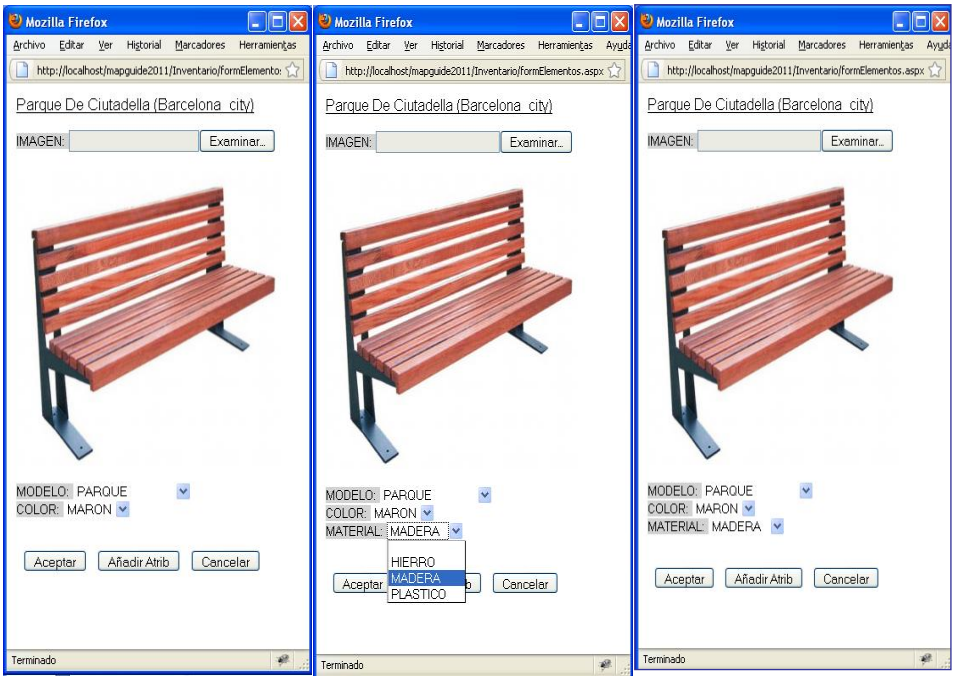


Figura 4.8 Añadir atributo (ejemplo 2), formulario inicial a la derecha y resultado final a izquierda.

El visor permite la consulta en la base de datos por medio de un formulario que podrá ser activado a través un botón de la barra de herramienta añadido exclusivamente por esta funcionalidad (ver Figura 4.9).

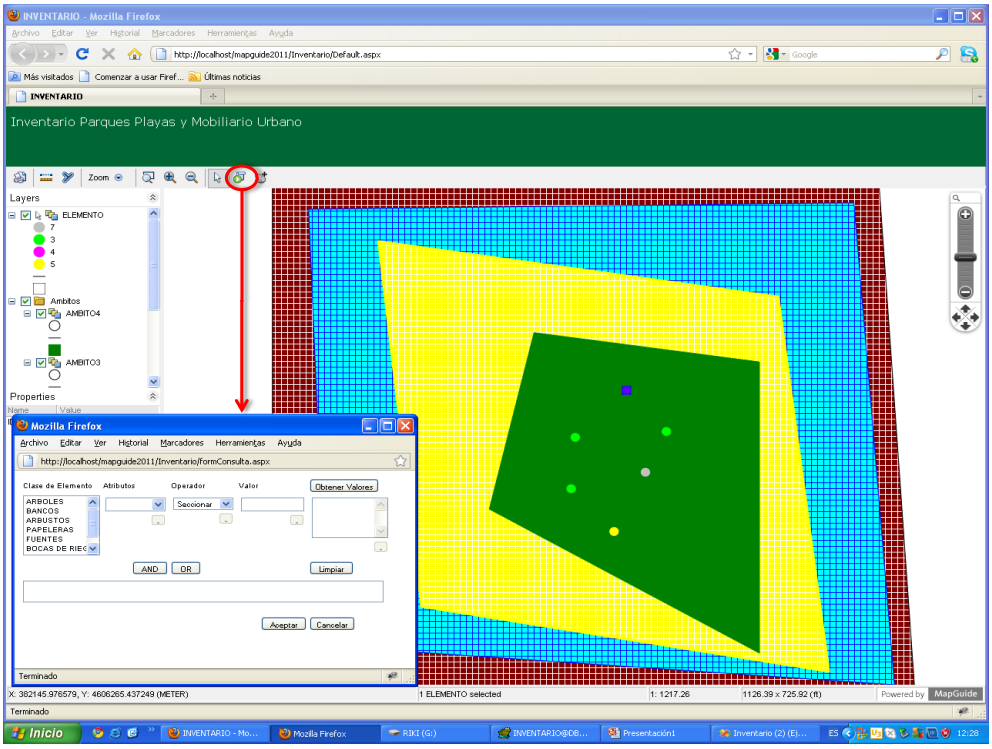


Figura 4.9 Formulario de consulta

La Figura 4.10 muestra en detalle el formulario de consulta. El usuario tendrá que elegir en primer lugar la “Clase de elemento”, al elegir la clase automáticamente el programa cargará sus atributos. Con la lista desplegable “Operador” el usuario podrá elegir uno de los típicos operadores lógicos del lenguaje SQL (>, <, =, <>, >=, <=).

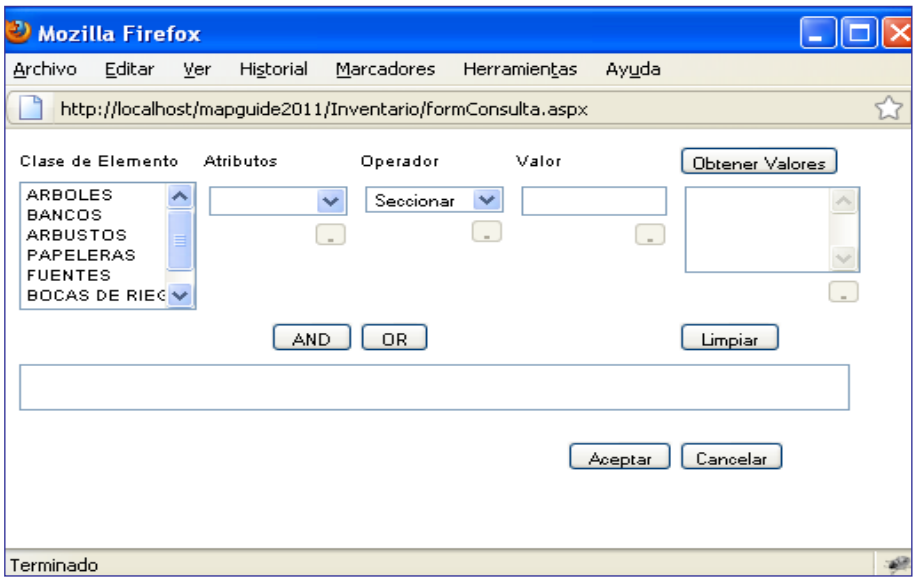


Figura 4.10 Detalle, formulario de consulta

En el campo “Valor” se deberá insertar el valor que el usuario quiere buscar en base de datos, en el caso haya valores posibles ellos serán visibles apretando el botón “Obtener Valores”, este operación permitirá visualizar todos los valores que hay en base de datos y el usuario solo deberá seleccionar lo que quiere buscar y automáticamente el programa escribirá el dato en al campo “Valor”.

En Figura 4.11 se muestra un ejemplo de búsqueda compuesta. Una vez definida la primera parte de la consulta el usuario podrá terminar la consulta apretando el botón “Aceptar” o seguir añadiendo términos a la búsqueda. Seleccionando los botones “AND” o “OR” todo el formulario se limpia y el usuario podrá volver a digitar una nueva sentencia para crear una consulta compuesta. En el ejemplo en Figura 4.11 el usuario pretende buscar arboles con diámetro mayor que 10 y bancos de color marón. El cuadro de texto enseña la sentencia de búsqueda mientras la correspondiente sentencia en lenguaje SQL será guardada exclusivamente en el código, sin enseñarla nunca al usuario.

Los resultados se enseñaran en una tabla (*GridView*) insertada en un *Place Holder,* es decir que dependiendo de la clase de elementos habrá una tabla que se crea de manera dinámica (ver explicación del código, Capitulo 3.3.7). En el ejemplo la búsqueda ha devuelto cuatro elementos (tres arboles y un banco) el programa será capaz de separar los elementos y agruparlos por clases en diferentes tablas.

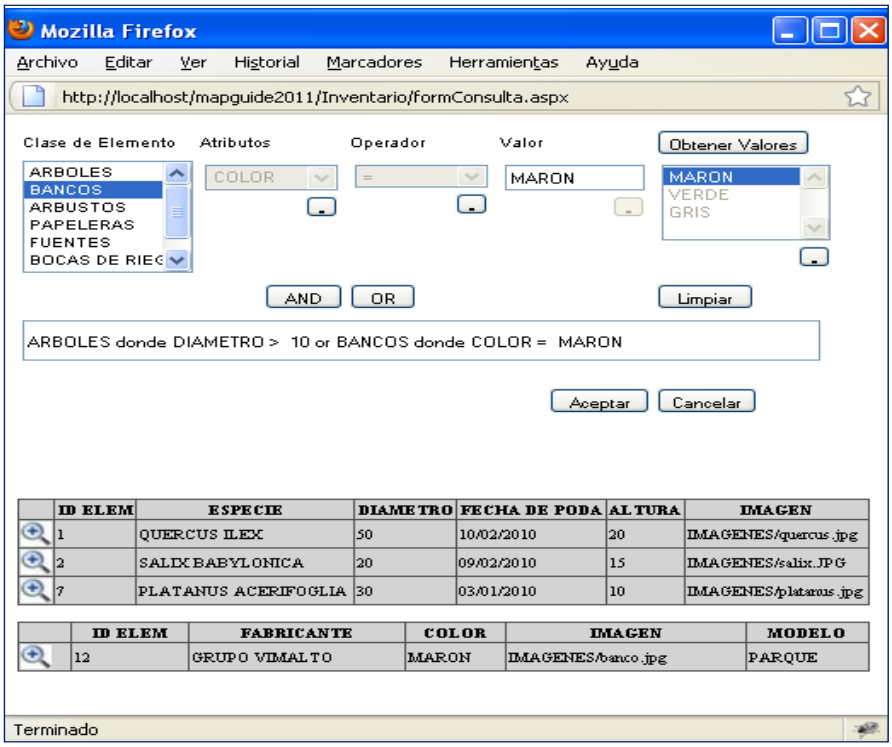


Figura 4.11 Ejemplo, formulario de consulta

Una vez que la tabla resultado esté visible seleccionando el icono al lado izquierdo de la misma el programa enseñará la posición del elemento en el mapa (el elemento resultará evidenciado con un cuadrado de diferente color), ver Figura 4.12.

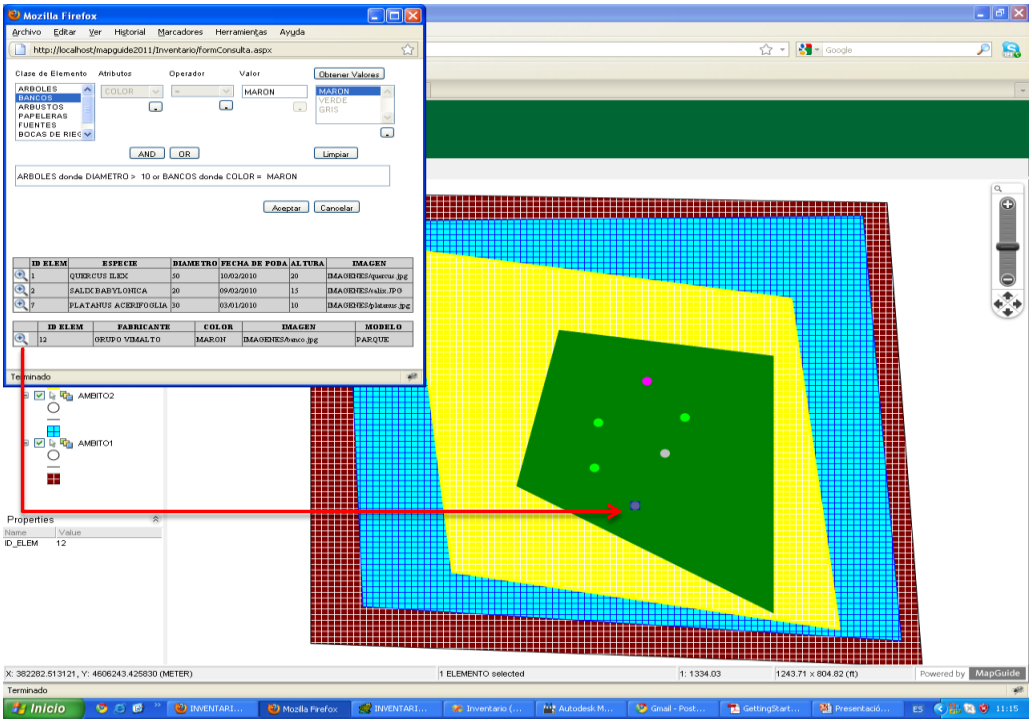


Figura 4.12 Resultado, formulario de consulta

5. CONCLUSIONES

Este trabajo perseguía como objetivo fundamental desarrollar una aplicación web de inventario de fácil uso adaptable a más de un ámbito de estudio que permita la identificación de los elementos puntuales, la consulta e actualización periódica de los datos. Todos los objetivos específicos han sido logrados gracias también a la ventaja de tener unos manuales oficiales disponibles en la empresa que han permitido al autor de familiarizar de manera rápida con el funcionamiento de los software que se han utilizado para el desarrollo del proyecto. Podemos considerar esto junto al control de calidad que muchas organizaciones necesitan, como las principales ventaja de trabajar con software de pago.

MapGuide es una aplicación de sistemas de información geográfica (GIS) centrada en red. Es un GIS construido desde el principio para la arquitectura de Internet y de Intranets corporativas. Comparando su comportamiento con otras tecnologías utilizadas durante el Máster se considera MaGuide un valido instrumento para la publicación de mapas y datos en la red. En Este proyecto ha permitido crear una aplicación con una interfaz intuitiva de fácil uso, crear y cargar archivos de datos incluso en formato DWG, conectarse ágilmente a la base de datos desde la página web. La ventaja más grande que el autor ha encontrado en el utilizo de MapGuide es la posibilidad de construir el *layout web* del visor utilizando la interfaz de usuario del programa sin tener que programar en Java Script. MapGuide proporciona por defecto una barra de herramienta con las principales funcionalidades que nuececita un visor, una barra de zoom y construye automáticamente la leyenda reservándole un *div* en la página web. También la barra de estado es posicionada y rellenada directamente por el programa con escala, coordenadas y informaciones sobre los elementos seleccionados.

Las mayores dificultades que se han encontrado utilizando MapGuide residen en la complicada estructura de los ficheros de Java Script que lo componen, es muy complicado individuar adentro las carpetas del programa donde está el código relativo a una determinada funcionalidad para poder hacer modificas.

En conclusión podemos afirmar que el proyecto ha tenido un éxito positivo, dejando satisfechas las partes interesadas. Se ha logrado cumplir los objetivos propuestos y también ha permitido al autor profundizar sus conocimientos de programación web. La empresa retiene que el proyecto pueda servir para futuros pedidos de sus clientes y sobretodo el hecho que se haya conseguido implementar una base de datos totalmente genérica permite al proyecto una cierta versatilidad dato que podrá ser adaptado a

cualquier ámbito que requiera inventariar sus elementos puntuales (parque, jardines, playas, mobiliario urbano).

Para mejorar la aplicación se recomienda la implementación de nuevas funcionalidades como la posibilidad de añadir nuevos elementos puntuales desde la página web, la edición grafica de los elementos (por ejemplo desplazamientos de los puntos) y la implementación del la aplicación utilizando el *layout Fusion* de MapGuide que mejoraría mucho la presentación de la aplicación desde el punto de vista grafico.

6. BIBLIOGRAFÍA

Referencias web

[1] http://www.jorgesanchez.net/bd/disenoBD.pdf

[2] http://www.um.es/docencia/barzana/DIVULGACION/INFORMATICA/sgbd.html

[3] http://es.wikipedia.org/

[4] http://www.mappinginteractivo.com/plantilla-ante.asp?id_articulo=649 (Entrevista con Joe Astroth, vicepresidente división Gs de Autodesk, 1998)

[5] http://msdn.microsoft.com/en-us/library/ex526337.aspx (Microsoft® on line manual)

Referencias bibliográficas

[6] Autodesk MapGuide® Enterprise 2011 (Junio 2010). Get Starting Guide, Manual Oficial, 26-89.

[7] Fernando Berzal, Francisco J. Cortijo & Juan Carlos Cubero (ISBN 84-609-4245-7). Desarrollo Profesional de Aplicaciones Web con ASP.NET, 44-80. Disponible en internet a la dirección: http://elvex.ugr.es/decsai/csharp/pdf/web/web-book-b5.pdf

[8] Softstar Research, David M. Rubin (Julio 1998). Uses of Use Cases, 4-31.

Apuntes de clase

[9] Apuntes de la asignatura Programación orientada a objetos. Valls, J. (2010). MTIG, 12ª edición. UAB.

[10] Apuntes de la asignatura Programación de aplicaciones SIG en Internet. Ferrero, I. (2010). MTIG, 12ª edición. UAB.

[11] Apuntes de la asignatura Bases de datos espaciales. Nunes, J. (2010). MTIG, 12ª edición. UAB.