Gestión en el territorio de una empresa de seguridad mediante estándares GIS e integración de datos con Google Maps.

Memoria

Autor: David Rodríguez Martínez

Dirigido por: Oscar Fonts Bartolomé

Proyecto de Fin Carrera – Ingeniería Informática

Curso 2006-2007 – 2º Semestre

Agradecimientos

Gracias a Rosi y a mi familia. Vuestro apoyo, comprensión y cariño me ayudaron desde el inicio de esta aventura que finaliza con la construcción de este proyecto.

Gracias también a Oscar Fonts por sus sabios consejos y su comprensión.

Abstact

Management in the territory of a security company by means of GIS standards and data integration with Google Maps

This work is divided into two different parts. In the first one we will show a brief theoretical study about the **Geographical Information Systems** (GIS) and the specifications published by the **Open Geospatial Consortium** (OGC) as open standards that allow the communications among different GIS. Next, we will have an approach to Google´s alternative: its **Google Maps** service for showing geographic information through the Web.

In the second part, we will give a solution to location and distribution of static and mobile resources owned by a certain security company. This solution will use Google Maps as a geocoding system.

Developing the concepts which will allow to build a system to make the representation of the information about maps that Google Maps provides easier will be the general objective of this document. This will be carried out from the general aspects to the more specific ones.

The computing work starts from a definition of objectives and through the design of the application, it will conclude with the development of a system that fulfils the initial purposes. These are the steps followed for the system development: requirements analysis and architecture design, application implementation using Google Maps API for the required inquiries.

Finally, the “conclusions” chapter will show a evaluation of the project, of the different existing mechanisms used to introduce geocode information and the possibilities that Google Maps service offers .It will emphasize the practical work as the main result, that is, the possibility of locating and distributing the different resources that a security company has developing an application that uses the Google Maps presentations.

Resumen

Gestión en el territorio de una empresa de seguridad mediante estándares GIS e integración de datos con Google Maps

El presente trabajo está estructurado en dos partes claramente diferenciadas. En la primera parte mostraremos un pequeño estudio teórico sobre los **Sistemas de Información Geográfica** (GIS) y sobre las especificaciones que está publicando el **Open Geospatial Consortium (**OGC) como estándares abiertos que permitan las comunicaciones entre distintos GIS. A continuación expondremos una aproximación a la alternativa que ofrece Google con su servicio **Google Maps** para la presentación de información geográfica accesible vía web.

En la segunda parte de este trabajo ofreceremos una solución que permita la localización y distribución de los recursos (móviles y estáticos) que dispone una determinada empresa de seguridad. Dicha solución utilizará Google Maps como sistema de representación de elementos georreferenciados.

El objetivo general de este documento será desarrollar, progresivamente desde aspectos genéricos hasta el detalle, los conceptos que permitirán la construcción de un sistema que facilitará la representación de la información sobre los mapas que provee Google Maps.

El trabajo práctico parte de una definición de objetivos, y mediante el diseño de la aplicación, concluirá con el desarrollo de un sistema que cumplirá íntegramente los objetivos iniciales. El desarrollo de la aplicación se compone principalmente de los siguientes pasos: análisis de los requisitos, diseño de la arquitectura necesaria e implementación de la aplicación con las llamadas necesarias a la API de Google Maps para la realización de las consultas requeridas.

Finalmente, el capítulo “conclusiones” mostrará una valoración del proyecto, de los distintos mecanismos que existen para presentar información georreferenciada y de las posibilidades que ofrece el servicio Google Maps en particular. Destacando como principal resultado el sistema desarrollado a través del trabajo práctico, es decir, la posibilidad de localizar y distribuir los distintos recursos con los que cuenta una empresa de seguridad, mediante el desarrollo de una aplicación que hace uso de Google Maps.

Índice

1 Introducción ... 1

1.1 Justificación ... 1
1.2 Objetivos ... 2
1.3 Enfoque y método seguido ... 2
1.4 Planificación ... 3
1.4.1 Elaboración del plan de proyecto (PEC1) ... 6
1.4.2 Estudio teórico (PEC2) ... 6
1.4.3 Diseño y desarrollo de un sistema de información (PEC3) ... 7
1.4.4 Memoria Final y Presentación ... 7
1.5 Productos obtenidos ... 8

2 Sistemas de información geográfica ... 9

2.1 Definición ... 9
2.1.1 Funciones básicas de un GIS ... 9
2.1.2 Componentes de un GIS ... 9
2.1.3 Aplicaciones de un GIS ... 10
2.2 Geodesia y cartografía ... 10
2.2.1 Cartografía ... 10
2.2.1.1 Coordenadas geográficas ... 11
2.2.1.2 Meridianos y paralelos ... 11
2.2.1.3 Longitud y latitud ... 12
2.2.1.4 Azimut ... 13
2.2.1.5 Proyecciones cartográficas ... 13
2.2.1.6 Proyección Universal Transversa de Mercator (UTM) ... 14
2.2.2 Geodesia ... 15
2.2.2.1 Elipsoide de referencia ... 16
2.2.2.2 DATUM ... 17
2.3 Representación de información ... 17
2.3.1 Modelo raster ... 17
2.3.2 Modelo vectorial ... 18
2.3.3 Comparación de modelo vectorial y raster ... 19

3 OGC ... 20

3.1 Introducción ... 20
3.2 Estándares publicados ... 21
3.2.1 Servicio de catalogo (CSW) ... 21
3.2.2 Acceso de las entidades simples (Simple Features - SQL, CORBA, OLE) ... 21
3.2.3 Servicio de Coberturas (WCS) ... 22
3.2.4 Servicios de Transformación de Coordenadas (WCTS) ... 22
3.2.5 Lenguaje de marcado para información geográfica (GML) ... 22
3.2.6 Especificación de codificación de los Filtro (FE) ... 23
3.2.7 Servicios de Mapas por el Web (WMS) ... 24
3.2.8 Servicio de objetos geográficos (WFS) ... 24
3.2.9 Descriptor de estilos de capa (SLD) ... 24
3.3 Web Feature Service ... 24
3.3.1 Tipos de WFS ... 24
3.3.2 Operaciones ... 25
3.3.3 Secuencia de una petición de consulta ... 25
3.3.4 Productos comerciales ... 26
3.4 Web Map Service ... 26
3.4.1 Operaciones ... 27
3.4.2 Tipos de WMS ... 27
3.4.3 Secuencia de una petición de consulta ... 28
3.4.4 Productos comerciales ... 30
3.4.5 Clientes ligeros ... 30

3.4.6 Clientes pesados ... 30
3.5 Ejemplos de implementaciones WFS y WMS ... 30
3.5.1 Dirección general de catastro ... 31
3.5.2 Sistema de información territorial de Galicia (SITGA) ... 32
3.5.3 Otros ejemplos ... 32

4 API de Google Maps ... 33

4.1 Introducción ... 33
4.2 Licencia de uso del servicio y de la API ... 33
4.3 Conceptos previos ... 34
4.4 Ventajas y desventajas de Google Maps ... 35
4.5 Comparación de Google Maps y WMS ... 36
4.6 La API de Google Maps ... 37
4.7 Ejemplos de aplicaciones integradas con Google Maps ... 42
4.7.1 Usos comerciales ... 42
4.7.1.1 Muchoviaje.com ... 43
4.7.1.2 Sistema de información de asociaciones de productores de mejillón ... 43
4.7.1.3 Servicio de trenes del Reino Unido ... 45
4.7.2 Usos no comerciales ... 45

5 Diseño del GIS ... 46

5.1 Requisitos de información geográfica ... 46
5.2 Análisis de requisitos ... 46
5.3 Arquitectura ... 46
5.4 Diseño ... 48

6 Sistema de localización de elementos de seguridad – SIGSE ... 51

6.1 Introducción ... 51
6.2 Captura de requisitos ... 51
6.3 Análisis ... 51
6.3.1 Actores ... 52
6.3.2 Casos de uso ... 52
6.3.2.1 Caso de uso 1: Localizar Videocámaras ... 52
6.3.2.2 Caso de uso 2: Localizar posición actualizada de los elementos móviles ... 53
6.3.2.3 Caso de uso 3: Localizar la ruta asociada a un elemento móvil ... 54
6.3.3 Identificación de problemas ... 56
6.4 Diseño ... 56
6.4.1 Modelo de datos ... 56
6.4.2 Diagramas de secuencia ... 60
6.4.2.1 Localizar Videocámaras ... 60
6.4.2.2 Localizar posición actualizada de los elementos móviles ... 61
6.4.2.3 Localizar la ruta asociada a un elemento móvil ... 62
6.4.3 Arquitectura del sistema ... 63
6.4.3.1 Capa de presentación ... 63
6.4.3.2 Capa de negocio ... 64
6.4.3.3 Capa de persistencia ... 65
6.4.3.4 Diagrama de capas ... 66
6.4.3.5 Secuencia de llamadas entre componentes ... 66
6.5 Implementación ... 67
6.5.1 Aplicación Web ... 67
6.5.1.1 Decisiones de implementación ... 67
6.5.1.2 Interfaces ... 67
6.5.1.3 Funcionamiento ... 68
6.5.1.4 Código fuente ... 68
6.5.2 Cliente ... 68
6.5.2.1 Presentación personalizada de la información en el mapa ... 68
6.5.2.2 Interfaces ... 71
6.5.2.3 Funcionamiento ... 72
6.5.2.4 Código fuente ... 72
6.5.3 Aplicación para la recolección de coordenadas ... 73

6.5.3.1 Funcionamiento ... 74
6.6 Requisitos hardware y software ... 74
6.6.1 Entorno de desarrollo ... 74
6.6.2 Entorno de explotación ... 74
7 Conclusiones ... 76
7.1 Líneas futuras de trabajo ... 76
8 Anexo ... 78
8.1 Código fuente - servidor ... 78
8.2 Código fuente - cliente ... 79
9 Glosario ... 86
10 Referencias ... 88
10.1 Sistemas de información geográfica ... 88
10.2 Cartografía ... 88
10.3 OGC ... 89
10.4 API Google Maps ... 89

Índice de ilustraciones

Figura 1.1 Diagrama Gantt: Plan de proyecto 6
Figura 1.2 Diagrama Gantt: Estudio teórico 6
Figura 1.3 Diagrama Gantt: Diseño y desarrollo de SI 7
Figura 1.4 Diagrama Gantt: Presentación de memoria final 7
Figura 2.1 Geoide y elipsoide 11
Figura 2.2 Meridiano de Greenwich 11
Figura 2.3 Paralelo del ecuador 12
Figura 2.4 Longitud 12
Figura 2.5 Latitud 13
Figura 2.6 Azimut 13
Figura 2.7 Proyección Mercator 14
Figura 2.8 Proyección UTM 14
Figura 2.9 Zonas UTM 15
Figura 2.10 Relación del geoide y elipsoide 16
Figura 2.11 Elipsoide 16
Figura 2.12 Punto fundamental 17
Figura 2.13 Modelo raster 18
Figura 2.14 Modelo vectorial 19
Figura 2.15 Comparación de modelo vectorial y raster 19
Figura 3.1 Puntos de vista RM-ODP 20
Figura 3.2 Simple features 22
Figura 3.3 Jerarquía de clases de GML 23
Figura 3.4 Petición WFS 25
Figura 3.5 Petición WMS 28
Figura 3.6 Ejemplo de llamada a WMS 28
Figura 3.7 Respuesta GetCapabilities SITGA 29
Figura 3.8 Respuesta GetMap SITGA 29
Figura 3.9 Portal de información geográfica de Cataluña 31
Figura 3.10 Integración de datos de catastro con Google Earth 31
Figura 3.11 Esquema del portal de información territorial de Galicia 32
Figura 4.1 Ventana de información estándar 39
Figura 4.2 Ventana de información con solapas 39
Figura 4.3 Ejemplo de polilíneas 40
Figura 4.4 Controles de movimiento: GLargeMapControl, GSmallMapControl, GSmallZoomControl 41
Figura 4.5 Ejemplo de uso comercial de los servicios de Google Maps 43
Figura 4.6 Ejemplos de presentaciones de datos con Google Maps 44
Figura 4.7 Diagrama de secuencia de la integración del SI de asociaciones de mejillón con Google Maps 44
Figura 4.8 Servicio de trenes del Reino Unido 45
Figura 4.9 Ejemplo de uso no comercial de Google Maps 45
Figura 5.1 Arquitectura del sistema 47
Figura 5.2 Distribución del sistema 48
Figura 5.3 Modelo de datos GIS 49
Figura 5.4 Diseño de tablas del GIS 49
Figura 5.5 Sistema recolector de coordenadas 50
Figura 6.1 Caso de uso 1: Localizar videocámaras 53
Figura 6.2 Caso de uso 2: Localizar posición actualizada de los elementos móviles 54
Figura 6.3 Caso de uso 3: Localizar la ruta asociada a un elemento móvil 55
Figura 6.4 Diagrama de entidades del sistema 57
Figura 6.5 Diagrama relacional del sistema 58
Figura 6.6 Diagrama de secuencia: Localizar Videocámaras 60

Figura 6.7 Diagrama de secuencia: Localizar posición actualizada de los elementos móviles 61
Figura 6.8 Diagrama de secuencia: Localizar la ruta asociada a un elemento móvil 62
Figura 6.9 Capas del sistema 66
Figura 6.10 Diagrama de secuencia entre capas 66
Figura 6.11 Localización de rutas SIGSE 69
Figura 6.12 Ventana de información por defecto 70
Figura 6.13 Ventana de información personalizada 70
Figura 6.14 Menú SIGSE 71
Figura 6.15 Pantalla de localización de elementos SIGSE 71
Figura 6.16 Ejemplo navegación en la aplicación SIGSE 72

Índice de tablas

Tabla 1 Planificación de tareas 4
Tabla 2 Categorías de servicio definidas por OpenGIS 21
Tabla 3 Entidades del sistema SIGSE 60
Tabla 4 Definición de XML SIGSE 65
Tabla 5 Atributos de pantalla de información Google Maps 69

1 Introducción

El presente documento recoge los trabajos y conclusiones abordados con objeto del Proyecto de Fin de Carrera (en adelante PFC) titulado **Gestión en el territorio de una empresa de seguridad mediante estándares SIG e integración de datos con Google Maps**.

La presente memoria incluye las siguientes secciones:

- Una **introducción** del proyecto que identifica los principales **objetivos** del mismo. Por otro lado se identifican los **resultados obtenidos** con este PFC y la **metodología** utilizada para abordar los trabajos y cumplir el **calendario** de ejecución de los mismos.
- Una aproximación a los conceptos básicos necesarios para entender los **Sistemas de Información Geográfica**. Esta sección está compuesta por:
    - Una presentación de las **definiciones** de los conceptos fundamentales para trabajar con sistemas geográficos
    - Un breve estudio de los conceptos relacionados con **Cartografía y Geodesia**.
    - Una introducción a los distintos **sistemas de representación** utilizados por los sistemas de información geográficos.
- Una aproximación a los trabajos realizados por **Open Geospatial Consortium (OGC)** compuesta de:
    - Una **introducción** que permita entender la organización y objetivos que persigue el OGC.
    - Un conciso estudio de los estándares y **especificaciones** que ha publicado hasta el momento el OGC.
    - Un estudio en detalle de los dos principales estándares que ha publicado hasta el momento el OGC: **Web Feature Service (WFS) y Web Map Service (WMS)**. Este estudio presentará algunas implementaciones realizadas de estos servicios en la actualidad.
- Una aproximación a las funcionalidades que publica el servicio desarrollado por Google, denominado **Google Maps** compuesto de:
    - Una i**ntroducción** del servicio ofertado por Google y de los términos de la **licencia de uso** de este servicio.
    - Una descripción de la **API de Google Maps** publicada en la actualidad, centrándonos en las posibilidades que oferta dicho servicio para las aplicaciones externas.
    - Una presentación de posibilidades y **ejemplos** reales de explotación del servicio.
- Un **diseño de un sistema GIS** que permita la gestión y localización de los recursos de una compañía de seguridad.
- Un diseño de un **Sistema de localización de elementos de seguridad** que emplee Google Maps como proveedor de mapas.
- Una presentación de las **conclusiones** y resultados obtenidos con el presente PFC.
- La **bibliografía** de referencia para la elaboración del presente PFC.

1.1 Justificación

Este proyecto surge como respuesta a la necesidad de una empresa de seguridad, de localizar y distribuir los distintos elementos y recursos que dispone para ofrecer sus servicios en función de la localización de sus clientes.

Dentro del panorama formado por los GIS tradicionales están apareciendo una serie de movimientos que están transformando el paradigma clásico de los GIS. Por un lado destacamos la estandarización de

formatos de intercambio de datos geoespaciales. A mayores contamos con la aparición del fenómeno Google Maps que facilita enormemente la presentación de información georreferenciada sobre mapas o fotos por satélite.

Para ello se considera que los GIS y en concreto las especificaciones liberadas por el OGC puede ser herramientas interesantes para gestionar la información de dicha empresa. Al mismo tiempo se analizan soluciones de presentación de información georreferenciada a través de servicios de mapas "simples" como Google Maps.

1.2 Objetivos

Este proyecto tiene como objetivo la investigación de los estándares abiertos *Open Geospatial Consortium* (OGC) que permiten la consulta de datos geoespaciales y su posible representación en sistemas GIS. Por otro lado nos interesa la investigación sobre las nuevas posibilidades que ofrecen aplicaciones del tipo de *Google Maps* y *Google Earth*. Dichas aplicaciones nos permiten representar geográficamente los datos de sistemas de información tradicionales sin necesidad de recurrir a los GIS tradicionales, eliminando los costes económicos que ello implican.

El objetivo final del presente proyecto será el diseño y desarrollo de un sistema web que permita la localización de los recursos (videocámaras y elementos móviles) que dispone una determinada empresa de seguridad, facilitando su distribución y gestión de los mismos.

A continuación presentamos los objetivos generales que abarca el proyecto:

- Conocer qué es un GIS.
- Saber utilizar los estándares más extendidos del OGC: construir peticiones e interpretar las posibles respuestas.
- Descubrir los servicios y aplicaciones web existentes basadas en estándares abiertos (OGC) y de facto (Google). Valorar y comparar su implantación actual y potencial futuro.
- Desarrollar aplicaciones web complejas con Javascript, mediante la integración de servicios existentes.

Durante la parte práctica del proyecto se construirá un sistema de información que permitirá a la empresa de seguridad localizar sobre un mapa todos los recursos de seguridad de los que dispone. Entre otras cosas podrá:

- Localizar las videocámaras instaladas, permitiendo consultar su posición y estado.
- Localizar y posicionar en tiempo real de los elementos móviles de seguridad con los que cuenta la compañía.

La aplicación mostrará toda la información posicionada sobre los mapas que nos ofrece *Google* en su servicio *Google Maps*.

1.3 Enfoque y método seguido

El enfoque general que se ha aplicado al presente PFC parte de un estudio riguroso que permita la compresión de los distintos elementos que intervienen en los sistemas de información geográfica y los conceptos básicos sobre la información geográfica, para adentrarnos con posterioridad en el objetivo práctico de este PFC: Diseñar y desarrollar un sistema web que presente información georreferenciada utilizando el servicio de Google Maps. Por garantizar el seguimiento de este enfoque han sido básicos los siguientes puntos:

- **Búsqueda y selección de bibliografía**: Uno de los puntos más importantes y críticos es la búsqueda y selección de buenas fuentes de información que posibiliten el desarrollo del presente proyecto, tanto para su parte práctica como para su parte teórica.
- **Búsqueda y selección de casos de uso** reales de implementaciones de los servicios adaptados a las especificaciones establecidas por el OGC y aplicaciones que utilicen los servicios ofrecidos por Google Maps.

La metodología que se ha empleado en los puntos más teóricos ha sido la siguiente:

- Buscar y contrastar información de fuentes fiables del tema a tratar.
- Seleccionar bibliografía a utilizar, obtener toda la información posible de la misma y tenerla accesible.
- Desarrollar un borrador en base a la documentación anterior.
- Confeccionar la parte del documento correspondiente, incorporando las figuras necesarias.
- Repasar la documentación redactada.

Para la elaboración de la parte práctica del proyecto se ha utilizado un ciclo de vida en cascada, en el cual las diferentes etapas de la construcción del software se ven como una secuencia casi lineal, en la que una etapa debe finalizar antes de que se inicie la siguiente. A continuación explicamos las etapas en las que hemos estructurado la construcción del sistema:

- **Análisis de requisitos**: Análisis y elaboración posterior de las especificaciones, las funciones y los objetivos del sistema informático que se quiere implementar.
- **Diseño del sistema**: Diseño de una solución técnica concreta que satisfaga las especificaciones establecidas en la fase de análisis
- **Implementación**: Programación del diseño efectuado para el sistema utilizando los lenguajes de programación seleccionados.
- **Pruebas**: Definición y ejecución de un conjunto de test sobre el sistema construido, con el objetivo de asegurar el perfecto funcionamiento del sistema y el cumplimiento de los requisitos identificados durante el análisis del sistema.

1.4 Planificación

A continuación presentamos la planificación del proyecto. En esta planificación se presentan las distintas entregas y el contenido de cada una de ellas.

Tarea	Nombre tarea	Actividad	Horas	Inicio	Fin
1	Definición del proyecto		5	28/02/2007	05/03/2007
		Obtener la documentación inicial			
		Leer del enunciado del PFC			
2	Elaboración del plan de proyecto (PEC1)			06/03/2007	12/03/2007
		Obtener la documentación	1		
		Leer documentación	5		
		Buscar bibliografía	2		
		Redactar índice del borrador	1		
		Redactar borrador	9		
		Planificar tareas	5		
		Acabar borrador		09/03/2007	
		Corregir borrador	2	11/03/2007	12/03/2007
		Entregar Plan de Proyecto		12/03/2007	09/04/2007
3	Sistemas de información Geográfica			13/03/2007	19/03/2007
		Recoger documentación	4	13/03/2007	13/03/2007
		Estudiar los GIS	10	14/03/2007	17/03/2007
		Redactar la documentación	4	18/03/2007	19/03/2007
4	OGC			20/03/2007	30/03/2007
		Recoger documentación	6	20/03/2007	22/03/2007

Tarea	Nombre tarea	Actividad	Horas	Inicio	Fin
		Estudiar los estándares abiertos OGC	18	23/03/2007	27/03/2007
		Redactar la documentación	6	28/03/2007	30/03/2007
5	API de Google Maps			31/03/2007	09/04/2007
		Recoger documentación	4	31/03/2007	31/03/2007
		Estudiar la API de Google Maps	20	01/04/2007	07/04/2007
		Redactar la documentación	6	08/04/2007	09/04/2007
	Entrega borrador PEC2			10/04/2007	
	Correcciones PEC2		20	12/04/2007	16/04/2007
	Entrega oficial PEC2			17/04/2007	
6	Diseño del GIS			17/04/2007	26/04/2007
		Investigar	4	17/04/2007	18/04/2007
		Diseñar la representación de los elementos de seguridad	20	19/04/2007	23/04/2007
		Redactar la documentación	6	24/04/2007	26/04/2007
7	Sistema de localización de elementos de seguridad			27/04/2007	16/05/2007
		Capturar los requisitos necesarios	2	27/04/2007	27/04/2007
		Analizar los requisitos recopilados	5	27/04/2007	28/04/2007
		Realizar el diseño del sistema	5	29/04/2007	01/05/2007
		Implementar del sistema	30	02/05/2007	10/05/2007
		Realizar las pruebas unitarias e integración interna	10	11/05/2007	12/05/2007
		Organizar la documentación	10	13/05/2007	16/05/2007
	Entrega borrador PEC3			16/05/2007	
	Correcciones PEC3		20	18/05/2007	22/05/2007
	Entrega oficial PEC3			23/05/2007	
8	Elaboración de la memoria			23/05/2007	30/05/2007
		Redactar líneas futuras de trabajo	4	23/05/2007	24/05/2007
		Revisar la documentación	5	25/05/2007	25/05/2007
		Redactar conclusiones	10	26/05/2007	27/05/2007
		Redactar capítulos formales	2	27/05/2007	27/05/2007
		Hacer repaso semántico	2	28/05/2007	28/05/2007
		Hacer repaso sintáctico	2	29/05/2007	29/05/2007
		Hacer repaso ortográfico	1	30/05/2007	30/05/2007
9	Elaboración de la presentación			30/05/2007	04/06/2007
		Escoger contenidos	7	30/05/2007	01/06/2007
		Redactar presentación	12	02/06/2007	04/06/2007
	Entrega borrador Memoria			04/06/2007	
	Correcciones memoria		20	06/06/2007	10/06/2007
	Entrega oficial Memoria			11/06/2007	
10	Debate virtual				
		Contestar preguntas	10	18/06/2007	22/06/2007
	Total		**315**		

Tabla 1 Planificación de tareas

A continuación presentamos los diagramas de Gantt correspondientes a las cuatro fases en las que se ha dividido el proyecto:

- **Fase inicial del proyecto**: Iniciación y elaboración del plan de proyecto.
- **Estudio teórico** sobre los siguientes conceptos: GIS, especificaciones OGC y servicio de Google Maps.
- **Diseño y desarrollo** de un sistema de información que permita la localización de recursos de seguridad en mapas solicitados al servicio Google Maps.
- Entrega de la **memoria final y presentación** del proyecto.

1.4.1 Elaboración del plan de proyecto (PEC1)

A continuación presentamos el diagrama de Gantt correspondiente a la fase inicial del proyecto, en la que se elaboró el Plan de proyecto (PEC1):

Id	Nombre de tarea	Comienzo	Fin	% completado
1	**PEC 1**	**sáb 03/03/07**	**lun 12/03/07**	**100%**
2	**Definición del proyecto**	**sáb 03/03/07**	**dom 04/03/07**	**100%**
3	Obtener la documentación inicial	sáb 03/03/07	sáb 03/03/07	100%
4	Lectura del enunciado del PFC	dom 04/03/07	dom 04/03/07	100%
5	**Elaboración del plan de proyecto (PEC1)**	**mié 07/03/07**	**lun 12/03/07**	**100%**
6	Obtener la documentación	mié 07/03/07	mié 07/03/07	100%
7	Leer documentación	jue 08/03/07	jue 08/03/07	100%
8	Buscar bibliografía	jue 08/03/07	jue 08/03/07	100%
9	Redactar índice del borrador	jue 08/03/07	jue 08/03/07	100%
10	Redactar borrador	vie 09/03/07	vie 09/03/07	100%
11	Planificar tareas	vie 09/03/07	vie 09/03/07	100%
12	Acabar borrador	vie 09/03/07	vie 09/03/07	100%
13	Corregir borrador	sáb 10/03/07	sáb 10/03/07	100%
14	Entregar Plan de Proyecto	lun 12/03/07	lun 12/03/07	100%

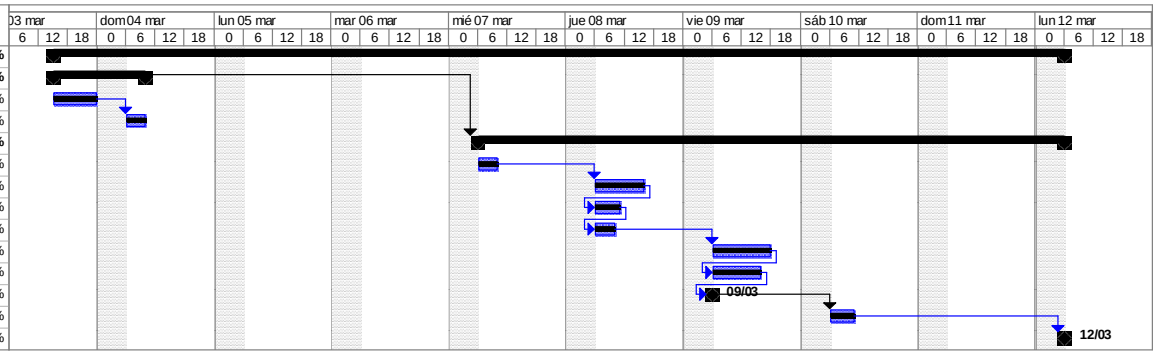

Figura 1.1 Diagrama Gantt: Plan de proyecto

1.4.2 Estudio teórico (PEC2)

A continuación presentamos el diagrama de Gantt correspondiente a la fase del estudio teórico sobre los siguientes conceptos: GIS, especificaciones OGC y servicio de Google Maps (PEC2):

Id	Nombre de tarea	Comienzo	Fin	% completado
1	**PEC 1**	**sáb 03/03/07**	**lun 12/03/07**	**100%**
15	**PEC2**	**mar 13/03/07**	**mar 17/04/07**	**100%**
16	**Sistemas de información Geográfica**	**mar 13/03/07**	**lun 19/03/07**	**100%**
17	Recoger documentación	mar 13/03/07	mar 13/03/07	100%
18	Estudiar los GIS	mié 14/03/07	vie 16/03/07	100%
19	Redactar la documentación	dom 18/03/07	lun 19/03/07	100%
20	**OGC**	**mar 20/03/07**	**vie 30/03/07**	**100%**
21	Recoger documentación	mar 20/03/07	mié 21/03/07	100%
22	Estudiar los estándares abiertos OGC	vie 23/03/07	lun 26/03/07	100%
23	Redactar la documentación	mié 28/03/07	vie 30/03/07	100%
24	**API de Google Maps**	**sáb 31/03/07**	**lun 09/04/07**	**100%**
25	Recoger documentación	sáb 31/03/07	sáb 31/03/07	100%
26	Estudiar la API de Google Maps	dom 01/04/07	vie 06/04/07	100%
27	Redactar la documentación	dom 08/04/07	lun 09/04/07	100%
28	Entrega borrador PEC2	mar 10/04/07	mar 10/04/07	100%
29	Correcciones PEC2	jue 12/04/07	lun 16/04/07	100%
30	Entrega oficial PEC2	mar 17/04/07	mar 17/04/07	100%
31	**PEC3**	**mar 17/04/07**	**mié 23/05/07**	**83%**
46	**Memoria**	**mié 23/05/07**	**lun 11/06/07**	**0%**
61	**Debate virtual**	**lun 18/06/07**	**vie 22/06/07**	**0%**
62	Contestar preguntas	lun 18/06/07	vie 22/06/07	0%

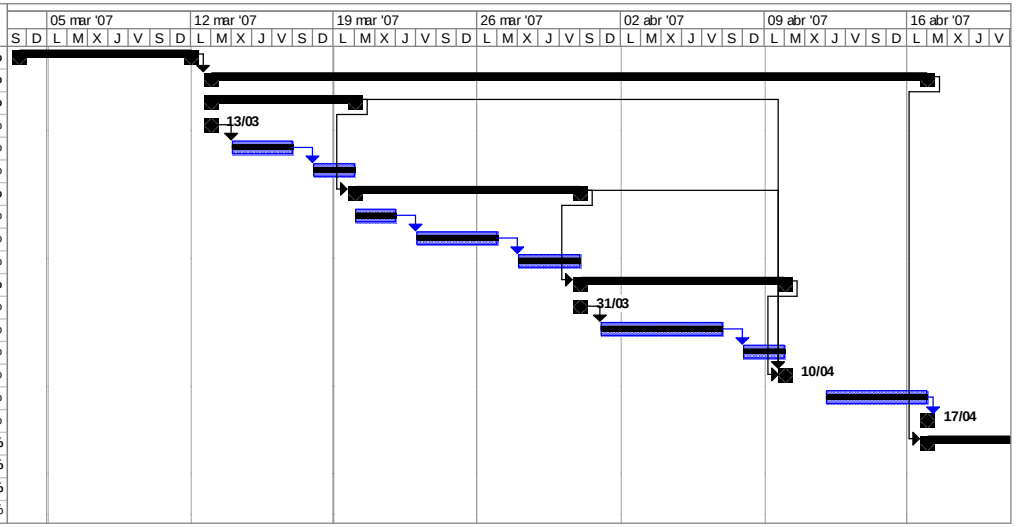

Figura 1.2 Diagrama Gantt: Estudio teórico

1.4.3 Diseño y desarrollo de un sistema de información (PEC3)

A continuación presentamos el diagrama de Gantt correspondiente a la fase de diseño y desarrollo de un sistema de información que permita la localización de recursos de seguridad en mapas solicitados al servicio Google Maps (PEC3):

Id	Nombre de tarea	Comienzo	Fin	% completado
1	**PEC 1**	**sáb 03/03/07**	**lun 12/03/07**	**100%**
15	**PEC2**	**mar 13/03/07**	**mar 17/04/07**	**100%**
31	**PEC3**	**mar 17/04/07**	**mié 23/05/07**	**83%**
32	**Diseño del GIS**	**mar 17/04/07**	**jue 26/04/07**	**100%**
33	Investigar	mar 17/04/07	mar 17/04/07	100%
34	Diseñar la representación de los elemento	jue 19/04/07	dom 22/04/07	100%
35	Redactar la documentación	mar 24/04/07	jue 26/04/07	100%
36	**Sistema de localización de elementos de**	**vie 27/04/07**	**mié 16/05/07**	**100%**
37	Captura de requisitos	vie 27/04/07	vie 27/04/07	100%
38	Análisis de requisitos	vie 27/04/07	vie 27/04/07	100%
39	Diseño	dom 29/04/07	lun 30/04/07	100%
40	Implementación del sistema	mié 02/05/07	mié 09/05/07	100%
41	Pruebas unitarias e integración interna	vie 11/05/07	vie 11/05/07	100%
42	Organización de documentación	dom 13/05/07	mié 16/05/07	100%
43	Entrega borrador PEC3	mié 16/05/07	mié 16/05/07	0%
44	Correcciones PEC3	vie 18/05/07	mar 22/05/07	0%
45	Entrega oficial PEC3	mié 23/05/07	mié 23/05/07	0%
46	**Memoria**	**mié 23/05/07**	**lun 11/06/07**	**0%**
61	**Debate virtual**	**lun 18/06/07**	**vie 22/06/07**	**0%**
62	Contestar preguntas	lun 18/06/07	vie 22/06/07	0%

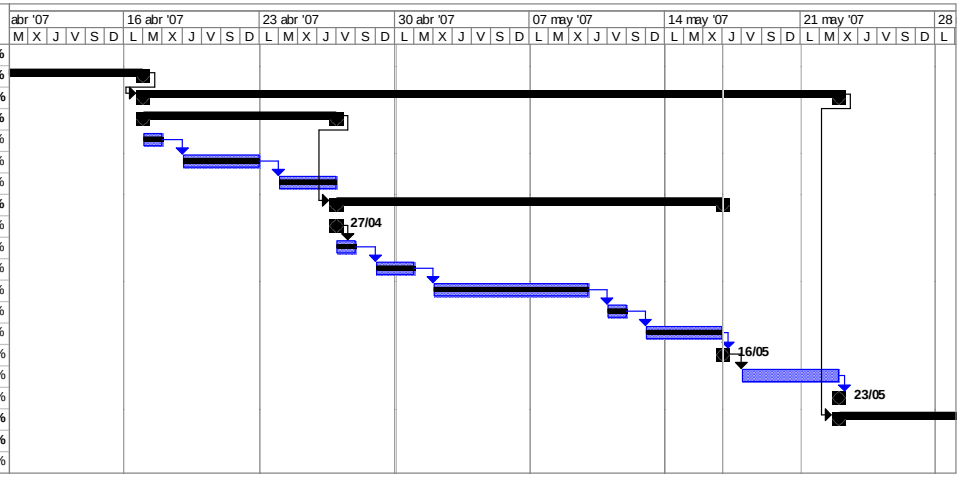

Figura 1.3 Diagrama Gantt: Diseño y desarrollo de SI

1.4.4 Memoria Final y Presentación

A continuación presentamos el diagrama de Gantt correspondiente a la entrega de la memoria final y a la presentación del proyecto:

Id	Nombre de tarea	Comienzo	Fin	% completado
1	**PEC 1**	**sáb 03/03/07**	**lun 12/03/07**	**100%**
15	**PEC2**	**mar 13/03/07**	**mar 17/04/07**	**100%**
31	**PEC3**	**mar 17/04/07**	**mié 23/05/07**	**83%**
46	**Memoria**	**mié 23/05/07**	**lun 11/06/07**	**0%**
47	**Elaboración de la memoria**	**mié 23/05/07**	**mié 30/05/07**	**0%**
48	Redactar líneas futuras de trabajo	mié 23/05/07	mié 23/05/07	0%
49	Revisión de la documentación	vie 25/05/07	vie 25/05/07	0%
50	Redactar conclusiones	sáb 26/05/07	sáb 26/05/07	0%
51	Redactar capítulos formales	dom 27/05/07	dom 27/05/07	0%
52	Hacer repaso semántico	lun 28/05/07	lun 28/05/07	0%
53	Hacer repaso sintáctico	mar 29/05/07	mar 29/05/07	0%
54	Hacer repaso ortográfico	mié 30/05/07	mié 30/05/07	0%
55	**Elaboración de la presentación**	**mié 30/05/07**	**lun 04/06/07**	**0%**
56	Escoger contenidos	mié 30/05/07	jue 31/05/07	0%
57	Redactar presentación	sáb 02/06/07	lun 04/06/07	0%
58	Entrega borrador Memoria	lun 04/06/07	lun 04/06/07	0%
59	Correcciones memoria	mié 06/06/07	sáb 09/06/07	0%
60	Entrega oficial Memoria	lun 11/06/07	lun 11/06/07	0%
61	**Debate virtual**	**lun 18/06/07**	**vie 22/06/07**	**0%**
62	Contestar preguntas	lun 18/06/07	vie 22/06/07	0%

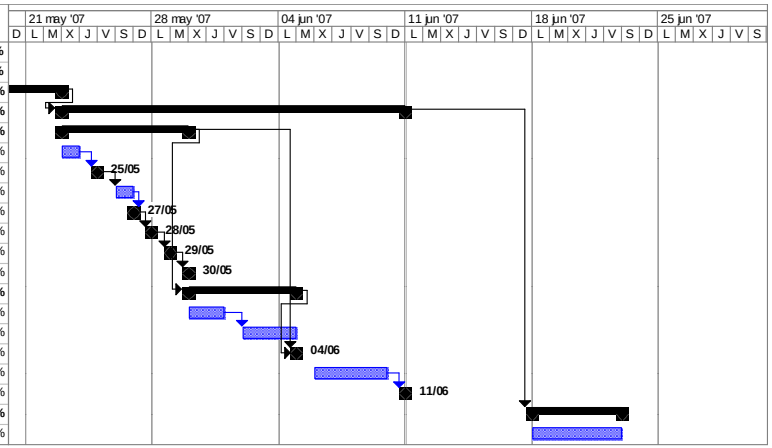

Figura 1.4 Diagrama Gantt: Presentación de memoria final

1.5 Productos obtenidos

Una vez completada la fase inicial de este proyecto se han obtenido los siguientes resultados:

- Una presentación de los principales estándares abiertos establecidos por el OGC.
- Un estudio detallado de los siguientes servicios web más populares que ha definido hasta el momento la OGC:
  - Web Feature Service (WFS): Servicio web que proporciona información geográfica en formato vectorial.
  - Web Map Service (WMS). Servicio web que produce mapas en formato imagen como respuesta a las consultas formuladas los clientes y que pueden ser visualizadas en clientes ligeros.
- Análisis de caso de uso reales de los servicios WMS y WFS.
- Un estudio sobre las posibilidades de programación que ofrece la API de Google Maps y sus capacidades de presentación de información georreferenciada sobre los mapas y fotografías (aéreas y por satélite).
- Análisis de casos de uso reales de sistemas de información que utilicen los servicios Google Maps.

En la segunda parte del presente proyecto se han obtenido, a mayores, los siguientes resultados:

- El diseño y desarrollo de un sistema de información que ofrezca a una empresa de seguridad las siguientes funcionalidades:
  - Una herramienta que le permita hacer un seguimiento de las videocámaras que tiene instaladas a través de su posicionamiento sobre un mapa o foto por satélite.
  - Una utilidad que facilite el seguimiento de los elementos móviles que tiene distribuidos por todo su territorio de actuación a través de su posicionamiento sobre un mapa o foto por satélite.
  - Un sistema de análisis que posibilite aumentar la capacidad de respuesta de la empresa a la hora de dirigir los movimientos de sus elementos móviles en situaciones de emergencia, ya que al conocer la posición de todos sus elementos puede estudiar la mejor estrategia de respuesta posible.

2 Sistemas de información geográfica

Antes de comenzar nuestro estudio sobre los GIS es necesario que establezcamos una definición del término GIS y los conceptos básicos que debemos conocer para comprender estos sistemas.

En los siguientes apartados analizaremos las definiciones de GIS e introduciremos los conceptos más importantes de cartografía y geodesia. Finalizaremos el capítulo con una breve descripción de los principales sistemas que existen para la representación de la información geográfica.

2.1 Definición

Un Sistema de Información Geográfica (GIS en inglés) [1] es un caso particular de SI (sistema de información) en el que la información aparece georreferenciada, es decir, incluye su posición en el espacio utilizando un sistema de coordenadas estandarizado resultado de una proyección cartográfica.

Ante la dificultad de encontrar una definición única que permita comprender completamente el concepto de los sistemas GIS, vamos a establecer una serie de afirmaciones que nos acercarán a dicho concepto:

- Los GIS gestionan información de carácter espacial en bases de datos informáticas
- Los GIS son capaces de representar información georreferenciada sobre cartografía.
- Los GIS están compuestos de distintas herramientas que permiten reunir, introducir, almacenar, recuperar, transformar y cartografiar datos espaciales para un conjunto particular de objetivos.
- Los GIS permiten realizar análisis espaciales de la información gestionada por el sistema.
- Los GIS son sistemas hardware y software que facilitan obtención y representación de datos georreferenciados.

Como se puede observar existen dos elementos claves en las posibles definiciones de GIS:

- Siempre hablamos de sistemas informáticos
- Siempre hablamos de información con un objetivo concreto y georreferenciada.

De acuerdo a estos elementos clave podemos considerar que un GIS es una tecnología aplicada a la resolución de problemas representándolos geográficamente. De esta forma todos los datos contenidos en los sistemas de este tipo tienen asociada una posición geográfica con coordenadas (latitud, longitud), lo que permitirá la realización de análisis espaciales con la ayuda de una base de datos y de software asociado.

2.1.1 Funciones básicas de un GIS

Podemos considerar que las funciones básicas de un GIS son las siguientes [1]:

- Permitir la introducción de información geográfica en formato analógico, convirtiéndola a un formato digital que pueda ser gestionado por el GIS, sin ocasionar pérdida de información (digitalización de información).
- Permitir la representación de los datos introducidos por los usuarios, en algún formato que permita que dicha información sea referenciada geográficamente (mapas, gráficos, etc.).
- Permitir el análisis de los datos introducidos en el SI y su localización geográfica, gracias a herramientas específicas de análisis.

2.1.2 Componentes de un GIS

Identificamos los siguientes componentes básicos que conforman los sistemas GIS [2][3]:

- **Hardware**: Conjunto de ordenadores y periféricos de entrada y salida que constituyen el soporte físico del GIS (incluye tanto el programa de gestión de GIS como otros programas de apoyo). Debido a los requerimientos de capacidad de cálculo y almacenamiento de un GIS, generalmente es preferible destinar equipos específicos en exclusiva a la implementación del GIS (uno o un conjunto de ordenadores u otros dispositivos específicos).

- **Software**: Los programas GIS proveen las herramientas y funcionalidades necesarias para almacenar, analizar y mostrar información geográfica. Sus componentes principales son:
    - Sistema gestor de base de datos espacial: será el encargado de almacenar el modelo de datos que permitirá georreferenciar toda la información del sistema.
    - Sistema gestor de base de datos temática: será el encargado de almacenar la información específica del SI (datos no georreferenciados). Su vinculación con la base de datos cartográfica permite asignar a cada punto, línea o área del territorio unos valores temáticos.

  En la mayoría de las implementaciones comerciales nos encontramos que para el sistema gestor de base de datos se utiliza un SGBD tradicional, al que se le añade una capa de software que incorpora al conjunto la funcionalidad geográfica. De esta forma en un mismo SGBD se almacenan los datos "espaciales" y "temáticos" (explicados anteriormente) y la capa de software que incorpora la funcionalidad geográfica es la responsable de unir dicha información para su explotación.
- **Datos geográficos**: La información geográfica que utilizaremos para la presentación de la información georreferenciada puede obtenerse de proveedores u organismos oficiales o pueden obtenerse por recursos propios (en la actualizad, la digitalización de cartografía en formato analógico sólo se efectúa en casos excepcionales en los que no es posible obtener cartografía digitalizada).
- **Usuarios**: Debido a la complejidad y a la utilidad de este tipo de sistemas resulta importante distinguir entre tres formas de interacción con el GIS:
    - Usuarios, su misión es obtener información del GIS y tomar decisiones en función de la misma
    - Técnicos en GIS, encargados de seleccionar las herramientas, los datos, la escala adecuada de representación para los fines propuestos para el Sistema, y los procedimientos para su introducción en el GIS
    - Informáticos, en sistemas de cierta importancia, son los encargados de su administración.

2.1.3 Aplicaciones de un GIS

Existen multitud de aplicaciones de los sistemas GIS. En general podríamos decir que los GIS son aplicables a cualquier sistema de información que gestione datos que son susceptibles de ser georreferenciados [3]. A modo de ejemplo indicamos algunos campos de aplicación:

- **Gestión de recursos naturales**: Gestión de datos orientados a la consecución de una mayor productividad, rutas de migración, ríos, actuación frente a emergencias (incendios, maremotos, terremotos…), etc.
- **Gestión de infraestructuras**: Localización y gestión de averías, mantenimiento de redes de suministros, etc.
- **Construcción**: Localización exacta de líneas de metro, tuberías, etc.
- **Localización geográfica**: Gestión de personas (móviles), logística (flotas de camiones), transporte, etc.
- **Sistemas de seguridad** basados en localización geográfica: Localización de personas (excursionistas, etc.).
- **Gestión del tráfico**: Radares, carreteras, peajes, rutas, etc.
- **Telecomunicaciones**: Redes de comunicaciones (GSM, WIFI…), elementos de la misma, cobertura, etc.

2.2 Geodesia y cartografía

En el presente apartado presentaremos los conceptos geográficos básicos que nos permitirán trabajar con los GIS. De esta forma intentaremos realizar una pequeña presentación de la cartografía y de la geodesia [17].

2.2.1 Cartografía

Para comenzar podemos decir la cartografía tiene como campo de estudio la representación de datos espaciales. De una forma más concreta, podemos definir la cartografía como la ciencia que trata la

representación de la Tierra sobre un mapa. Este estudio abarca una serie de técnicas necesarias para representar en dos dimensiones (plano) una entidad que está en el espacio (3 dimensiones) [17].

En la geometría no existe una figura que se adapte a la forma exacta de la tierra. Por su aproximación a la realidad se ha elegido el elipsoide para su representación. En el apartado *2.2.2 Geodesia* se explicará con detalle el concepto de geoide, lo que nos permitirá entender la necesidad de establecer un modelo matemático que nos permita representar con la mayor exactitud posible la forma de la tierra.

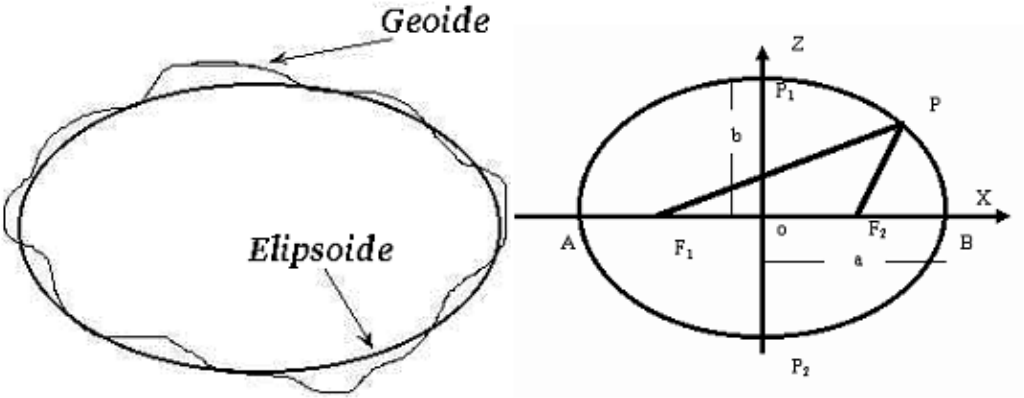

Figura 2.1 Geoide y elipsoide

Se indican a continuación algunas definiciones de conceptos básicos relacionados con las coordenadas geográficas, que serán necesarios para comprender los conceptos sobre proyecciones que presentarán en aparatados posteriores.

2.2.1.1 Coordenadas geográficas

El sistema de coordenadas geográficas nos permitirá la localización de un punto sobre la esfera terrestre. Esta localización se realiza estableciendo una relación a los meridianos y a los paralelos del planeta. De esta forma tenemos que un punto estará definido por una longitud y una latitud [17].

2.2.1.2 Meridianos y paralelos

Los meridianos son las líneas de intersección con la Tierra de los infinitos planos que contienen el eje de la Tierra.

El meridiano de referencia es el meridiano 0º, también llamado meridiano de Greenwich porque pasa por este suburbio de Londres. Este meridiano divide la tierra en dos zonas: este y oeste. A partir de este meridiano se definen el resto de los meridianos [17].

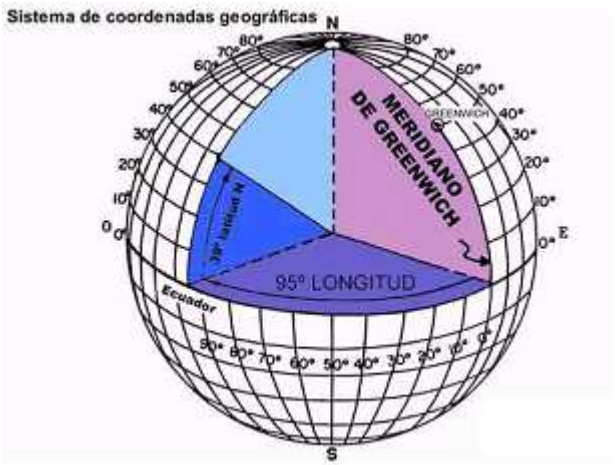

Figura 2.2 Meridiano de Greenwich

Los paralelos son las líneas de intersección de los infinitos planos perpendiculares al eje terrestre con la superficie de la Tierra.

Al paralelo que se encuentra a mayor distancia del eje de la Tierra se le denomina Ecuador, y divide la Tierra en hemisferio norte y sur.

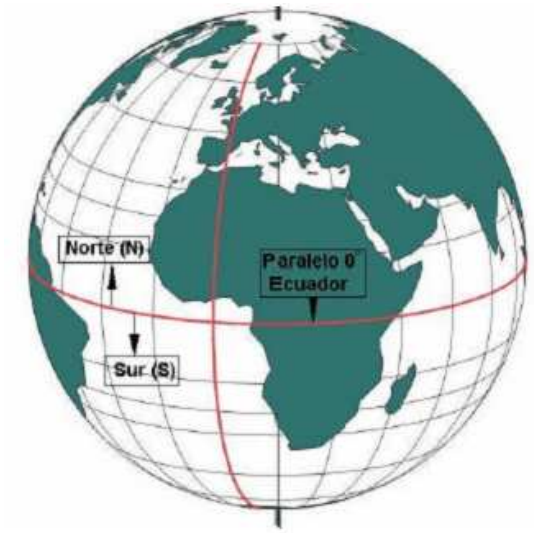

Figura 2.3 Paralelo del ecuador

Todas las coordenadas geográficas establecen una relación con el meridiano 0 (meridiano de Greenwich) y el paralelo 0 (ecuador).

2.2.1.3 Longitud y latitud

La longitud de un punto es el ángulo que forman el plano meridiano que pasa por el mismo y el meridiano origen (meridiano 0). En la figura 2.4 podemos apreciar el ángulo que define la longitud: ángulo OAB.

La longitud abarca un rango de valores que está comprendido entre 0º y 180º hacia el este, y el mismo rango hacia el oeste. Para la localización unívoca de un punto es necesario que se indique si es una longitud oeste o este. Por ejemplo, 9º 17’ 00’’ O (oeste) o 9º 17’ 00’’ E (este) [17].

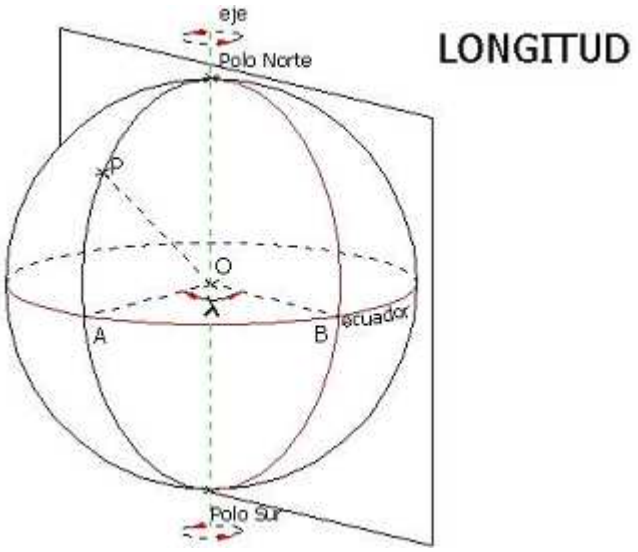

Figura 2.4 Longitud

La latitud de un punto es el ángulo que forma el plano que pasa por el punto y el centro de la Tierra con el plano del Ecuador. En la figura 2.5 podemos apreciar el ángulo que define la longitud: ángulo OAP.

La latitud abarca un rango de valores que está comprendido entre 0º y 90º hacia el norte, y el mismo rango hacia el sur. Para la localización unívoca de un punto es necesario que se indique si es una latitud norte o sur. Por ejemplo, 43º 70’ 00’’ N (norte) o 43º 70’ 00’’ S (sur).

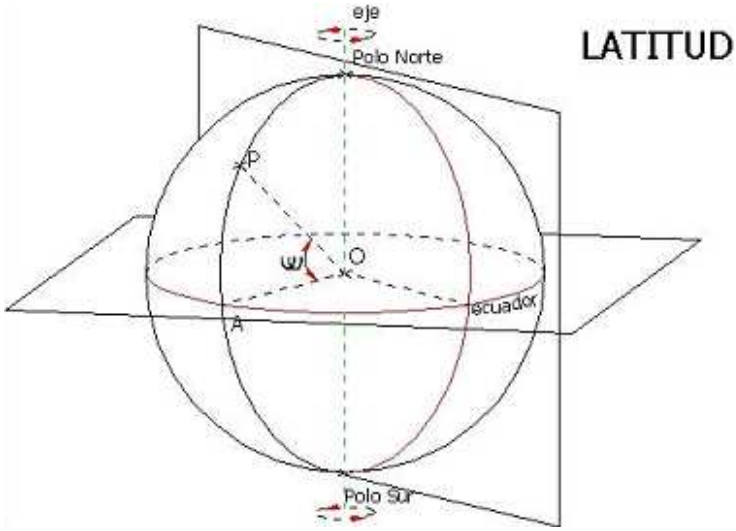

Figura 2.5 Latitud

2.2.1.4 Azimut

Es el ángulo formado por la línea que une el punto de partida y el Norte y la línea que une el punto de partida (origen) con el de llegada (destino). Se expresa en ángulos medidos en el sentido de las agujas del reloj desde la dirección Norte. Varía entre 0º y 360º [18].

El azimut nos permite definir la dirección entre dos puntos sobre un elipsoide. La dirección que define el Norte es uno de los componentes del DATUM y una forma de establecerla es con el acimut.

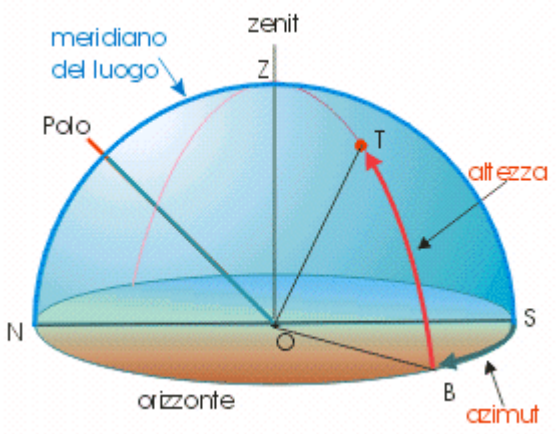

Figura 2.6 Azimut

2.2.1.5 Proyecciones cartográficas

La representación cartográfica de la tierra sobre una superficie plana supone un problema porque no existe modo alguno de representar toda la superficie sin deformarla. Las proyecciones vienen a solventar este problema, estudiando las distintas formas de desarrollar la superficie terrestre minimizando las deformaciones. Existen distintos sistemas de proyección que minimizan la deformación de una de las siguientes magnitudes físicas: superficies, distancias o ángulos.

Se muestra a continuación una clasificación de proyecciones existentes en función de la variable física que conservan tras la proyección [17]:

- **Planas**: Si la superficie a representar es pequeña y la esfericidad terrestre no va a influir en la representación cartográfica, tiene sentido una proyección desde arriba. Se aplican sobre todo en Topografía.
- **Geodésicas**: Es un tipo de proyección en la cual la esfericidad de la tierra repercute enormemente sobre la representación de las posiciones geográficas, sus superficies, sus ángulos y sus distancias. Una de las proyecciones geodésicas de mayor utilización es la proyección *Mercator*, utilizada por el sistema UTM. Dentro de las transformaciones posibles destacamos las siguientes:

- **Conformes**: Aquellas en las que se conservan los ángulos.
- **Equivalentes**: Aquellas en las que la superficie se conserva después de la proyección.
- **Afilacticas**: No conservan ni los ángulos ni las distancias.

Otra forma de clasificar las proyecciones es en función del objeto geométrico utilizado para la proyección. De esta forma podemos hablar de proyecciones cilíndricas, cónicas y planas.

2.2.1.6 Proyección Universal Transversa de Mercator (UTM)

El sistema UTM, Universal Transverse Mercator [17], es un sistema de proyección cartográfica muy extendido, sobre todo a partir de su uso por el servicio de defensa de EEUU en la década de los 40.

La proyección UTM proviene de la proyección transversa de Mercator de Lambert del año 1772 [21], fue modificada posteriormente por Gauss en 1822. La proyección Mercator utiliza para la proyección un cilindro situado de forma tangente al elipsoide en el ecuador. En la figura 2.7 se puede observar la posición del cilindro sobre la superficie terrestre.

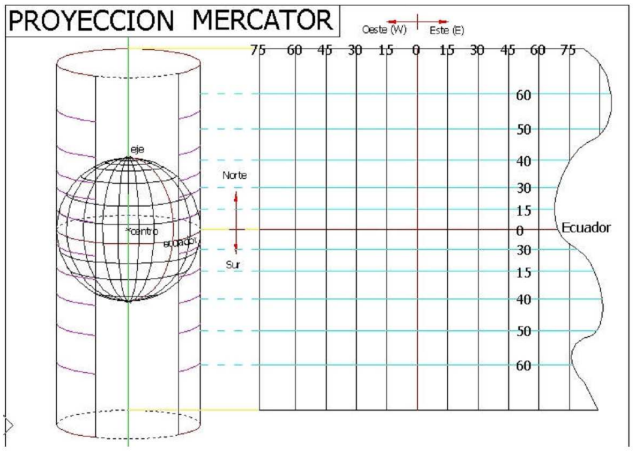

Figura 2.7 Proyección Mercator

La proyección transversal Mercator (UTM) varía la posición del cilindro con respecto a la proyección Mercator. De esta forma el cilindro se dispone de forma transversal al eje de la tierra. En la figura 2.8 se puede observar la disposición del cilindro sobre la superficie terrestre.

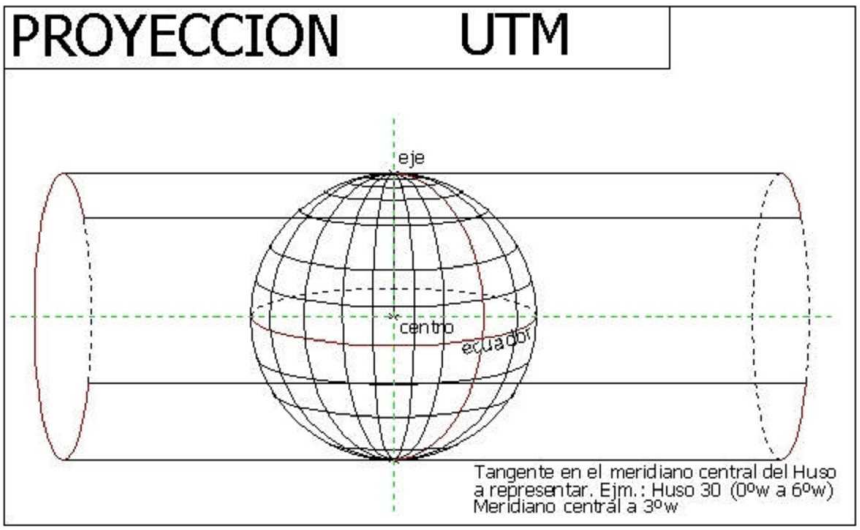

Figura 2.8 Proyección UTM

Con esta disposición del cilindro, tanto los meridianos como los paralelos son líneas rectas y se cortan perpendicularmente. Los meridianos mantienen la distancia entre ellos, pero entre los paralelos la distancia aumenta a medida que un observador se aleja del Ecuador.

La proyección UTM se divide la tierra en 60 husos, con una anchura de 6 grados de longitud, empezando desde el meridiano de Greenwich. Se define un huso como las posiciones geográficas que ocupan todos los puntos comprendidos entre dos meridianos. En la figura 2.9 se puede observar que España está ubica en tres diferentes (29, 30 y 31), lo que ocasiona un problema de representación cartográfico: Una región situada a caballo entre dos husos deberá optar por uno u otro con lo que se incrementan las deformaciones. En la práctica la cartografía española se genera asumiendo que toda la Península se sitúa en el huso 30 por lo que las deformaciones hacia los extremos Este y Oeste son mayores alcanzándose un 4% de distorsión lineal.

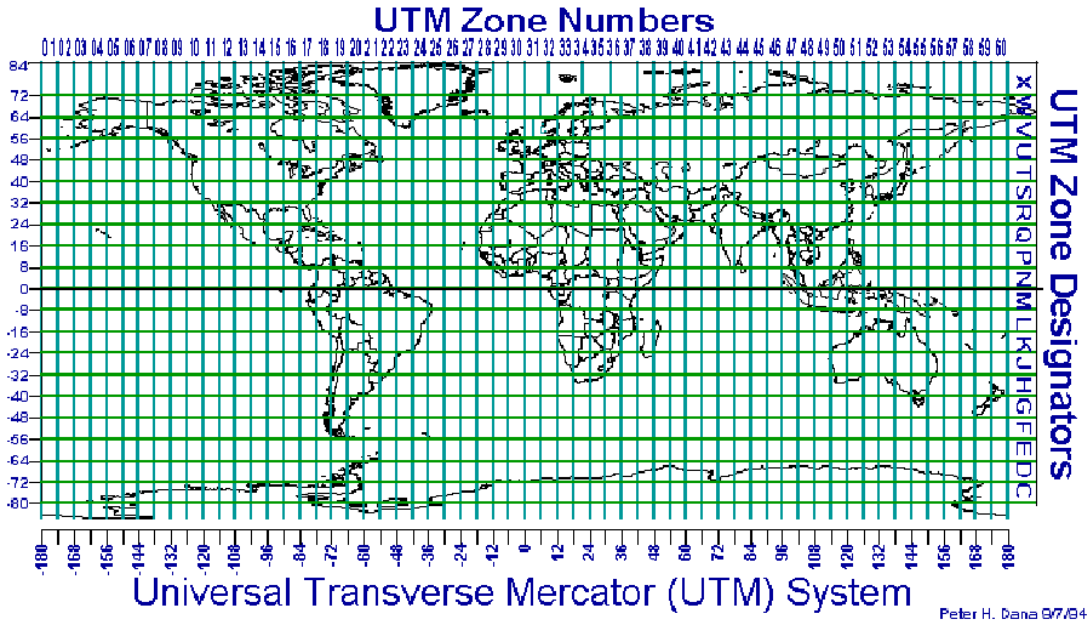

Figura 2.9 Zonas UTM

Para definir la proyección UTM se necesita:

- **Elipsoide de referencia**: En el sistema de proyección UTM se puede usar cualquier elipsoide válido. El recomendado por la *Asociación Internacional de Geodesia*, desde el año 1930 es el elipsoide *Hayford*.
- **DATUM o punto fundamental**: En Europa occidental, el DATUM de referencia es el de *Postdam* en Alemania. En el apartado 2.2.2 explicaremos en profundidad el concepto de DATUM.
- **Sistema de representación plano conforme**: El sistema utilizado es el de *Gauss*, que no es aplicable en grandes extensiones de terreno debido a las deformaciones.

Como conclusión destacamos que el sistema UTM conserva los ángulos pero distorsiona todas las superficies sobre los objetos originales, así como las distancias existentes entre los puntos.

2.2.2 Geodesia

Geodesia es la ciencia que estudia la forma y tamaño de la Tierra y las posiciones sobre la misma. La Geodesia define el geoide como una superficie en la que todos sus puntos experimentan la misma atracción gravitatoria siendo esta equivalente a la experimentada al nivel del mar. Debido a las diferentes densidades de los materiales que componen la corteza y el manto terrestre y a alteraciones debidas a los movimientos isostáticos, esta superficie no es regular sino que contiene ondulaciones que alteran los cálculos de localizaciones y distancias [17]. Es decir, el campo gravitatorio de la tierra no es perfectamente elipsoidal porque la distribución de masas en su interior y en su superficie no es uniforme.

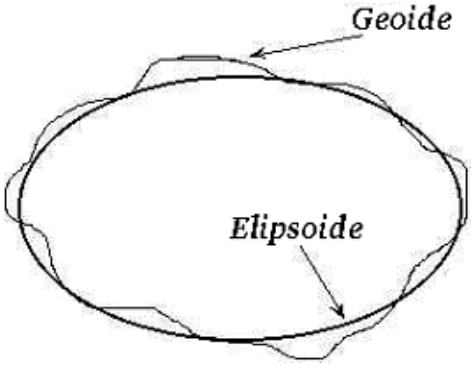

Figura 2.10 Relación del geoide y elipsoide

Para referenciar estos puntos, es necesario establecer un modelo de representación de la superficie terrestre. Como hemos comentado en anteriormente, se utiliza una superficie denominada elipsoide para la representación terrestre.

2.2.2.1 Elipsoide de referencia

Dadas las irregularidades existentes en la superficie terrestre (no extrapolables a todos los puntos) no existe un modelo matemático que represente con satisfacción toda la superficie terrestre. Por esta razón cada continente, nación, etc. utiliza un modelo matemático distinto que se adapte de la mejor forma posible a la superficie de la tierra para la zona que se quiere representar [17].

Un elipsoide es el resultado de revolucionar una elipse sobre su eje. Por lo tanto podemos considerarlo como una simplificación del geoide y se define de manera matemática como un objeto que tiene como características principales dos radios: radio mayor y radio menor. El achatamiento del esferoide (elipsoide de revolución) se define entonces mediante un coeficiente, cuya fórmula es la siguiente: f = (a – b) / a, siendo *a* el radio Mayor y *b* el radio Menor.

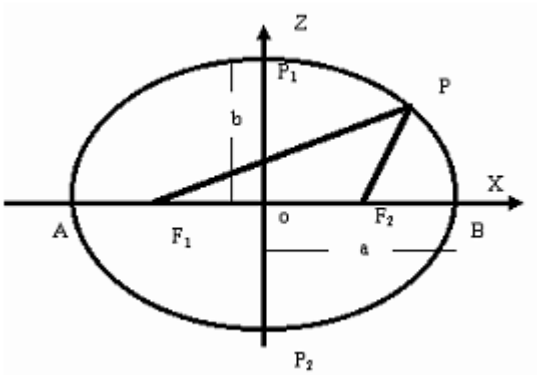

Figura 2.11 Elipsoide

Si partimos de la base de que el radio polar de la tierra es de 6.357 Km (radio b) y que el radio ecuatorial es de 6.378 Km (radio a), podemos concluir que el coeficiente de aplastamiento de la tierra es de aproximadamente 1/300. Como se puede observar la diferencia entre el radio polar y el radio ecuatorial es de 20 Km. De esto deducimos que en caso de utilizar una esfera para representar la tierra podríamos cometer errores de hasta 20 Km. Utilizando un buen elipsoide, la diferencia entre este y el geoide puede reducirse hasta 200 metros.

Como hemos comentado con anterioridad, cada país usa definiciones de elipsoides locales, que representan aproximaciones más ajustadas al geoide en la zona que se quiere representar. Este hecho crea dificultades para estudios en que participen varios países (se pueden utilizar transformaciones matemáticas, pero introduciendo errores). Existen elipsoides que se pueden utilizar para todo el planeta. En concreto el sistema GPS utiliza el WGS84.

Si comparamos el elipsoide con el geoide (ver figura 2.6) podemos observar que las desigualdades de distribución de la gravedad superficial, causa que existan zonas del elipsoide que sobresalgan por encima del geoide y otras en que es al revés. Estas diferencias son causadas por la composición terrestre y por la presencia de una gran masa de agua en los océanos que reduce la atracción terrestre.

2.2.2.2 DATUM

Para elaborar la cartografía de un lugar, es necesario establecer unos parámetros de referencia que relacionen un punto origen del geoide con el del elipsoide. Con el DATUM se puede establecer dicha relación. Se define DATUM como el punto tangente al elipsoide y al geoide, donde ambos coinciden [17].

Un DATUM está compuesto de:

- Un elipsoide definido por sus componentes: un radio menor y mayor y su coeficiente de aplastamiento.
- Un punto denominado fundamental en el que la vertical del elipsoide y el geoide coinciden. Es decir un punto en las que las superficies del elipsoide y del geoide son tangentes. Este punto se define por sus coordenadas geográficas de latitud, longitud y el azimut de una dirección con origen en el punto fundamental.

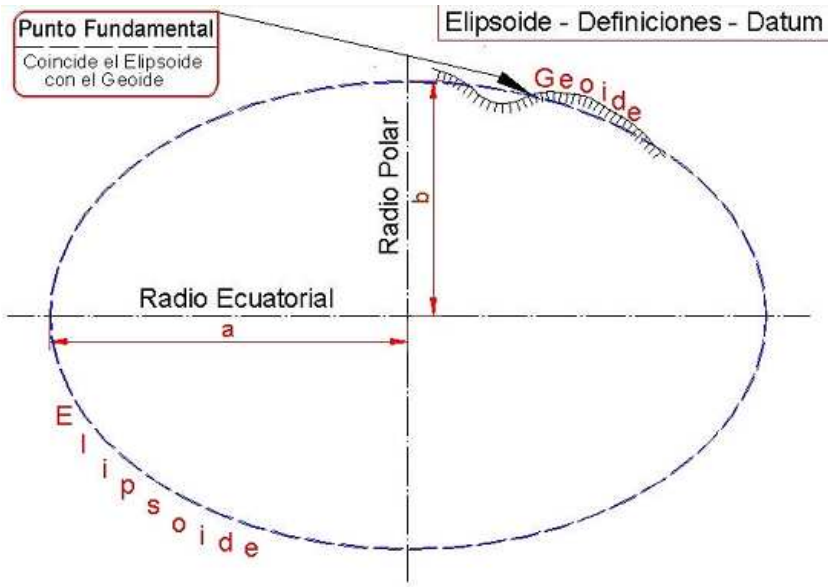

Figura 2.12 Punto fundamental

En el punto fundamental se calculan dos parámetros imprescindibles para el DATUM como son la desviación vertical (Eta) y la desviación sobre el meridiano (Xi).

Un DATUM es uno de los puntos en que coinciden el elipsoide de referencia y la superficie terrestre y se utiliza como referencia para el cálculo y determinación de coordenadas. Define un origen y situación de un sistema de coordenadas válido para una determinada zona de la Tierra. Existen un gran número de DATUM (cada uno adaptado a la zona de aplicación). A modo de ejemplo diremos que en Europa se usa el European 50.

2.3 Representación de información

A continuación explicaremos las distintas técnicas que utilizan los GIS para representar información digitalizada.

Los sistemas más importantes de representación de los datos geográficos son el formato raster y el vectorial. Es muy importante conocer las diferencias entre ambos formatos, ya que una de las fases críticas en el diseño de un GIS es precisamente la decisión del modelo de representación al que se convertirá la información del mundo real.

2.3.1 Modelo raster

El modelo raster se caracteriza por la utilización de una trama de celdas o píxeles en la que cada una de ellas tiene una única propiedad espacial. De esta forma en el formato raster se divide el espacio en un conjunto regular de celdillas, cada una de estas celdillas o píxeles contiene un número que puede ser el identificador de un objeto o del valor de una variable (ejemplos de valores podrían ser la temperatura, altura, niveles de lluvia, etc.).

De esta forma en el modelo raster prevalece la capacidad de asociar información a cada de unas celdillas que se forman sobre la cartografía sobre la exactitud de los límites de la representación.

Los modelos raster son una abstracción de la realidad, donde se representa ésta como una rejilla de celdas o píxeles (ver figura 2.10), en la que la posición de cada elemento es implícita según el orden que ocupa en dicha rejilla. En el modelo raster el espacio no es continuo sino que se divide en unidades discretas.

Las estructuras raster pueden implicar en ocasiones un incremento del espacio de almacenamiento, ya que almacenan cada celda de la matriz sin tener en cuenta si se trata de una entidad o simplemente de un espacio "vacío" (información no significativa).

Lógicamente, para tener una descripción precisa de los objetos geográficos contenidos en la base de datos el tamaño del pixel ha de ser reducido (en función de la escala), lo que dotará a la malla de una resolución alta. Sin embargo, a mayor número de filas y columnas en la malla (más resolución), mayor esfuerzo en el proceso de captura de la información y mayor costo computacional a la hora de procesar la misma. No obstante, el modelo de datos raster es especialmente útil cuando tenemos que describir objetos geográficos con límites difusos, como por ejemplo puede ser la dispersión de una nube de contaminantes, o los niveles de contaminación de un acuífero subterráneo, donde los contornos no son absolutamente nítidos; en esos casos, el modelo raster es más apropiado que el vectorial.

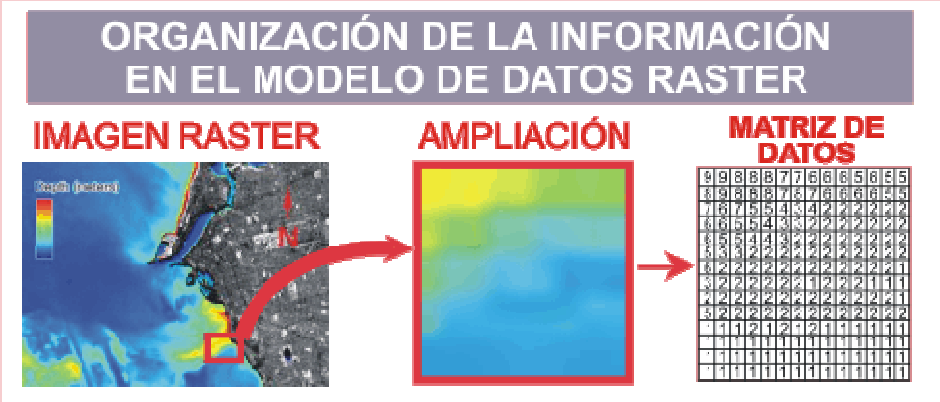

Figura 2.13 Modelo raster

2.3.2 Modelo vectorial

En el formato vectorial los diferentes objetos se representan como puntos, líneas o polígonos. Para ello se utilizan vectores definidos por pares de coordenadas relativas a algún sistema cartográfico. Con un par de coordenadas y su altitud gestionan un punto, con dos puntos generan una línea, y con una agrupación de líneas forman polígonos. Es decir, el formato vectorial se basa en la digitalización de los objetos del mundo real en tres tipos de elementos que representan la información (primitivas geométricas):

- **Nodo**: es la unidad básica para representar entidades con posición pero sin dimensión
- **Línea o arco**: representa entidades de una dimensión y está restringido a línea recta en algunas implementaciones.
- **Polígono o área**: se utiliza para representar las entidades de superficie (2 dimensiones).

En el modelo vectorial prima la precisión y la eficiencia del proceso sobre la capacidad de almacenamiento de información. En el modelo vectorial, los procedimientos de análisis son mucho más laboriosos que el modelo raster.

Por otro lado el modelo vectorial nos permite establecer relaciones entre los elementos. Por ejemplo podemos establecer líneas que representan carreteras que se cruzan en una intersección o líneas que representan carreteras que se cruzan a distintos niveles (por ejemplo un puente).

En general, el modelo de datos vectorial es adecuado cuando trabajamos con objetos geográficos con límites bien establecidos, como pueden ser fincas, carreteras, etc.

En la figura 2.14 se puede observar como el modelo vectorial representa el mundo real mediante el uso de polígonos y líneas. En la parte izquierda de la imagen podemos observar que para definir una línea es necesario identificar el nodo de origen y el nodo de fin con sus coordenadas. En la figura de la derecha podemos observar que un polígono está formado por un conjunto de arcos o líneas que a su vez están compuestos de un nodo de origen y un nodo de destino.

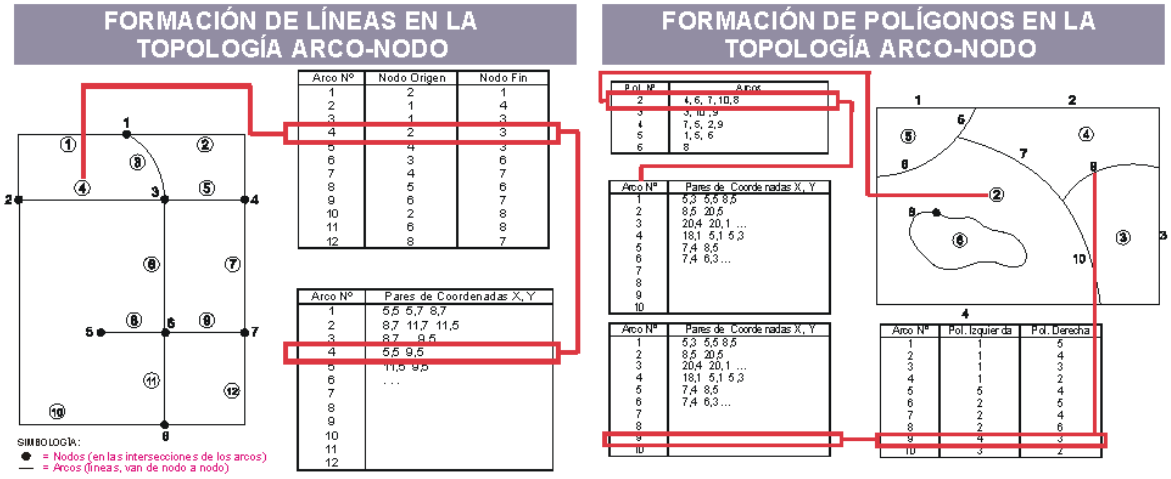

Figura 2.14 Modelo vectorial

2.3.3 Comparación de modelo vectorial y raster

Como conclusión de las explicaciones que se han realizado en los apartados anteriores destacamos que de los dos modelos más utilizados para la representación de la información geográfica en los GIS, el modelo vectorial será más adecuado para los siguientes casos:

- Construcción de mapas precisos con datos de localización definidos y con relaciones entre los elementos.
- Análisis de datos en los que prima la exactitud de la localización de la información.

Por el contrario será mejor utilizar el modelo raster en los siguientes casos:

- Composiciones de mapas en los que se representa información con límites difusos, sobre los que sólo es necesario conocer su posición global, pero de forma precisa y detallada.

En algunos casos podremos combinar ambos modelos para adaptar el GIS a las necesidades concretas de la aplicación.

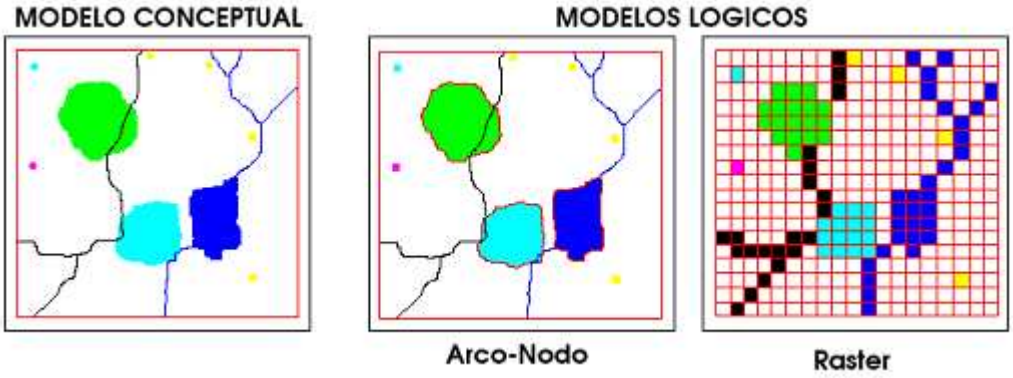

Figura 2.15 Comparación de modelo vectorial y raster

3 OGC

A lo largo de los últimos años, la industria de software ha desarrollado diferentes métodos para obtener, analizar, procesar, y mostrar la información geográfica. A lo largo de este tiempo, la falta de estándares ha provocado que se hayan desarrollado formatos muy diferentes para representar la información dentro de los Sistemas de Información Geográfica. Esto ocasiona que los sistemas generados no se puedan comunicar entre sí, lo que limita las posibilidades de conexión y de compartición de información [28].

El Open Geospatial Consortium (OGC) es una organización internacional que agrupa a 341 compañías y organismos públicos[1] que trabaja en la definición de especificaciones públicas para el intercambio de información entre los sistemas de información geográfica [28]. El principal objetivo de dicho consorcio es la definición de estándares abiertos e interoperables dentro de los Sistemas de Información Geográfica que faciliten el intercambio de la información geográfica entre distintos sistemas GIS en beneficio de los usuarios y de la industria.

3.1 Introducción

Los trabajos realizados por el OGC desde 1994 han propiciado la publicación de una serie de estándares que permitan el desarrollo de aplicaciones GIS y el intercambio de información georreferenciada. Las especificaciones de estos estándares establecen unas bases que posibilitan el desarrollo de aplicaciones Open Source y la utilización de arquitecturas basadas en servicios que ofrecen soluciones mucho más modulares. Los sistemas OpenGIS representan una evolución de los sistemas SIG tradicionales, en los cuales las aplicaciones monolíticas y propietarias se vuelven interoperables y extensibles.

Todas las especificaciones en las que trabaja el consorcio OGC se basan en un modelo de referencia de OpenGIS (ORM). El modelo de referencia OpenGIS es una aplicación del Modelo de Referencia para Procesamiento de Distribución Abierta (RM-ODP). Como todo RM-ODP está construido y definido en torno a 5 puntos de vista: punto de vista de la información, punto de vista de la empresa, punto de vista de la computación, punto de vista de ingeniería y punto de vista de tecnología .

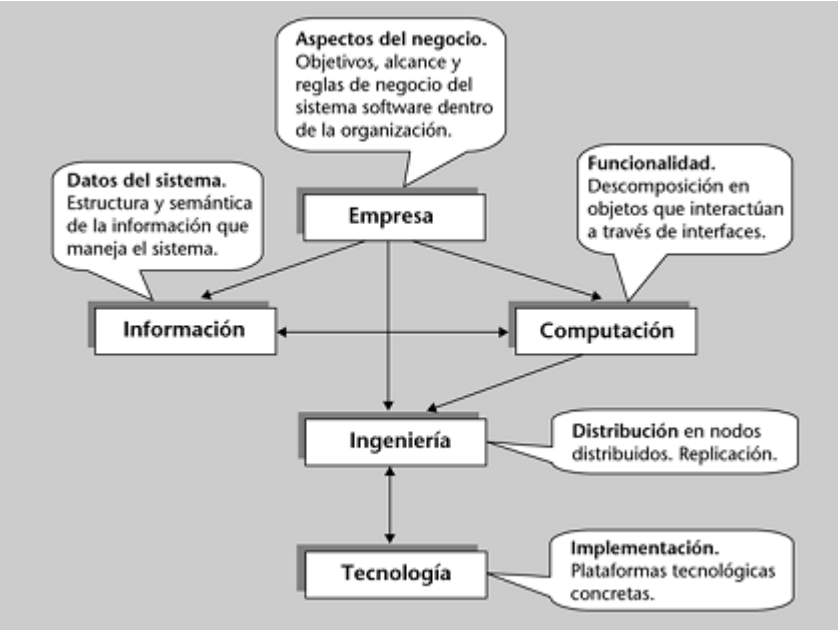

Figura 3.1 Puntos de vista RM-ODP

[1] Dato consultado en la página web de la organización (marzo 2007)

3.2 Estándares publicados

Todos los estándares publicados por el OGC trabajan en dos líneas:

- Construir servicios que consuman o proporcionen información geográfica.
- Definir lenguajes basados en XML que permitan la conexión y la comunicación entre distintos GIS.

El modelo de referencia de OpenGIS establece 6 categorías de servicios:

Categoría de servicio	Descripción
Interacción humana	Servicios para gestionar interfaces de usuario, gráficos, multimedia y para presentar documentos compuestos.
Gestión de la información	Servicios para gestionar el desarrollo, manipulación y almacenamiento de metadatos, esquemas conceptuales y conjuntos de datos.
Volumen de trabajo	Servicios de apoyo a tareas específicas o actividades relacionadas con el trabajo
Procesamiento	Servicios de ejecución de cálculos a gran escala. Un servicio de procesamiento no incluye capacidades para proveer almacenamiento persistente de datos o transferencia de datos a redes. Sub-categorías de procesamiento: • Servicios de procesamiento geográfico espacial • Servicios de procesamiento geográfico temático • Servicios de procesamiento geográfico temporal • Servicios de procesamiento geográfico de metadatos
Comunicación	Servicios de codificación y transferencia de datos a través de redes.
Gestión del sistema	Servicios de gestión de componentes del sistema, aplicaciones y redes.

Tabla 2 Categorías de servicio definidas por OpenGIS

A continuación presentamos las especificaciones aprobadas del OGC más importantes:

3.2.1 Servicio de catalogo (CSW)

Este servicio OGC tiene como objetivo la publicación y búsqueda de información que describe datos, servicios, aplicaciones y en general todo tipo de recursos de información geográfica. Los servicios de catálogo son necesarios para proporcionar capacidades de búsqueda e invocación sobre los recursos registrados dentro de un determinado ámbito de servicios web que cubran distintas especificaciones de OGC. La especificación de OGC establece cómo debe ser un Servicio de Catálogo estándar e interoperable [29].

3.2.2 Acceso de las entidades simples (Simple Features - SQL, CORBA, OLE)

El punto de vista de ingeniería del modelo de referencia de OpenGIS (ORM) establece que en los cimientos de la arquitectura se encuentran los protocolos de comunicaciones, con excepción de los datos puramente binarios. Los datos estructurados se pueden codificar como XML. Los formatos utilizados para la codificación se describen en un algún lenguaje de esquemas como puede ser DTD (*Document Type Definition*: descripción de estructura y sintaxis de un documento XML), *XML Schema* (lenguaje de esquema utilizado para describir la estructura y las restricciones de los contenidos de un documento XML), RDF (*Resource Description Framework*: framework para describir e intercambiar metadatos) o XMI (XML de Intercambio de Metadatos). El OGC ha definido una familia de esquemas para la codificación de entidades simples y “bien conocidas” en binario (WKB), en texto plano (WKT) así como también en GML, para codificar entidades en XML [30][31].

Entendemos por *Feature* un objeto de información geográfica. Tal y como hemos presentado en el apartado 2.3.2, estos objetos pueden ser puntos, líneas, polilíneas, polígonos y en general cualquier elemento geométrico de base cartográfico vectorial.

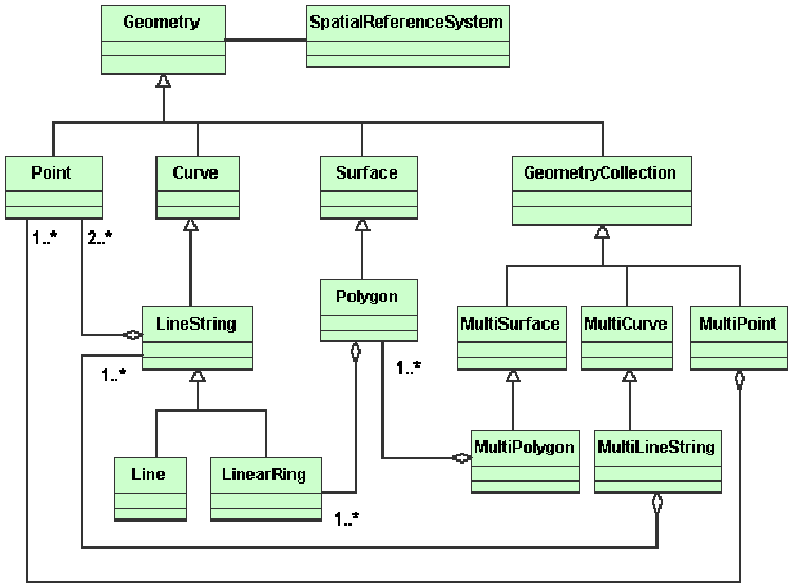

Figura 3.2 Simple features

En las aplicaciones GIS existen una serie de estándares de enlace de datos ampliamente utilizados por la industria, como son COM, CORBA, J2EE y SQL. Por ello el OGC ha definido una serie de especificaciones para el acceso y la manipulación de *simple features* mediante la utilización de estos estándares de enlace de datos.

3.2.3 Servicio de Coberturas (WCS)

Este servicio OGC tiene como objetivo obtener e intercambiar información geoespacial en forma de objetos (coberturas). Una cobertura por tanto es un objeto al que se le asocian posiciones y valores dentro de un espacio. Por ejemplo una imagen por satélite o una imagen raster podrían ser una cobertura [32].

Una diferencia fundamental entre los WCS y otros servicios OGC es que retornan objetos con datos asociados y con distribución continua.

El OGC define que un servicio WCS tiene que implementar las siguientes operaciones:

- *GetCapabilities*: retorna metadatos del servicio y de las coberturas que ofrece. El servicio retorna un documento en XML que describe el servicio y con una breve descripción de las coberturas que ofrece.
- *DescribeCoverage*: retorna una descripción detallada de una o varias coberturas. El servicio responde con una descripción completa de una o más coberturas.
- *GetCoverage*: permite obtener una cobertura o parte de ella. El servicio responderá con la cobertura solicitada en el formato indicado.

3.2.4 Servicios de Transformación de Coordenadas (WCTS)

Este servicio OGC tiene como objetivo transformación el sistema de coordenadas de un conjunto de datos geográficos a otro sistema. La utilización de este tipo de servicios nos permitirá transformar datos geográficos de un sistema de coordenadas a otro, de forma que puedan ser consultados o intercambiados entre distintos sistemas GIS [33].

3.2.5 Lenguaje de marcado para información geográfica (GML)

El OGC ha definido un lenguaje de marcado para información geográfica (GML) para la modelización, transporte y almacenamiento de Información Geográfica. Para conseguirlo, GML ofrece una amplia variedad de objetos que nos permiten describir la geografía (entidades, sistemas de coordenadas, geometría, topología, tiempo, etc.) [34].

GML es una gramática XML definida como un schemas en XML. Los schemas de la especificación de GML los podemos personalizar para nuestro modelo de datos extendiéndolos o especializándolos.

En la actualidad GML está soportado por una gran cantidad de fabricantes de software, lo que permite que su uso sea un mecanismo ideal para el intercambio de datos entre distintos sistemas.

A continuación presentamos la jerarquía de clases de GML. En esta jerarquía se puede observar que tenemos disponible distintas clases para la descripción de la información geográfica: primitiva geométrica, Geometría, topología, cobertura, sistema de coordenadas, estilo y marca de tiempo.

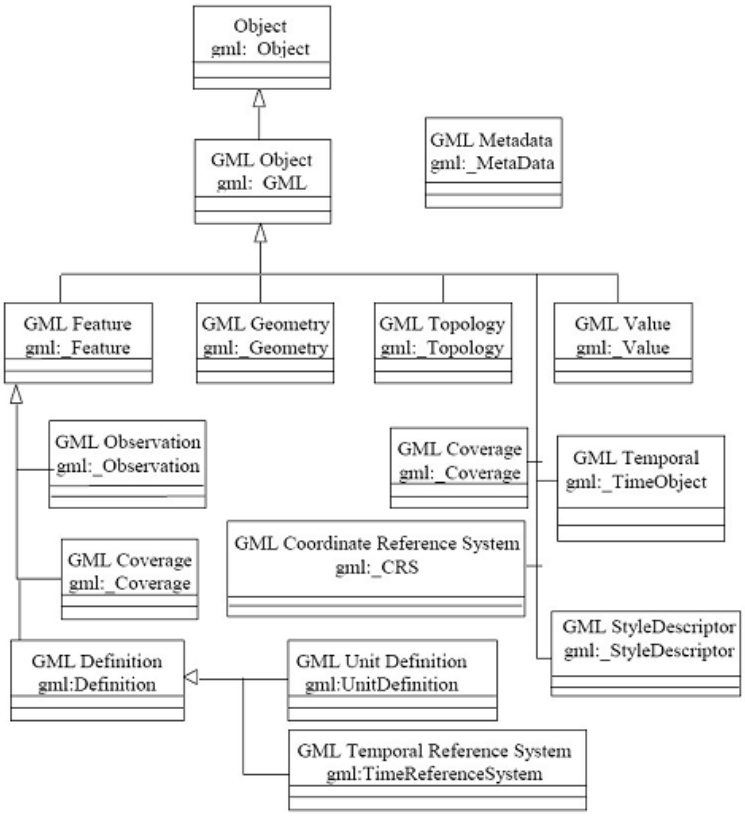

Figura 3.3 Jerarquía de clases de GML

Mediante la aplicación de transformaciones u hojas de estilo podremos visualizar la información geográfica contenida en GML en cualquier formato gráfico.

De forma general podemos decir que GML nos permite describir entidades geográficas a través de sus propiedades.

Como principal desventaja de GML tenemos que decir que al ser un fichero de texto, este puede llegar a ocupar bastante espacio, siendo necesaria la compresión de la información o la transformación en XML Binario.

3.2.6 Especificación de codificación de los Filtro (FE)

El OGC ha definido un lenguaje XML que permite el filtrado de información geográfica. Será imprescindible su utilización para consultar a servicios web de mapas y de objetos gráficos en caso de que necesitemos filtrar las respuestas ofrecidas por el mismo [35].

La especificación de OGC define el esquema de un documento FE, y por lo tanto debemos respetar esta definición para que un documento FE se considere bien formado. Un documento FE tendrá el siguiente esquema:

- Filtro. Definición del filtro y lista de operaciones
- Operaciones: Podemos indicar operaciones lógicas, de comparación o especiales.

3.2.7 Servicios de Mapas por el Web (WMS)

El OGC ha definido la especificación de los servicios web de mapas (WMS). Este produce mapas de datos espaciales referidos de forma dinámica a partir de información geográfica. Los mapas producidos por WMS se generan normalmente en un formato de imagen [36].

En el apartado 4.4 se puede encontrar más información sobre este tipo de servicios.

3.2.8 Servicio de objetos geográficos (WFS)

El OGC ha definido la especificación de los servicios de consulta de objetos de información geográfica (WFS). Este servicio permite consultar y recuperar datos vectoriales de forma remota. Habitualmente los datos proporcionados por este tipo de servicios están codificados en GML. Con GML se codifican vectores de datos. Por otro lado, los datos geométricos pueden ir acompañados de información no espacial asociada al objeto consultado [37].

En el apartado 4.3 se puede encontrar más información sobre este tipo de servicios.

3.2.9 Descriptor de estilos de capa (SLD)

SLD es una especificación de OGC que describe un lenguaje para producir mapas georreferenciados con estilos definidos por usuario. SLD amplia la especificación del servicio web de mapas (WMS) para incorporar capas definidas por el usuario sobre los mapas, definiendo que elementos se muestran y cuales se ocultan. SLD al igual que GML es un lenguaje XML que contiene una secuencia de definiciones de estilos para las capas que permitan modificar la apariencia del mapa [38].

La especificación de OGC define el esquema de un documento SLD, y por lo tanto debemos respetar esta definición para que un documento SLD se considere bien formado. Un documento SLD tendrá el siguiente esquema:

- **Cabecera**: En ella se especifica el orden en el que se dibujarán las capas.
- **Elementos**:
    - *NameLayer*: Define la capa que queremos modificar y el estilo del usuario.
    - *UserLayer*: Capa definida por el usuario
    - *UserStyle*: Definición del estilo especificado por el usuario que se aplicará a la capa indicada.

3.3 Web Feature Service

Como hemos comentado en el apartado 3.2.8, un servicio WFS (Web Feature Service) es un servicio web que publica información geográfica en formato vectorial, ofreciendo un servicio de consulta y gestión de datos geográficos on-line.

Un servicio WFS estándar debe cumplir las especificaciones definidas por el OGC. El cumplimiento de las mismas garantizará la interoperatibilidad entre sistemas (será capaz de entender y responder a cualquier petición realizada según el estándar definido por el OGC en el “Web Feature Service (WFS) Implementation Specification 1.0”) [36].

3.3.1 Tipos de WFS

El estándar de la OGC nos permite clasificar los WFS en dos tipos bien diferenciados en función del tipo de operaciones que nos permitan realizar.

- WFS básico: Servicio web que sólo permite la consulta de información georreferenciada en formato vectorial. Todas las operaciones que implementan son de consulta de información (sobre el servicio o sobre la información geográfica).
- WFS Transaccional: Servicio web que además de las operaciones de consulta que incluyen los WFS básicos, permiten la edición (alta, modificación y baja) de los objetos de información geográfica publicados con el servicio WFS.

3.3.2 Operaciones

El estándar del OGC define que todo WFS debe implementar las siguientes operaciones [36]:

1. *GetCapabilities*: describe las capacidades del servicio, que tipos de objetos puede servir y que operaciones soporta en cada tipo de objeto (obligatorio).
2. *DescribeFeatureType*: Describe la estructura del tipo de objeto solicitado (obligatorio).
3. *GetFeature*: Devuelve el objeto solicitado en formato GML (obligatorio).
4. *GetFeatureWithLock*: Devuelve el objeto solicitado en formato GML bloqueando el registro (opcional).
5. *Transaction*: Crear, Actualizar y Borrar objetos geográficos (opcional).
6. *LockFeature*: Petición de bloqueo sobre uno o más objetos mientras dure la transacción (opcional).

Todas las operaciones que se identifican como obligatorias tienen que estar implementadas en todos los servicios WFS. Las operaciones de tipo opcional sólo es necesario implementarlas en los WFS transaccionales.

Cuando solicitamos un objeto a un servicio WFS este responderá con datos en formato GML. Como hemos indicado con anterioridad el formato GML es una especificación de XML definida por el OGC que permite intercambiar datos geográficos. Mediante la aplicación de una hoja de estilo o una transformación podremos convertir el GML en una imagen en binario.

3.3.3 Secuencia de una petición de consulta

A continuación presentamos un diagrama de secuencia de una petición de información a un servicio WFS. En la secuencia se representarán todos los pasos necesarios desde la consulta de información del servicio hasta la petición de un objeto concreto.

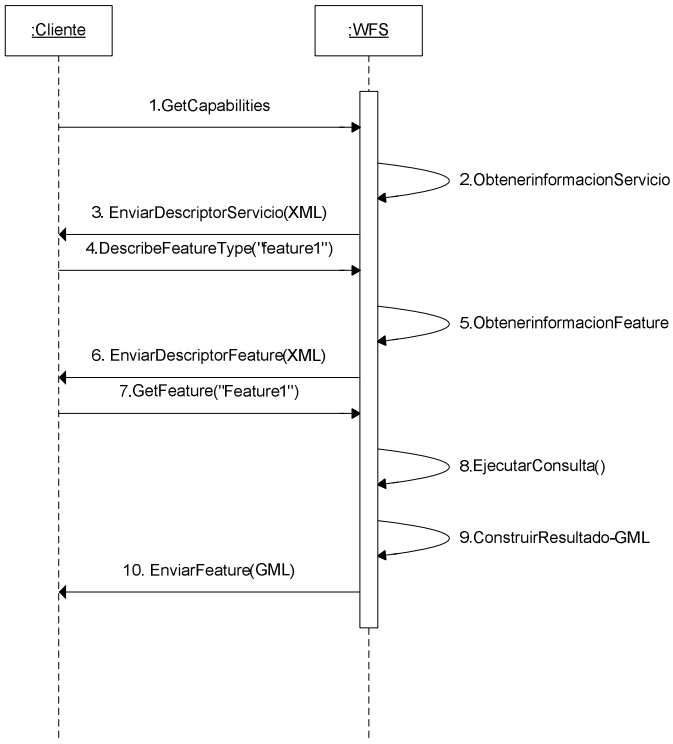

Figura 3.4 Petición WFS

Las peticiones al servicio web las tenemos que enviar utilizando el protocolo http. Las peticiones las podemos enviar utilizando dos métodos:

- GET: Todos los datos de la petición van insertados en la URL. Conviene recordar que algunos navegadores y servidores web establecen longitudes máximas para la URL (Opera establece un límite de 4 Kb, Internet Explorer establece un límite de 2 Kb, Firefox establece una limitación de 65.536 caractres, Apache establece un límite de 8.192 bytes y Internet Information Service permite configurar dicho límite).
- POST: Los datos se envían en XML.

A continuación presentamos un ejemplo de llamada al servicio WFS de la Dirección General de Catastro: http://ovc2.catastro.meh.es/cartografia/wfs/servidorwfs.aspx?municipiodgc=47900&PROVINCIAINE=47&service=wfs&version=1.0.0&request=getfeature&typename=PARCELA&bbox=354500,4611070,355000,4611450. En este ejemplo se puede observar que se solicita al servicio un municipio y provincia dentro de un encuadre determinado para que muestre la entidad “parcela”.

3.3.4 Productos comerciales

A la hora de implementar un WFS podemos construir un servicio web desde cero que cumpla las especificaciones establecidas por el OGC o utilizar alguno de los productos que ofrecen distintos fabricantes o los numerosos productos de distribución libre que soportan la creación de servicios WFS. A continuación destacamos aquellos productos de mayor implantación:

- **MapServer**: entorno de desarrollo de código abierto (Open Source) para la creación de aplicaciones GIS, que permiten la gestión, visualización, consulta y análisis de información geográfica a través de la red mediante la tecnología Internet Map Server (IMS) de la Universidad de Minesota. MapServer está disponible para Linux y Windows y puede emplearse para crear servicios WFS no transaccionales (cumple la especificación de OGC 1.0). El sistema está desarrollado en C y su código es de libre distribución.
- **GeoServer**: Sistema desarrollado en J2EE de código abierto (Open Source) que permite la implementación de GIS. Permite la conexión con orígenes de datos de Oracle, PostGIS, ArcSDE o Shape. Con GeoServer podemos crear un servicio WFS transaccional (cumple la especificación de OGC 1.0).
- **GO Publisher WFS Edition**: sistema comercial de la compañía Snowflake que permite la construcción de servicios WFS. Este sistema permite la conexión con orígenes de datos de Oracle, SQL Server, DB2, MySQL, MS Access, MS Excel.

3.4 Web Map Service

Como hemos comentado en el apartado 3.2.7, un servicio WMS (Web Map Service) es un servicio web que publica información geográfica en formato raster, ofreciendo un servicio de consulta y gestión de datos geográficos on-line.

Un servicio WMS estándar debe cumplir las especificaciones definidas por el OGC. El cumplimiento de las mismas garantizará la interoperabilidad entre sistemas (será capaz de entender y responder a cualquier petición realizada según el estándar de definido por el OGC en el “Web Map Service (WMS) Implementation Specification”) [37][39].

La WMS es una de las especificaciones de OGC más extendida. Un servicio WMS al recibir una petición HTTP de un cliente en la que se solicita un mapa, recupera los objetos geográficos que componen el mapa de una base de datos o un servidor de información geográfica y, de acuerdo a unas características de estilo, genera un mapa en alguno de los formatos soportados para representar información gráfica. Los formatos más habituales son GIF (Graphics Interchange Format), JPEG (Joint Photographic Experts Group) o PNG (Portable Network Graphics). Tal y como explicaremos en el apartado 3.4.1 los formatos soportados por el servicio pueden obtenerse mediante la operación *GetCapabilities*.

El mapa que retorna un WMS puede contener distintas capas de información. Las capas se muestran y pueden solicitarse individualmente o mediante un mapa compuesto.

Entendemos por mapa una representación de la información geográfica en formato digital que permita su tratamiento y visualización por equipos de procesamiento de información. Los mapas pueden contener información de capas raster (servicios WMS) y/o vectorial (servicios WFS).

Un WMS describe la apariencia de un mapa en términos de capas con estilo. Un mapa está constituido por un número concreto de capas con estilo colocadas unas sobre otras en un orden especificado. Los usuarios pueden definir mapas más sencillos o complejos añadiendo o eliminando algunas de las capas que lo componen. El estilo visual de cada una de las capas del mapa, es definido en un WMS utilizando el lenguaje Styled Layer Descriptor (podemos consultar más información sobre los SLD en el apartado 3.2.9). Con este lenguaje, los clientes pueden definir estilos con los que quieren que se representen objetos geográficos en cada capa del mapa.

En la actualidad existe una proposición de una extensión de la especificación de WMS para soportar mapas activos (AWMS). Esta proposición no está aprobada en la actualidad (la última versión liberada es la 0.9).

3.4.1 Operaciones

El estándar del OGC define que todo WMS debe implementar las siguientes operaciones (2 obligatorias y 1 opcional) [41]:

1. *GetCapabilities* (obligatoria): Esta funcionalidad permite obtener información de las funcionalidades soportadas por el servicio, información específica sobre las capas de información geográfica disponibles, formatos soportados, sistemas de referencia soportados y encuadres posibles para cada uno de ellos. La información es devuelta al cliente codificada en un documento XML.
2. *GetMap* (obligatoria): Esta funcionalidad es la más importante de un servicio WMS. Con esta petición le solicitamos al servicio web que genere un mapa a partir de un conjunto de entidades geográficas y de una descripción de las capas que lo componen junto con el estilo de visualización correspondiente a cada una de ellas. Esta operación acepta los siguientes parámetros:
    - Sistema de Referencia Espacial (SRS) en el que se representará el mapa.
    - *Bounding Box*: Rectángulo que define la extensión geográfica del mapa en el espacio especificado por el SRS.
    - *Layers*: Lista de capas de información geográfica que componen el mapa.
    - *Styles* (opcional): Estilos de representación visual, descritos por un documento o conjunto de documentos SLD, que se utilizarán para generar el mapa.
    - *Format*: Formato en el que se codificará la imagen retornada por el servicio.
    - *Width y Height*: Tamaño de la imagen retornada por el servicio.
    - Otros parámetros, como color de fondo, trasparencias, etc.
3. *GetFeatureInfo* (opcional): devuelve información sobre entidades u objetos particulares mostrados en el mapa. Responde a consultas básicas sobre el contenido del mapa.

Todas las operaciones que se identifican como obligatorias tienen que estar implementadas en todos los servicios WMS.

3.4.2 Tipos de WMS

El estándar de la OGC nos permite clasificar los WMS en dos tipos bien diferenciados en función del tipo de operaciones que nos permitan realizar.

- WMS básico: Servicio web que sólo permite la petición de mapas con información georreferenciada en formato raster. Es decir, el servicio sólo implementa las operaciones definidas como obligatorias: GetCapabilities y GetMap.
- WMS Consultable: Servicio web que además de las operaciones que incluyen los WMS básicos, permiten la consulta de datos sobre las entidades representadas en el mapa de la información publicada en el servicio WMS. Es decir, el servicio implementa además de las operaciones definidas como obligatorias la operación *GetFeatureInfo*.

3.4.3 Secuencia de una petición de consulta

A continuación presentamos un diagrama de secuencia de una petición de información a un servicio WMS. En la secuencia se representarán todos los pasos necesarios desde la consulta de información del servicio hasta la petición de un objeto concreto.

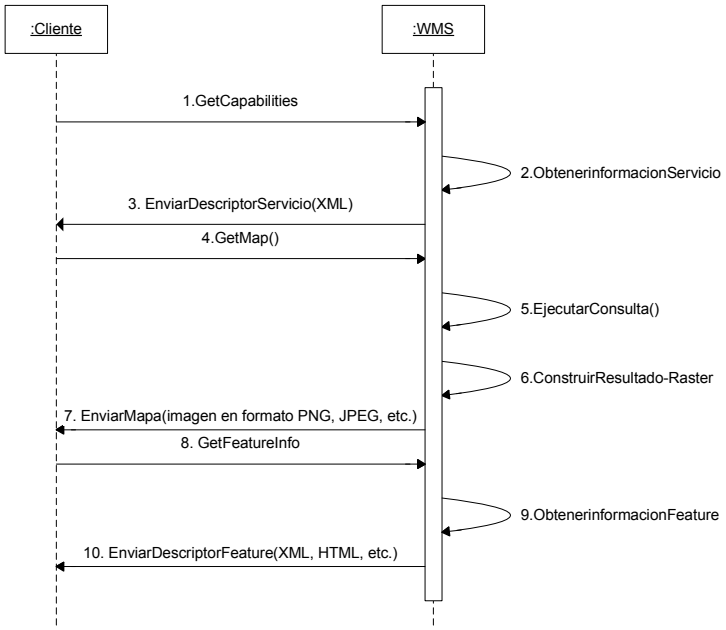

Figura 3.5 Petición WMS

Las peticiones al servicio web las tenemos que enviar utilizando el protocolo http. Las peticiones las podemos enviar utilizando dos métodos:

- GET: Todos los datos de la petición van insertados en la URL. Conviene recordar que algunos servidores web establecen longitudes máximas para la URL.
- POST: Los datos se envían en XML.

A continuación presentamos un pequeño ejemplo de una petición Getmap a un servicio WMS. Para el ejemplo utilizamos el WMS de la Dirección General de Catastro y utilizamos la siguiente URL: http://ovc.catastro.meh.es/Cartografia/WMS/ServidorWMS.aspx?SERVICE=WMS&VERSION=1.1.1&REQUEST=GetMap&SRS=EPSG:23030&BBox=256000,4038700,256100,4038900&WIDTH=600&HEIGHT=600&FORMAT=image/png&TRANSPARENT=False.

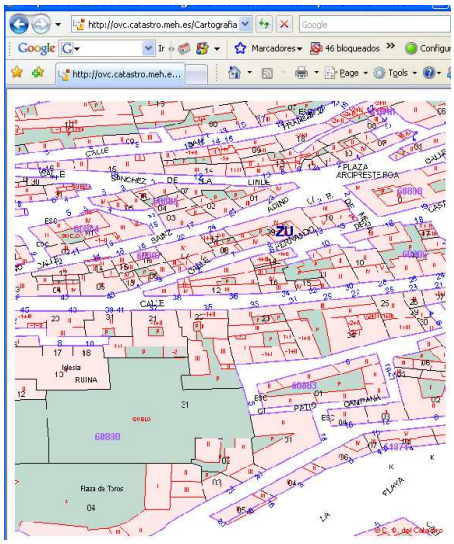

Figura 3.6 Ejemplo de llamada a WMS

La Xunta de Galicia dispone de un servicio WMS denominado SITGA [46]. Consultando dicho servicio mediante la operación GetCapabilities podemos consultar las funcionalidades que oferta dicho servicio (http://ideg.xunta.es/WMS/Request.aspx?SERVICE=WMS&VERSION=1.1.1&REQUEST=GetCapabilities):

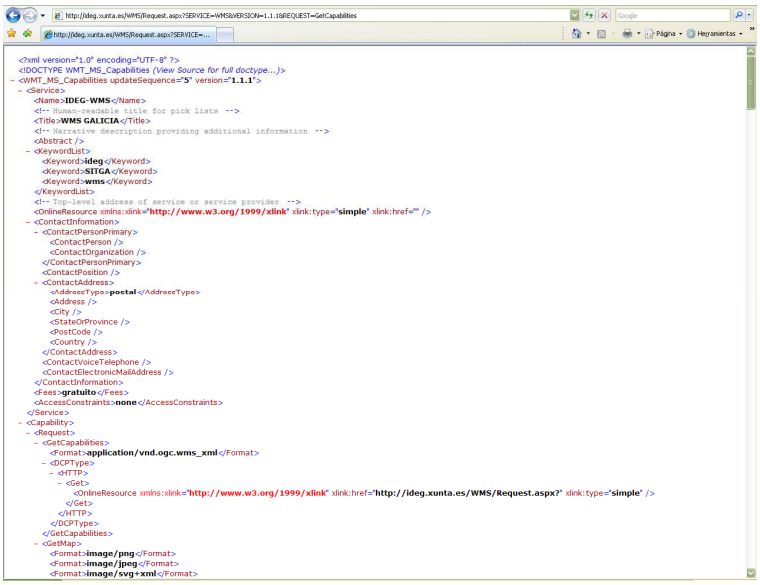

Figura 3.7 Respuesta GetCapabilities SITGA

Con la respuesta que nos da está operación podemos conocer que capas tiene el servicio y cuáles son las coordenadas mínimas y máximas que gestiona. Con estos datos pedimos un mapa de Galicia que tenga una capa para las autopistas y otro para las provincias (http://ideg.xunta.es/WMS/Request.aspx?SERVICE=WMS&VERSION=1.1.1&REQUEST=GetMap&SRS=EPSG:23029&BBox=358966.063872504,4622371.50184085,788234.050477277,4856636.05143132&WIDTH=800&HEIGHT=600&FORMAT=image/png&TRANSPARENT=False&LAYERS=AUTOESTRADA_AUTOVIA_25,PROVINCIA_10001):

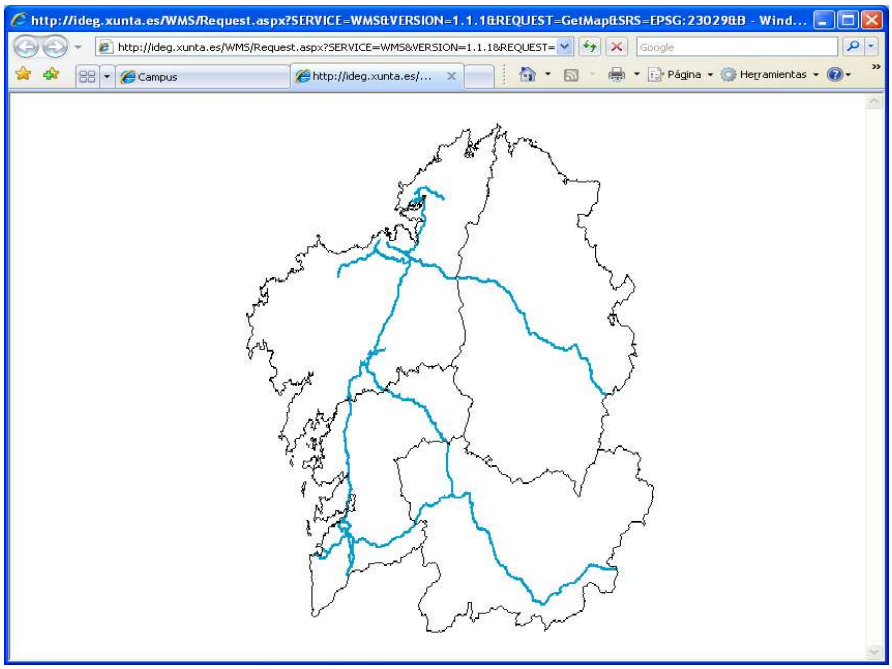

Figura 3.8 Respuesta GetMap SITGA

3.4.4 Productos comerciales

A la hora de implementar un WMS podemos construir un servicio web desde cero que cumpla las especificaciones establecidas por el OGC o utilizar alguno de los productos que ofrecen distintos fabricantes o los numerosos productos de distribución libre que soportan la creación de servicios WMS. A continuación destacamos aquellos productos de mayor implantación:

- **MapServer**: entorno de desarrollo de código abierto (Open Source) para la creación de aplicaciones GIS, que permiten la gestión, visualizar, consultar y análisis de información geográfica a través de la red mediante la tecnología Internet Map Server (IMS). MapServer está disponible para Linux y Windows y puede emplearse para crear servicios WMS (cumple la especificación de OGC 1.3).
- **GeoServer**: Sistema desarrollado en J2EE de código abierto (Open Source) que permite la implementación de GIS. Permite la conexión con orígenes de datos de Oracle, PostGIS, ArcSDE o Shape. Con GeoServer podemos crear un servicio WMS (cumple la especificación de OGC).
- **ArcGIS**: este potente sistema de información geográfica de la compañía ESRI dispone de una extensión que permite implementar con *ArcGIS* un servicio WFS que se conecte con *ServerGIS*.

3.4.5 Clientes ligeros

Dentro de las distintas alternativas que disponemos para la presentación de los mapas retornados por los servicios WMS tenemos la opción de utilizar clientes ligeros que normalmente estarán integrados con el navegador (normalmente un plug-in instalado en el navegador). Dentro de las distintas opciones que disponemos destacamos la ofertada por el proyecto *OpenLayers*. *OpenLayers* [55] está construido íntegramente en JavaScript y permite la consulta de información geoespacial ya sea de servidores que cumplan con el estándar WMS y WFS del OpenGeospatial Consortium (OGC) o de los principales servicios comerciales que hay en la actualidad como son *Google Maps*, *Virtual Earth*, *TerraServer*, *WorldWind*.

En conclusión, *OpenLayers* es una opción de código abierto similar a la API de Google Maps, pero con la libertad de acceder a más fuentes de datos (tanto las que cumplen los estándares OGC como algunos de las servicios comerciales más extendidos).

3.4.6 Clientes pesados

Otra alternativa para la visualización de los resultados retornados por un servicio de mapas es la utilización de clientes pesados con funcionalidades avanzadas que nos permitan un mayor tratamiento de los mapas visualizados.

Dentro de los clientes pesados disponemos de distintas opciones, pero destacamos los siguientes clientes por su implantación en el sector:

- **gvSIG**: Este cliente de escritorio mutiplataforma es capaz de consultar y comunicarse con servicios WMS, WFS, WCS y de catálogo. Por ello es una herramienta muy interesante en infraestructuras GIS que cumplan las especificaciones OGC. Este cliente está desarrollado en J2EE y se distribuye bajo licencia GNU GPL.
- **uDIG**: Este cliente de escritorio es capaz de consultar y comunicarse con servicios WMS y WFS no transaccionales. Este cliente está desarrollado en J2EE de código abierto.

3.5 Ejemplos de implementaciones WFS y WMS

Desde la aparición de los estándares publicados por OGC han aparecido múltiples servicios WFS y WMS que ofrecen cartografía e información de distintas zonas y con múltiples objetivos. Los servicios geográficos de muchas administraciones públicas han realizado grandes esfuerzos para adaptarse a las especificaciones definidas por los estándares e implementar este tipo de servicio para proveer información geográfica a los ciudadanos y a las empresas que lo requieran. En concreto, en España existen muchos ejemplos de utilización de servicios web WFS y WMS de la administración central y autonómica para proveer múltiples informaciones de carácter geográfico.

A continuación presentamos algunos de los ejemplos más significativos y relevantes de WMS y WFS creados por entidades públicas en España.:

3.5.1 Dirección general de catastro

La Dirección General de Catastro, siguiendo las múltiples iniciativas que han surgido en el seno de la Unión Europea, dispone desde hace algunos años de un servicio web WMS que provee los mapas de catastro de toda España (excepto Navarra y País Vasco). Dicho servicio cumple la especificación completa definida por OGC (3 operaciones) y puede ofrecer cartografía básica y catastral a grandes escalas desde 1:500 a 1:5.000 [47][48][49][50].

A mayores la Dirección General de Catastro ha desarrollado en los últimos años un servicio WFS [48] que retorna cartografía vectorial de toda España (excepto Navarra y País Vasco). Este servicio está construido con MapServer. El servicio WFS de Catastro [47] es no transaccional, y por lo tanto sólo nos dejará consultar datos geográficos y catastrales, sin posibilidad de actualización.

Según los datos de la Dirección General de Catastro, en su base de datos geográfica se almacenan datos de 41,7 millones de parcelas rústicas y 12,5 millones de parcelas urbanas.

Gracias al cumplimiento de las especificaciones de los estándares WMS y WFS, la información ofertada por los servicios de Catastro es consultada y a su vez ofertada por distintos portales de algunas comunidades autonómicas.

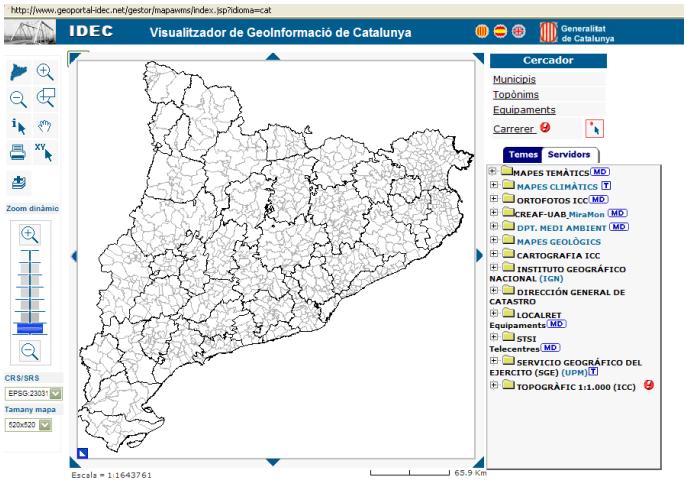

Figura 3.9 Portal de información geográfica de Cataluña

Como curiosidad destacamos que en la actualidad la Dirección General de Catastro dispone de una utilidad que podemos descargar de su página web que permite consultar información catastral de sus servicios web para representarla de forma integrada dentro de la aplicación Google Earth.

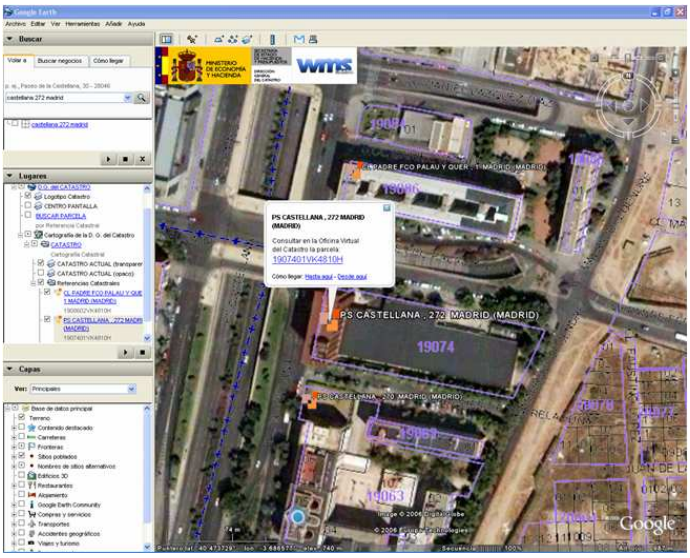

Figura 3.10 Integración de datos de catastro con Google Earth

3.5.2 Sistema de información territorial de Galicia (SITGA)

La sociedad para el desarrollo comarcal de Galicia ha implantado sendos servicios de WMS y WFS que proveen el portal de consulta de datos territoriales denominado SITGA. Dicho proyecto se enmarca dentro del "Programa de la Sociedad de la Información" de I+D+IT de la XUNTA de Galicia [46].

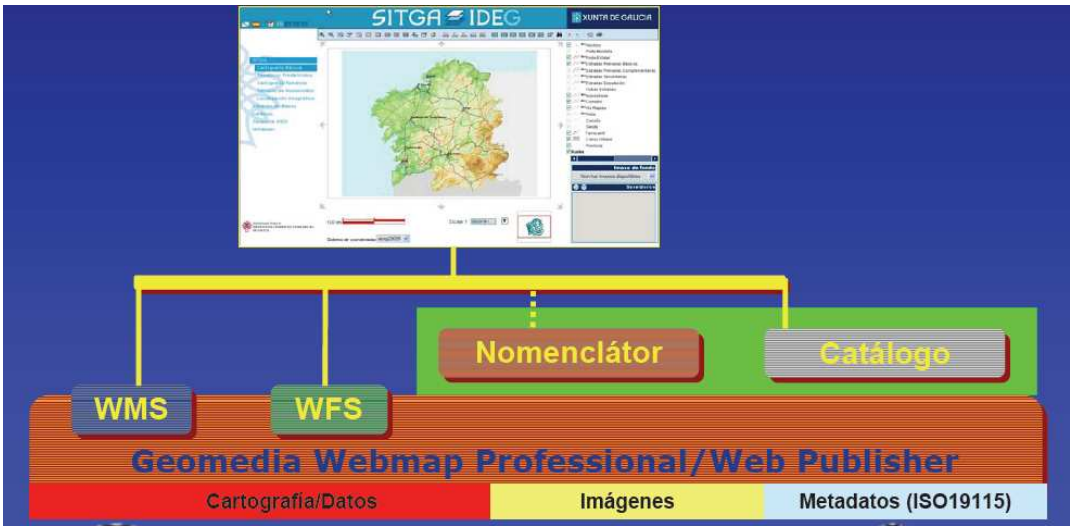

Figura 3.11 Esquema del portal de información territorial de Galicia

Dicho portal ofrece cartografía de Galicia con posibilidad de descargar distintas capas temáticas en función del objetivo de la consulta realizada:

- Consultas de límites administrativos.
- Consultas de carácter hidrogeológico.
- Consultas de carácter Litológico.
- Consultas de carácter Hidrográfico.
- Consultas turísticas (rutas del Camino de Santiago, recursos turísticos, etc.).

3.5.3 Otros ejemplos

Como hemos comentado al inicio de la sección, en la actualidad la práctica totalidad de comunidades autonómicas cuentan con portales que permiten consultar información de carácter geográfico bien accediendo a servicios WMS y WFS propios o consultando los servicios de otras administraciones.

En la actualidad podemos encontrar algunos ayuntamientos que ofrecen información geográfica en sus portales Web. Entre ellos destacamos los GeoPortales de los Ayuntamientos de Zaragoza y de Barcelona.

4 API de Google Maps

4.1 Introducción

Google Maps (http://maps.google.com) es un servicio gratuito (ver el apartado *4.2 Licencia de uso*) de la empresa Google Inc. que ofrece imágenes de mapas continuos (mapa desplazable), así como composiciones de fotografías tomadas por satélite que abarcan todo el globo terráqueo. Google Maps es el equivalente web a la aplicación multiplataforma Google Earth, desarrollada por la misma compañía. Google Maps proporciona una API abierta que permite personalizar cualquiera de los mapas para presentar en cualquier aplicación web, pudiendo incluso añadir datos específicos de la aplicación sobre dicho mapa. Una API (Interfaz de Programación de Aplicaciones) es un conjunto de especificaciones de comunicación entre componentes de software. Estas especificaciones establecerán las reglas necesarias para que se puedan realizar llamadas a los distintos servicios ofrecidos por el componente software en cuestión.

Con Google Maps se han roto muchas de las barreras que hasta ahora era necesario superar para entrar en el mundo de la presentación de información geográfica en entorno web. Gracias a esto, es relativamente sencillo incorporar mapas personalizados a cualquier aplicación, de tal forma que los usuarios pueden interactuar con dicho mapa utilizando simplemente el ratón de su ordenador y su navegador favorito.

Con Google Maps se puede ampliar un mapa desde una resolución de 10.000 Km por píxel hasta una resolución de 20 metros (excepcionalmente podemos encontrar zonas con resoluciones de 2,5 cm por pixel, como por ejemplo en la localización sede central de Google en California http://maps.google.com.au/maps?f=q&hl=es&q=1600+Amphitheatre+Parkway,+Mountain+View,+94043&sll=-33.860104,151.209632&sspn=0.000158,0.000338&layer=&ie=UTF8&om=1&z=22&ll=37.422004,-122.084069&spn=0.000939,0.002704&t=k&iwloc=addr, o incluso de 2 cm por píxel, como por ejemplo en una zona de Chad en el centro de África en la que podemos distinguir hasta el rostro de las personas http://maps.google.com/maps?f=q&hl=en&q=15.298683+19.429651&layer=&ie=UTF8&om=1&z=23&ll=15.298518,19.429731&spn=0.000094,0.000169&t=k&iwloc=addr), desplazarlo, calcular la distancia entre dos puntos de interés, marcar una ruta entre un origen y un destino, calcular el tiempo de desplazamiento para dicha ruta, localizar lugares de interés público como por ejemplo restaurantes o gasolineras, superponer imágenes de satélite sobre el mapa, etc. Las aplicaciones de este servicio son aún mayores si se enlazan los mapas con los datos procedentes de una aplicación empresarial: gestión de recursos humanos, localización instantánea de clientes, seguimiento de mercancías para gestionar cadenas de suministros, obtención de vistas con información de ventas y concentración de clientes para mejorar el entendimiento sobre el negocio, etc.

El servicio Google Maps fue anunciado oficialmente en febrero de 2005 y desde entonces no ha parado de crecer en cuanto a funcionalidades. En la actualidad este servicio está en fase Beta (aplicación en evolución permanente) a pesar de que su comportamiento es muy estable y ampliamente utilizado. Otros servicios que se nutren de Google Maps y que también están en fase Beta son: Google RideFinder (para localizar transportes públicos como taxis y limusinas), Google Local (para localizar negocios restringidos a una determinada área geográfica), Google Moon (versión de Google Maps pero dedicada a la Luna), Google Mars (equivalente a Google Maps pero específica del planeta Marte) y las ampliaciones de Google Maps para visualizar el tráfico o fotos a pie de calle (de momento sólo disponible en algunas ciudades de Estados Unidos).

4.2 Licencia de uso del servicio y de la API

La licencia actual de uso de la API de Google Maps establece una serie de condiciones que deben ser respetadas para la utilización del servicio. A continuación presentamos un resumen de las principales condiciones [51]:

- La API de Google Maps fue creada y liberada para que los desarrolladores pudieran incorporar a sus aplicaciones y sitios Web las presentaciones de Google Maps. Bajo ningún concepto se pueden utilizar para obtener los datos subyacentes del servicio (mapas, posiciones, etc.). Es decir, el código fuente y los datos que ofrece el servicio está protegidos por un copyright.

- No se pueden alterar la presentación de las imágenes fotográficas del servicio ni copiar para su posterior distribución.
- Google se reserva el derecho de modificar la versión de la API, lo que puede ocasionar que se tenga que revisar y actualizar el código desarrollado sobre dicha API.
- Google se descarga de cualquier responsabilidad de la finalidad para la que se utilice su servicio, siendo responsable la persona o personas que lo utilizan para fines ilegales (leyes de exportación de Estados Unidos y leyes del país de residencia de las personas que utilizan el servicio).
- En la actualidad el servicio de Google Maps no incluye publicidad en los mapas. De todas formas Google se reserva el derecho de introducir publicidad en el futuro (se compromete a anunciarlo con 90 días de antelación).
- Sólo se pueden realizar 50.000 peticiones de Geocode (consulta de longitud-latitud) por día y clave. En caso de exceder este límite el servicio de Google Maps denegará las peticiones de dicha clave de forma temporal. En caso de exceder de forma reiterada de dicho límite se bloqueará el servicio de forma permanente.
- El servicio de Google Maps no se puede utilizar para usos comerciales. Esto quiere decir que no se puede utilizar el servicio de Google Maps para desarrollar otros servicios comerciales, pero sí que lo podemos usar para presentar datos personales o internos.

4.3 Conceptos previos

Uno de los mayores exponentes de lo que hoy en día se hace llamar “Web 2.0” [64] es el servicio Google Maps. Este servicio hace un uso muy intensivo de la tecnología AJAX (Asynchronous JavaScript And XML) [59], cuyas aplicaciones se caracterizan por ejecutarse en un navegador web y por mantener una comunicación asíncrona con el servidor en segundo plano. De esta forma, se pueden realizar cambios dinámicos en la página web que se está visitando, sin necesidad de recargarla de nuevo. En el caso de Google Maps, se envían peticiones asíncronas desde el navegador al servidor para solicitar las distintas imágenes que componen el mapa visualizado. Cuando se amplía o reduce el mapa o cuando éste se desplaza, el navegador solicita las nuevas imágenes en segundo plano al servidor y éste las muestra al usuario según vayan recibiéndose. Pero toda esta funcionalidad está gestionada de forma automática por la API de Google Maps y por el código del servidor de este servicio, por lo que no habrá que preocuparse por ello.

La API de Google Maps está implementada en JavaScript [60], que es un lenguaje interpretado utilizable en páginas web y con una sintaxis muy parecida al lenguaje Java. Por lo tanto, para incluir la funcionalidad de Google Maps en una página web, es necesario realizar un trabajo de programación en este lenguaje. Uno de los problemas más importantes a los que se enfrentan este tipo de aplicaciones es el de las incompatibilidades existentes entre los navegadores web. Esto significa que para que una aplicación web soporte los distintos navegadores es necesario tener en cuenta durante su implementación las diferencias que existen entre éstos, pues mientras que algunos trozos de código funcionarán perfectamente en unos navegadores, a lo mejor en otros no funcionarán o lo harán de forma incorrecta. A pesar de todo esto, la API de Google Maps es totalmente compatible con los navegadores más actuales: Internet Explorer 6.0+, Firefox 0.8+, Safari 1.2.4+, Netscape 7.1+, Mozilla 1.4+ y Opera 8.02+. Pero esta compatibilidad de la API, no es suficiente para garantizar que el código de la aplicación web que use Google Maps será también compatible. Para ello, será necesario probar dicho código cliente de forma exhaustiva en los distintos navegadores. Esto puede ser minimizado en parte con la ayuda de librerías de programación a medida.

Como consecuencia del uso de la filosofía AJAX, también se utilizan las tecnologías XHTML (o HTML) para representar la información, CSS (Hojas de estilo en cascada) para modelar el aspecto visual de la información, DOM (Document Object Model) para manipular la información mostrada, el objeto XMLHttpRequest (o en algunos casos un iframe) para intercambiar datos asíncronos con el servidor, y XML como formato de codificación de los datos en las comunicaciones cliente-servidor (aunque pueden emplearse otros formatos como JSON).

Cuando el navegador es Internet Explorer, Google Maps utiliza VML (Vector Markup Language) para dibujar líneas sobre el mapa visualizado, mientras que en el resto de navegadores se solicita una imagen de la línea desde los servidores de Google y luego la sitúa sobre el mapa, refrescando la imagen cada vez que el mapa se desplace o se amplíe. Estas líneas sirven para marcar rutas, delimitar zonas, etc. VML es una aplicación de XML, que define un formato de codificación de información vectorial, que en el caso de Google Maps son gráficos vectoriales.

4.4 Ventajas y desventajas de Google Maps

Como Google Maps está construido bajo la filosofía AJAX, la mayoría de sus ventajas e inconvenientes provienen de AJAX. En cuanto a las ventajas, se pueden destacar las siguientes:

- Google Maps es altamente interactivo, sobre todo porque no se producen recargas completas de la página, con lo que da la sensación de estar casi ante una aplicación de escritorio. Cualquier actividad realizada con el ratón tiene respuesta por parte de la aplicación; y esto es algo muy a considerar, teniendo en cuenta que se trata de una aplicación web, con todas las limitaciones que se esperan de éstas. La rapidez en la carga de las partes de los distintos mapas es realmente asombrosa, siempre y cuando la conexión tenga el suficiente ancho de banda. Realmente se trata de un interfaz muy sencillo e intuitivo de utilizar. Comparando la interfaz de Google Maps con la de otros servicios de mapas on-line (como por ejemplo Mapquest, Yahoo! Maps o Ask Maps), quizás puede ser considerado como el más sencillo, el más fluido y el que más posibilidades ofrece de cara al usuario (sin tener en cuenta la versión en Flash de Yahoo! Maps).
- El servicio es escalable, es decir, no hay ningún sistema GIS (sistema de información geográfica) en el servidor. Lo único que se envía a los clientes son imágenes (millones de ellas) y unos pocos archivos de código JavaScript. Para convertir el servicio realmente en un GIS, es necesario combinar dichas imágenes con datos procedentes, por ejemplo, de la aplicación que va a utilizar el servicio. Para realizar esto, Google Maps proporciona su API en forma de archivos JavaScript. Por el contrario, el servicio de mapas de la empresa Yahoo! puede emplear Flash en sustitución de JavaScript, pero se trata de una tecnología propietaria que requiere la instalación de un “plug-in” en el navegador de cada usuario. Como compensación a esto, hay que decir que las interfaces en Flash son mucho más ricas y cuentan con más posibilidades que las que puede ofrecer HTML y Javascript.
- La cartografía es de alta calidad; todo se sirve a los clientes preprocesado, con lo que el tiempo de renderizado sobre el navegador es mínimo.
- La cobertura del servicio es bastante amplia. Por ejemplo, en la comunidad autónoma de Galicia, los mapas de la costa, de las grandes ciudades, así como el de muchos núcleos de población tienen un alto grado de detalle (incluso hasta poder ver los vehículos circulando), lo que permite realizar aplicaciones web localizadas en esta comunidad. En cambio, en otros servicios de mapas, o no se puede alcanzar tanto detalle, o ni siquiera es posible obtener una fotografía satélite de ninguna zona de Galicia.
- Ofrece un API gratuito y público (aunque no abierto) para personalizar los distintos mapas y poder integrarlos en cualquier aplicación web. La API de Google Maps es la API más popular y más utilizada que la de otros servicios de mapas, quizás por haber sido una de las primeras en ver la luz.

Y entre las desventajas se pueden mencionar las siguientes:

- Se hace un uso muy intensivo del ancho de banda. Cada vez que el usuario se desplaza por un mapa, lo amplía o lo reduce, se están solicitando nuevas imágenes al servidor. Además, para que los desplazamientos del mapa sean suaves (necesario para mejorar la experiencia del usuario), es necesario solicitar al servidor más imágenes que las que realmente se ven. Como consecuencia de esto se requiere una conexión con el suficiente caudal de bajada como para que no se vea saturado en el momento de recibir las distintas imágenes que componen el mapa visualizado.
- Para mejorar la eficiencia del servicio, las imágenes que componen los mapas ya vienen preprocesadas desde el servidor. Esto dificulta cambiar datos de forma rápida en el propio cliente, así como el trabajo con capas. De hecho, la interfaz de Google Maps no da al usuario la capacidad para intercambiar capas; sólo puede mostrarlas u ocultarlas, pero no intercambiarlas. En Google Maps, el orden de las capas desde las más interna a la más externa es el siguiente:
    - El mapa, compuesto de múltiples imágenes.
    - La capa en la que se colocan las sombras de los iconos de los marcadores.
    - La capa de los marcadores o puntos de interés.
    - La capa en la que se colocan las sombras de las ventanas de información.
    - La capa con los elementos invisibles que sirven para responder a los eventos del ratón.
    - La capa que contiene la ventana de información de un marcador.

A pesar de que estas capas no son intercambiables, el número de casos en que sería necesario realizar un intercambio de este tipo sería mínimo, por lo que la desventaja no es tan importante.

- Se requiere un navegador de última generación compatible con el servicio con JavaScript habilitado. Esto sería importante si los usuarios que van a utilizar la aplicación que utiliza el servicio de mapas de Google contasen en sus equipos con una navegador relativamente antiguo, puesto que les obligaría (a los usuarios o al departamento de sistemas) a realizar una actualización del mismo o a su sustitución.
- La API de Google Maps está en fase Beta, lo que implica que puede contener errores y sufrir cambios en futuras versiones. Además, está involucrada en un continuo y rápido proceso de evolución que provoca constantes cambios de versiones, bien para corregir errores, o bien para añadir nuevas posibilidades. Pero mientras la siguiente versión de la API sea compatible, no habrá que preocuparse por los cambios de versiones, puesto que la API se actualiza automáticamente, por lo que la aplicación web que haga uso de ella siempre estará utilizando la versión más reciente. Sin embargo, si el cambio de versión afecta a la compatibilidad, habría que volver a reprogramar el código de la aplicación (código JavaScript) que hace uso de dicha API, aunque esta incompatibilidad sería anunciada de antemano por Google.
- Para poder personalizar un mapa de Google Maps, es necesario contar con profesionales que tengan conocimiento de las distintas tecnologías involucradas en este servicio (ver el apartado 4.6 titulado “La API de Google Maps”) como JavaScript, XHTML, DOM, etc.
- En la actualidad Google permite la utilización de su servicio de forma gratuita gracias a la liberación de su API. De todas formas los términos de su licencia establecen que esta “gratuidad” puede ser modificada en futuras versiones (introducción de publicidad o cuotas por utilización del servicio). Dentro de Open Source, nos encontramos con una alternativa a Google Maps conocida como proyecto OpenLayers [55], aunque de momento con un menor potencial y desarrollo.

Desde el punto de vista de un GIS, Google Maps está “orientado al punto”, es decir, que para obtener la información geográfica, es necesario hacer clic sobre uno de los marcadores del mapa. En la última versión de Google Maps se puede asociar información a distintos objetos como rutas. Probablemente en el futuro se irán incorporando nuevas funcionalidades al servicio, pues se trata de un servicio en plena evolución y que todavía es muy reciente. Sin embargo, Google Maps no pretende alcanzar a los sistemas GIS tradicionales, muy orientados a grandes corporaciones o a instituciones públicas, que puedan permitirse sus altos costes. Su gran objetivo es hacer que la información geográfica sea accesible de la forma más simple al público en general y permitiendo que los desarrolladores de software puedan construir sus propias aplicaciones basadas en mapas. Realizar una aplicación para mostrar datos espaciales utilizando Google Maps resulta un proceso más rápido y sencillo que utilizar por ejemplo una aplicación GIS específica como *ArcGIS* para construir una aplicación similar. De todas formas tenemos que dejar claro que las funcionalidades que se pueden obtener con un GIS tradicional son superiores (tanto en capacidad de procesado de volúmenes grandes de datos geográficos como en las funcionalidades que ofrecen) a lo que se puede conseguir con un SI que localice su información en mapas servidos por Google Maps.

En este sentido tenemos que decir que los pocos años de existencia de Google Maps, este ha facilitado que usuarios individuales puedan construir aplicaciones basadas en mapas sin necesidad de las complejas infraestructuras que requieren los GIS tradicionales. Este efecto ha sido acentuado por hecho de que, a día de hoy, el servicio es totalmente gratuito y de muy fácil utilización (tanto para los usuarios como para los desarrolladores), factor clave para entender el gran éxito que está teniendo esta tecnología.

4.5 Comparación de Google Maps y WMS

Aunque en un principio pueda parecer que el objetivo del servicio de Google Maps y de los WMS sea el mismo (servir mapas en formato digital), existen múltiples diferencias que nos llevan a afirmar que Google Maps no es un WMS y que sobre todo que ofrece muchas menos posibilidades que estos:

- Google Maps no cumple la especificación definida por la OGC para los WMS.
- Google Maps dispone de un API de programación no estándar y propietaria.
- Google Maps no permite la definición de estilos y capas propios.
- Google Maps no permite modificar el orden de presentación de sus capas, ni siquiera permite ocultarlas. Simplemente permite modificarlas ligeramente.

Por el contrario podemos decir que Google Maps ofrece una API de programación con la que podemos incorporar de forma sencilla elementos geográficos e información asociada a los mapas que nos retorna el servicio. Otra ventaja de Google Maps es la frecuencia con la que actualizan los mapas (y las imágenes por satélite), lo que lo convierte en un servicio muy adecuado para la presentación de información georrefenciada sobre mapas actualizados.

Si analizamos los objetivos de ambos servicios observamos que, aunque comparten el objetivo principal (servicios que proveen mapas), se diferencian en los objetivos finales que persiguen:

- El objetivo de Google es el de ofrecer un servicio comercial que mediante la publicación de mapas le pueda dar ingresos publicitarios.
- El OGC tiene como objetivo establecer un estándar que permita el intercambio de información entre sistemas. En concreto en el caso de los servicios WMS este intercambio de información se realiza en formato raster o en formato XML con el que se transmite información concreta de los objetos que se presentan.

4.6 La API de Google Maps

Actualmente, la API de Google Maps se encuentra en fase Beta (en concreto Beta 2), esto significa que está sujeta a cambios, y por lo tanto que debemos tener ciertas precauciones sobre los posibles cambios que pueda sufrir dicha API en el futuro [52][53][54].

Para poder hacer uso de la API es necesario obtener una clave de los servidores de Google. Cada una de estas claves estará asociada a una URL de un sitio web. Según esto, hay que solicitar una clave para cada uno de los directorios de la aplicación que sirve mapas de Google. Por ejemplo, si se pretende servir mapas desde las direcciones http://www.servicioMapas.org/wwwroot y http://www.servicioMapas.org/wwwroot/mapas, hay que obtener una clave para cada una de estas direcciones. Además, para solicitar la clave es necesario contar con una cuenta de tipo “Google Account”.

El ejemplo más sencillo posible para mostrar un mapa de Google Maps en una aplicación web es el siguiente:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8"/>
    <title>Ejemplo de mapa de Google Maps</title>
    <script src="http://maps.google.com/maps?file=api&amp;v=2&amp;key=12345678"
            type="text/javascript"></script>
    <script type="text/javascript">
    //<![CDATA[

    function load() {
      if (GBrowserIsCompatible()) {
        var mapa = new GMap2(document.getElementById("mapa"));
        mapa.setCenter(new GLatLng(42.87596410, -8.55834961), 13);
      }
    }

    //]]>
    </script>
  </head>
  <body onload="load()" onunload="GUnload()">
    <div id="mapa" style="width: 500px; height: 300px"></div>
  </body>
</html>
```

Obsérvese cómo hay que hacer referencia a la API de Google Maps utilizando la clave correspondiente (`key=12345678`):

```
<script src="http://maps.google.com/maps?file=api&amp;v=2&amp;key=12345678"
        type="text/javascript"></script>
```

La API es un conjunto de archivos implementados en JavaScript, con lo que es necesario tener un conocimiento de este lenguaje para poder hacer uso de ella en una aplicación web.

A continuación se presentarán los distintos conceptos clave del API de Google Maps:

- **Los mapas:** Para colocar un mapa en una página de una aplicación web, es necesario un hueco donde ubicarlo. Para marcar dicho lugar dentro de la página se puede utilizar el elemento `div` de HTML. El tamaño del mapa vendrá determinado por el tamaño de su contenedor, que en este caso será el elemento `div`. Para que se cargue un mapa en dicho contenedor, es necesario instanciar un objeto de tipo `GMAP2`. Normalmente esta instancia se realiza atrapando el evento `onload` de la página, tal y como se muestra en el ejemplo anterior:

```
var mapa = new GMap2(document.getElementById("mapa"));
```

  Para realizar la instancia es obligatorio especificar el elemento que deberá contener el mapa, utilizando para ello DOM (Document Object Model). Una vez instanciado el mapa, se tendrá acceso a un conjunto de métodos para controlar la configuración del mapa, su estado, los controles que mostrará, los *overlays*, las ventanas de información y los eventos asociados al mapa. En el ejemplo de antes, se utiliza el método `setCenter()` para centrar el mapa en un punto geográfico especificado a partir de sus coordenadas y con un determinado nivel de "zoom".

- **Los puntos geográficos:** Para especificar un punto geográfico en el mapa, se puede hacer de dos formas distintas: a través de sus coordenadas geográficas o a través de sus coordenadas gráficas (píxeles). Para crear un punto a través de las coordenadas geográficas se utiliza la clase `GLatLng`:

```
var punto = new GLatLng(42.87596410, -8.55834961);
```

  Para crear el punto a partir de las coordenadas gráficas se emplea la clase `GPoint`:

```
var punto = new GPoint(6, 20);
```

- **Los marcadores:** Los marcadores se utilizan para marcar un punto de interés del mapa. Normalmente llevan asociada cierta información, que se mostrará al usuario en una pequeña ventana (infoWindow) cuando éste haga clic sobre dicho marcador. Para crear un marcador se utiliza la clase `GMarker`, proporcionando un punto a partir de sus coordenadas geográficas, es decir, de tipo `GLatLng`:

```
var mapa = new GMap2(document.getElementById("mapa"));
var marcador = new GMarker(new GLatLng(42.87596410, -8.55834961));
```

  Con el método `addOverlay` de la clase `GMap2`, se añade el marcador al mapa para su visualización. Por defecto, la API muestra el marcador con un icono por defecto, que se puede cambiar fácilmente haciendo uso de la clase `GIcon`. También existe la posibilidad de crear marcadores móviles (el usuario los puede arrastrar con el ratón), estableciendo la propiedad `draggable` del marcador a `true`.

- **Las ventanas de información:** Las ventanas de información muestran información sobre puntos geográficos, normalmente representados con marcadores. El contenido de una ventana de información puede ser cualquier representación en código HTML, teniendo siempre presente que el tamaño de estas ventanas suele ser muy reducido. Para minimizar este problema se pueden utilizar las ventanas de información con solapas:

Figura 4.1 Ventana de información estándar

Figura 4.2 Ventana de información con solapas

Para abrir una de estas ventanas se utiliza el método `openInfoWindowHtml()` para una ventana normal o `openInfoWindowTabsHtml()` para una ventana con solapas:

```
var mapa = new GMap2(document.getElementById("mapa"));
var marcador = new GMarker(new GLatLng(42.87596410, -8.55834961));
mapa.addOverlay(marcador);
var html = “Información para el marcador <br> número 1</br>.”;
```

Para mostrar una ventana con solapas hay que crear un vector de objetos `GInfoWindowTab` que contienen el código HTML y el título de la solapa.

```
var mapa = new GMap2(document.getElementById("mapa"));
var marcador = new GMarker(new GLatLng(42.87596410, -8.55834961));
mapa.addOverlay(marcador);
var solapas = [
      new GInfoWindowTab(“Solapa 1”, “Marcador <br> número 1</br>.”),
      new GInfoWindowTab(“Solapa 2”, “Marcador <br> numero 1</br>.”)
];
marcador.openInfoWindowTabsHtml(solapas);
```

Para que la ventana de información se abra justo en el momento en el que el usuario hace clic sobre el marcador, hay que asociar el evento “clic” del marcador con la apertura de la ventana. Para ello se hace uso del sistema de eventos de la API:

```
GEvent.addListener(marcador, “click”, function() {
      marcador.openInfoWindow(html);
});
```

En este ejemplo, se está asociando una función anónima al evento de tipo “click” del marcador, con lo que el contenido de la función se ejecutará al hacer clic sobre el marcador.

- **Las polilíneas:** Las polilíneas son un conjunto de líneas o segmentos concatenados unos con otros de forma ordenada. En la siguiente imagen se puede observar una polilínea que representa un área de un mapa:

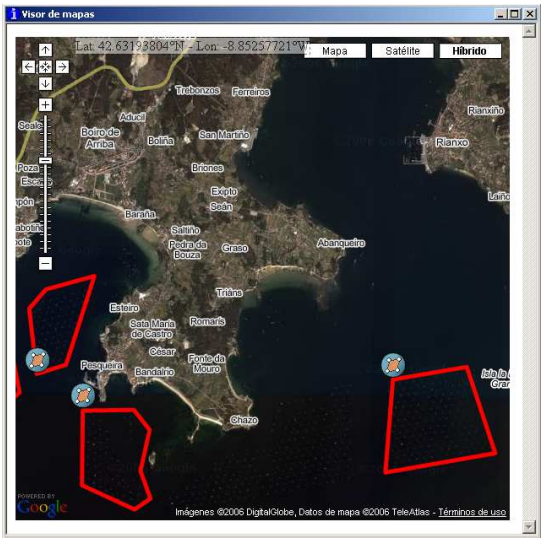

Figura 4.3 Ejemplo de polilíneas

Para crear una polilínea es necesario instanciar la clase `GPolyline` proporcionando al menos dos puntos geográficos utilizando la clase `GLatLng`, tal y como se muestra a continuación:

```
var puntos = [
      new GLatLng(45.36512573, -8.39355469),
      new GLatLng(44.49278281, -8.21777344),
      new GLatLng(41.01268089, -7.56958008),
];
var polilinea = new GPolyline(puntos);
mapa.addOverlay(polilinea);
```

También es posible especificar el color de la polilínea, su ancho o su nivel de transparencia. Para que las poli líneas puedan verse en el navegador Internet Explorer, hay que incluir en la página una referencia al espacio de nombres VML y un pequeño trozo de código CSS.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
 xmlns:v="urn:schemas-microsoft-com:vml">
  <head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8"/>
    <script src="http://maps.google.com/maps?file=api&amp;v=2&amp;key=12345678"
            type="text/javascript"></script>
    <style type="text/css">
    v\:* {
      behavior:url(#default#VML);
    }
    </style>
  </head>
```

Hay que tener mucho cuidado con las polilíneas, pues para representar una que sea larga y complicada (número elevado de puntos) se consume mucha memoria e incrementa de forma notable el tiempo para pintarla sobre el mapa. Además, las polilíneas se representan siempre en el mapa, independientemente de su nivel de “zoom”. Para minimizar este problema, se han definido las polilíneas codificadas. Con este tipo de polilíneas es posible especificar grupos de niveles de “zoom” en los que debe ser pintada, con lo que el pintado del mapa se vuelve un poco más eficiente. Pero la gran desventaja de este tipo de polilínea es que para indicar los puntos que la componen, hay que utilizar un algoritmo de codificación (de ahí su nombre), para obtener un formato

comprimido de caracteres ASCII que representan los distintos puntos. Desgraciadamente, este algoritmo no está incluido en la API de Google Maps, aunque sí está documentado con detalle en la documentación oficial de la API; con lo que es posible implementarlo para uso propio en cualquier aplicación web.

- **Los polígonos:** La idea de los polígonos es la de representar regiones en los mapas. Por el momento, esta capacidad no está ofrecida por la API, pero hay buenas expectativas de que en una futura versión se ofrezca esta capacidad de representación. No obstante, existen APIs de terceros que ofrecen esta posibilidad como XMAPS (http://xmaps.busmonster.com). También es posible simular los polígonos en cierta manera a través de las polilíneas, haciendo que el punto de origen sea igual al punto de destino. La desventaja de este método es que no es posible aplicar un color de relleno al polígono resultante.
- **Los controles:** Los controles sirven para que el usuario pueda manipular a su gusto ciertos aspectos del mapa visualizado, como por ejemplo, desplazarlo o ampliarlo. La API permite añadir hasta 4 tipos de controles diferentes en un mapa: controles de movimiento y "zoom", control de tipo de mapa, control de mapa general, y control de la escala del mapa. Para añadir un control al mapa se utiliza el método `addControl()` de la clase `GMap2`:
  - **Controles de movimiento y "zoom":** Este control permite al usuario desplazar el mapa, ampliarlo o reducirlo. A su vez, puede ser de tres tipos: `GLargeMapControl`, `GSmallMapControl` y `GSmallZoomControl`.

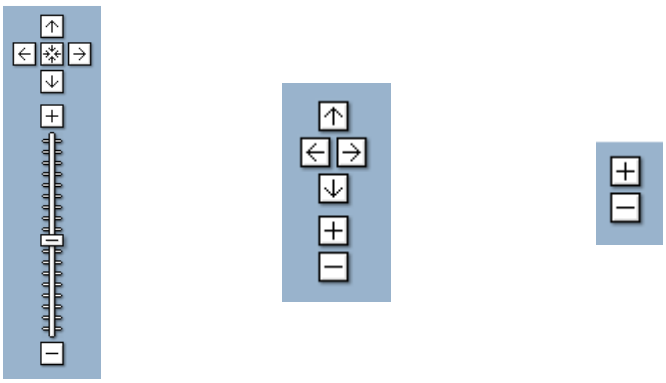

Figura 4.4 Controles de movimiento: GLargeMapControl, GSmallMapControl, GSmallZoomControl

  - **Control de tipo de mapa:** Sirve para cambiar el tipo de vista del mapa: vista de mapa, vista de foto satélite o vista híbrida. Por defecto, siempre se muestra este control. Para personalizarlo, hay que utilizar el argumento `options` del constructor de la clase `GMap2`.

```
var opciones=new Object();
opciones.mapTypes = [G_HYBRID_MAP, G_SATELLITE_MAP];
var mapa = new GMap2(document.getElementById("mapa"), opciones);
```

  - **Control de mapa general o "minimapa":** Permite adjuntar al mapa principal un pequeño mapa más general (con menos "zoom") que muestra un pequeño recuadro de la zona correspondiente al mapa principal. Además, permite desplazar el mapa principal con un mayor grado de precisión.

```
mapa.addControl(new GOverviewMapControl());
```

  - **Control de escala del mapa:** Sirve para mostrar al usuario la escala del mapa que se está visualizando.

```
map.addControl(new GScaleControl());
```

- **Los eventos:** Todo el trabajo con eventos debe realizarse a través de la clase `GEvent`.La API define una serie de tipos de eventos que es posible asociar para realizar una determinada acción, entre los que se encuentra, por ejemplo, "click". La lista completa se puede consultar en la documentación oficial de la API de Google Maps en http://www.google.com/apis/maps/documentation. Cada tipo de componente de un mapa (marcador, ventana de información, etc.) ofrece una serie de tipos de eventos a los que es posible asociar una acción. También es posible utilizar esta clase para definir eventos personalizados de componentes que no forman parte de la API, o incluso eventos de componentes DOM correspondientes al documento XTML actual.
- **Geocodificación:** La geocodificación es el proceso que permite obtener las coordenadas geográficas (latitud y longitud) de una dirección geográfica (por ejemplo la dirección de un cliente). La API ofrece la clase `GClientGeocoder` para resolver este tipo de peticiones. Para ello hace uso de la tecnología AJAX. En el siguiente ejemplo, se añade un marcador al mapa cada vez que se pregunte por una dirección, desplegando su ventana de información asociada:

```
var mapa = new GMap2(document.getElementById("mapa"));
var geocoder = new GClientGeocoder();

function mostrarDireccion(direccion) {
  geocoder.getLatLng(
    direccion,
    function(punto) {
      if (!punto) {
        alert("No se encuentra la dirección '" + direccion + "'");
      } else {
        mapa.setCenter(punto, 13);
        var marcador = new GMarker(punto);
        mapa.addOverlay(marcador);
        marker.openInfoWindowHtml(direccion);
      }
    }
  );
}
```

Actualmente, el servicio de geocodificación de Google Maps no es uno de sus puntos fuertes, sobre todo en España, donde no se pueden localizar ciertas capitales de provincia. Lógicamente el servicio es nuevo y se encuentra en fase beta, con lo que no debería sorprender. Si realmente se necesitase utilizar la geocodificación, lo mejor en la actualidad es utilizar algún servicio comercial.

Una de las últimas funcionalidades que han aparecido dentro de la API de Google Maps es la posibilidad de asociar etiquetas de geolocalización a los feeds de los servicios RSS. Estos es gracia a la adopción del estándar GeoRSS que establece un mecanismo para insertar datos geográficos en los feeds RSS.

Podemos obtener información mucho más detallada acerca del uso de la API de Google Maps en el sitio web oficial de Google Maps: http://www.google.com/apis/maps/documentation.

4.7 Ejemplos de aplicaciones integradas con Google Maps

Después de dos años de que apareciera el servicio de Google Maps nos podemos encontrar con numerosas explotaciones de dicho servicio. El servicio se ha integrado tanto en aplicaciones de uso comercial como en aplicaciones y páginas web de presentación de contenidos. A continuación presentamos algunos de los ejemplos que hemos encontrado durante la elaboración de este proyecto [58].

4.7.1 Usos comerciales

La facilidad de empleo de Google Maps y las numerosas funcionalidades que se están incorporando al proyecto hace que muchas empresas se decanten por usar sus servicios para incorporar presentaciones

georreferenciadas a sus aplicaciones. En la mayor parte de los casos el empleo de los mapas de Google son complementos a las funcionalidades que ya ofrecían sus aplicaciones, pero en ningún caso se puede considerar la funcionalidad principal de las mismas, y por eso en los casos que a continuación enumeramos no se han utilizado sistemas GIS profesionales.

4.7.1.1 Muchoviaje.com

Esta Web especializada en la comercialización de viajes y paquetes vacacionales incorpora a sus servicios la posibilidad de consulta de espectáculos en distintas ciudades y la localización en los mapas de Google.

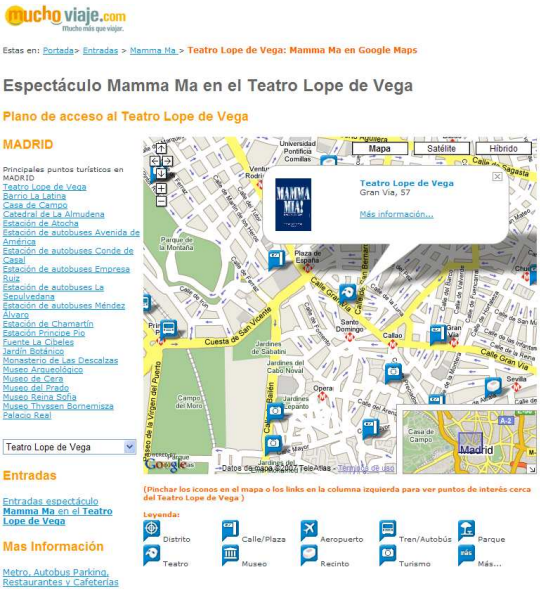

Figura 4.5 Ejemplo de uso comercial de los servicios de Google Maps

4.7.1.2 Sistema de información de asociaciones de productores de mejillón

La empresa Nática Sistemas de Información [56] desarrolló en el año 2002 un sistema de información especializado en la gestión de una asociación de mejillón. Dicho sistema gestiona además de los tipos procesos de gestión (pedidos, albaranes, facturación, contabilización, etc.), las órdenes de extracción de mejillón entre las distintas asociaciones que la conforman. La gestión de las órdenes de extracción implementa un sistema de sistema de reparto que controla los distintas incidencias medioambientales y sanitarias (cierres por toxinas) que afectan a las distintas zonas y rías en las que se ubican las delegaciones y sus bateas, intentando en todo momento compensar las ventas entre todos los productores de mejillón en función de sus posibilidades de producción y de los periodos que estén afectados por cierres sanitarios.

Este innovador sistema pretende cubrir un vacío en un sector con una escasa implantación tecnológica en sus procesos productivos y de gestión.

Dicho sistema de gestión dispone de un pequeño cuadro de mando que permite consultar y analizar distintos indicadores de producción que ha definido la asociación para medir su producción y la de sus asociados.

En el año 2006 se ha incorporado a dicho sistema la capacidad de representar la información gestionada por la aplicación sobre mapas suministrados por el servicio de Google Maps.

Dado que todo el sistema de gestión está construido bajo una arquitectura cliente-servidor, con clientes pesados, fue necesario generar una especificación propia de un fichero de intercambio en XML que permitiera la interoperatibilidad con el servicio de Google. La presentación de los datos se realiza mediante la utilización dentro del propio sistema de la API de Google, a través de JavaScript.

A continuación presentamos algunos ejemplos de representaciones de datos generadas por el sistema:

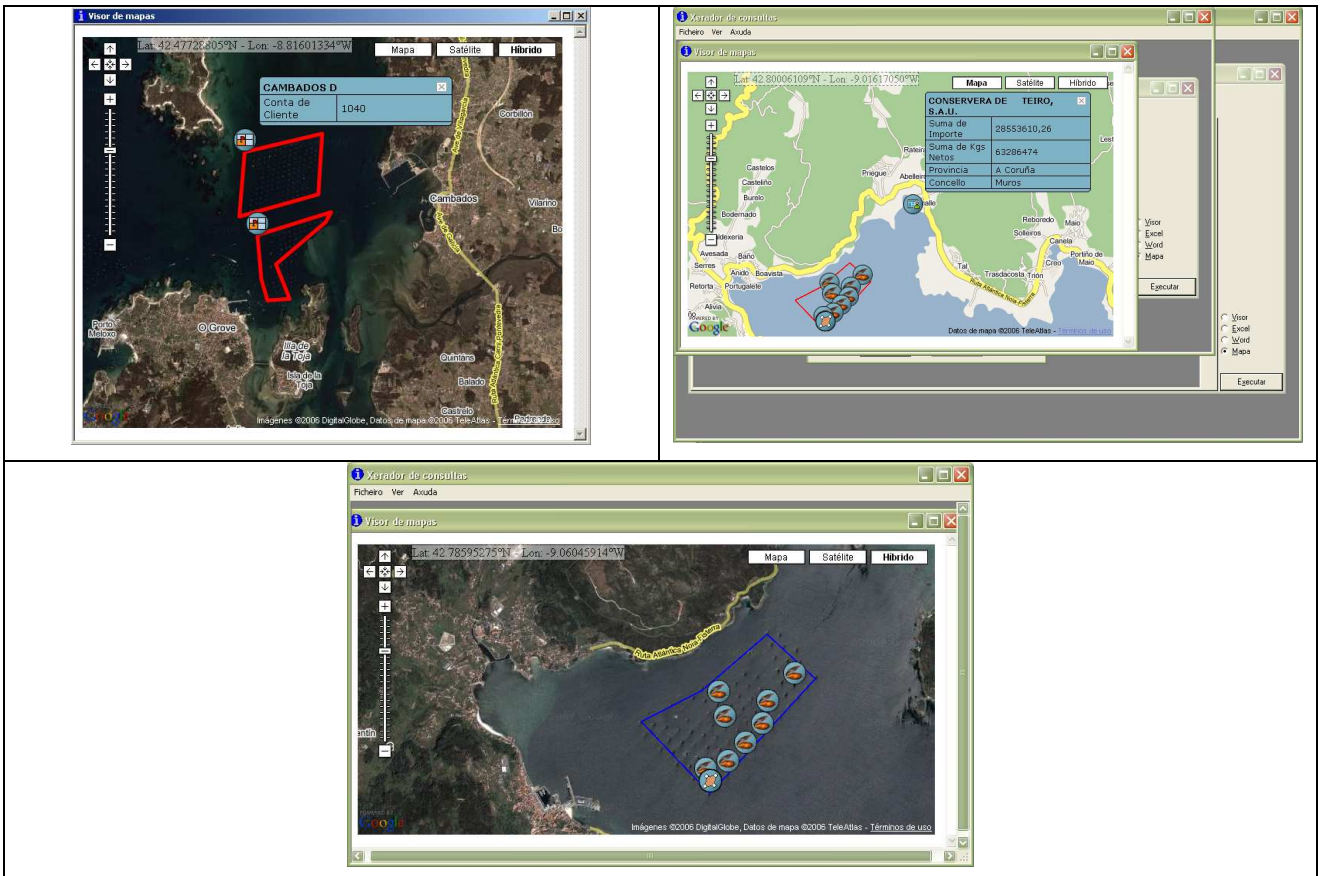

Figura 4.6 Ejemplos de presentaciones de datos con Google Maps

En la figura 4.7 podemos observar la secuencia de llamadas entre el módulo en el que el usuario realiza las petición de información (configura la consulta de información) y la pantalla que tiene incrustado un objeto Internet Explorer en el que se presentan los mapas suministrados por Google y los datos incorporados por el sistema.

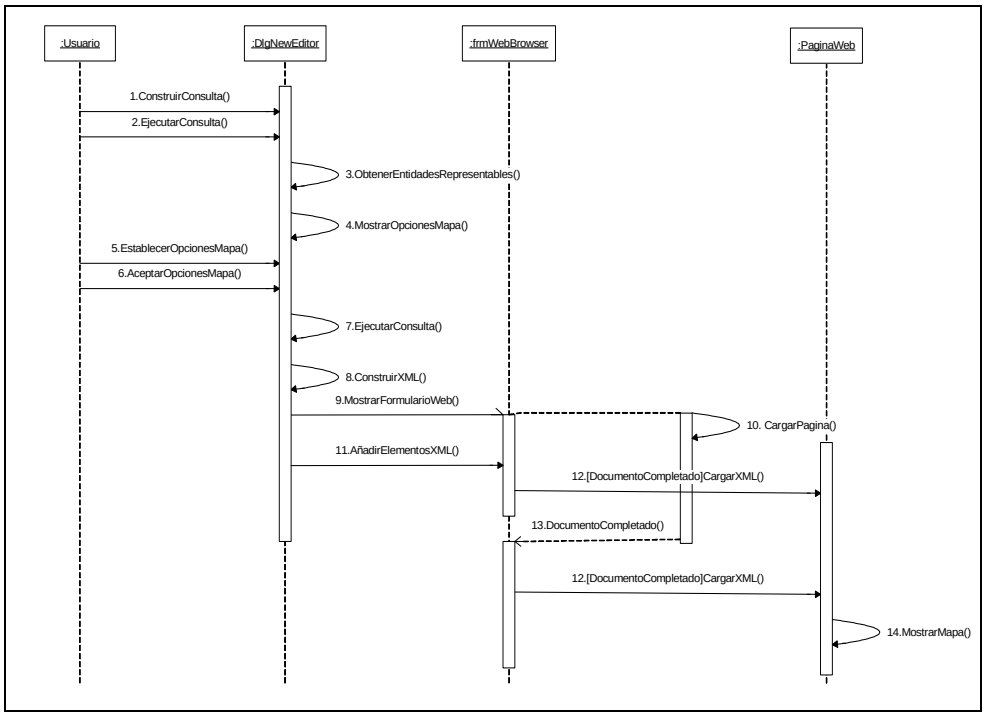

Figura 4.7 Diagrama de secuencia de la integración del SI de asociaciones de mejillón con Google Maps

4.7.1.3 Servicio de trenes del Reino Unido

El servicio de trenes del Reino Unido [65] nos permite consultar sobre un mapa la posición de cada uno de los trenes de este servicio. Además, sobre el propio mapa se puede observar cómo se van actualizando las posiciones de los trenes con una frecuencia de unos segundos.

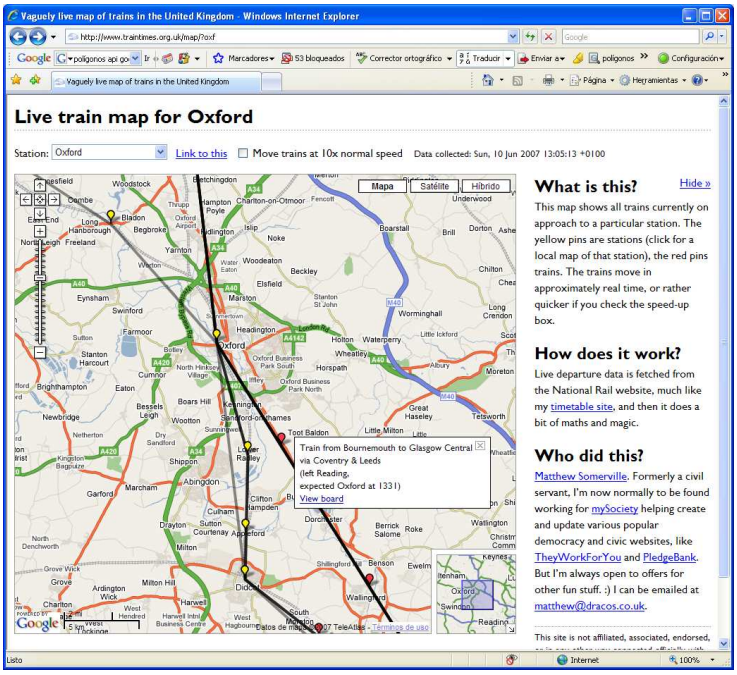

Figura 4.8 Servicio de trenes del Reino Unido

4.7.2 Usos no comerciales

A lo largo de toda la red podemos encontrar multitud de casos de web no comerciales que han utilizado el servicio de Google Maps para la presentación de información georreferenciada. Dentro de la información que se georreferencia, encontramos desde casos de lo más curiosos y extraños (como por ejemplo localización de famosos en Los Ángeles o localización de avistamientos de tiburones en las costa de Australia) hasta ejemplos de ingenio y georreferenciación de información útil para distintos colectivos (como por ejemplo localización de redes WIFI libres, localización de seísmos producidos a lo largo de todo el planeta, etc.).

Dentro de las múltiples web que permiten la presentación de datos geográficos gracias a la utilización de Google Maps destacamos la web Wikiloc [57]. Esta Web permite la publicación y localización de distintos tipos de rutas de ocio (bicicleta, senderismo, carreras, etc.) a lo largo de todo el planeta y su visualización en los mapas de Google.

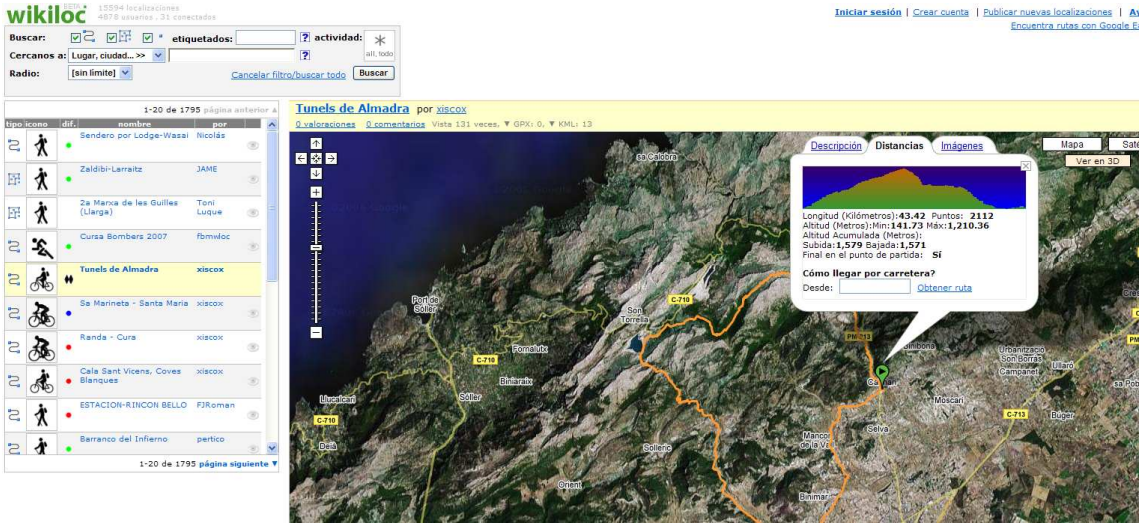

Figura 4.9 Ejemplo de uso no comercial de Google Maps

5 Diseño del GIS

En el presente apartado mostraremos las distintas alternativas que tenemos para la construcción del sistema de información de localización de elementos de seguridad. Por otro lado identificaremos las entidades que tienen que representar el sistema y para ello realizaremos un diseño de la información con la que tiene que trabajar la aplicación desde el punto de vista de la representación de los datos georreferenciados.

5.1 Requisitos de información geográfica

Desde el punto de vista de la localización de la información de forma georreferenciada el sistema tiene que cumplir con los siguientes requisitos:

- **Permitir la representación de elementos fijos con su información asociada**: Para el caso que nos ocupa, el sistema tiene que permitir localizar en un mapa las videocámaras de la empresa de seguridad. Entre la información que tiene que facilitar de cada elemento destacamos:
    - Código y descripción del elemento
    - Posición del elemento
    - Imagen capturada por la cámara con actualización constante
- **Permitir la representación de elementos móviles con su información asociada**: Para el caso que nos ocupa, el sistema tiene que localizar y actualizar la posición de los vehículos que dispone la empresa de seguridad. Cada elemento móvil dispondrá de un receptor *GPS (Global Positioning System)* con el que obtendremos la posición exacta del elemento en cada instante. Periódicamente cada elemento enviará su posición geográfica a la central. De cada elemento móvil el sistema tiene que ofrecer la siguiente información:
    - Código y descripción del elemento
    - Posición actualizada del elemento y ruta del mismo
    - Imagen capturada por la cámara asociada al vehículo

5.2 Análisis de requisitos

Una vez que conocemos los requisitos de la información que tenemos que representar en los mapas, estamos en disposición de determinar las entidades que necesitaremos para la gestión de la información georreferenciada y posteriormente su presentación mediante el sistema que determinemos.

De forma general determinamos que nuestro sistema de información geográfica trabajará con dos tipos de entidades:

- **Elementos**: Definen los elementos del sistema que queremos representar en los mapas. Estos elementos se especializarán en dos tipos de elementos: videocámaras y patrullas de vigilancia
- **Posiciones**: Establecen las coordenadas de cada elemento en un momento dado. Con estas coordenadas podremos establecer la última posición conocida de un elemento dado. Como es lógico cada elemento sólo podrá tener una coordenada en un momento temporal determinado.

5.3 Arquitectura

Para la construcción del módulo de presentación geográfica del sistema de localización de elementos de seguridad hemos decidido utilizar el servicio de mapas ofrecido por Google (Google Maps) [51].

Utilizar este servicio de mapas aporta grandes ventajas sobre la utilización de cualquier software GIS tradicional. A continuación destacamos las más representativas:

- **Facilita la distribución de mapas**: El servicio de Google dispone de cartografía actualizada a distintas resoluciones de todo el planeta. En nuestro caso nos interesa la cartografía de las grandes ciudades españolas. Google nos ofrece la cartografía actualizada y a buena resolución de estas ciudades, con lo que evitamos el problema que supone en los GIS tradicionales [1] la adquisición de cartografía de la zona de interés. Otra opción sería la utilización de un servicio de WMS que nos ofrezca la cartografía de nuestro interés.
- **Reducción de costes**: El servicio de Google en su versión actual es gratuito con las limitaciones que hemos presentado en el apartado *4.2 Licencia de uso del servicio y la API*. La implementación de un servicio de este estilo desde cero tendría un coste bastante elevado.
- **Facilidad de uso**: El servicio Google Maps dispone de un interfaz muy sencillo que facilita su utilización a los usuarios. Desde el punto de vista de la integración de sistemas, la API de programación Google Maps permite que de forma sencilla los programadores puedan presentar información georreferenciada sobre los mapas ofertados por Google.
- **Facilita la integración con sistemas externos**: Como hemos presentado en el apartado *4 API de Google Maps* del presente documento, Google facilita una API de programación que ofrece la posibilidad de interaccionar con sus servicio y la posibilidad de integrar sus mapas en aplicaciones externas. Por otro lado la API de Google Maps [52] ofrece la posibilidad de personalizar los mapas y la presentación de la información que se presente en ellos.

A continuación aclaramos la arquitectura del sistema[2] con un diagrama:

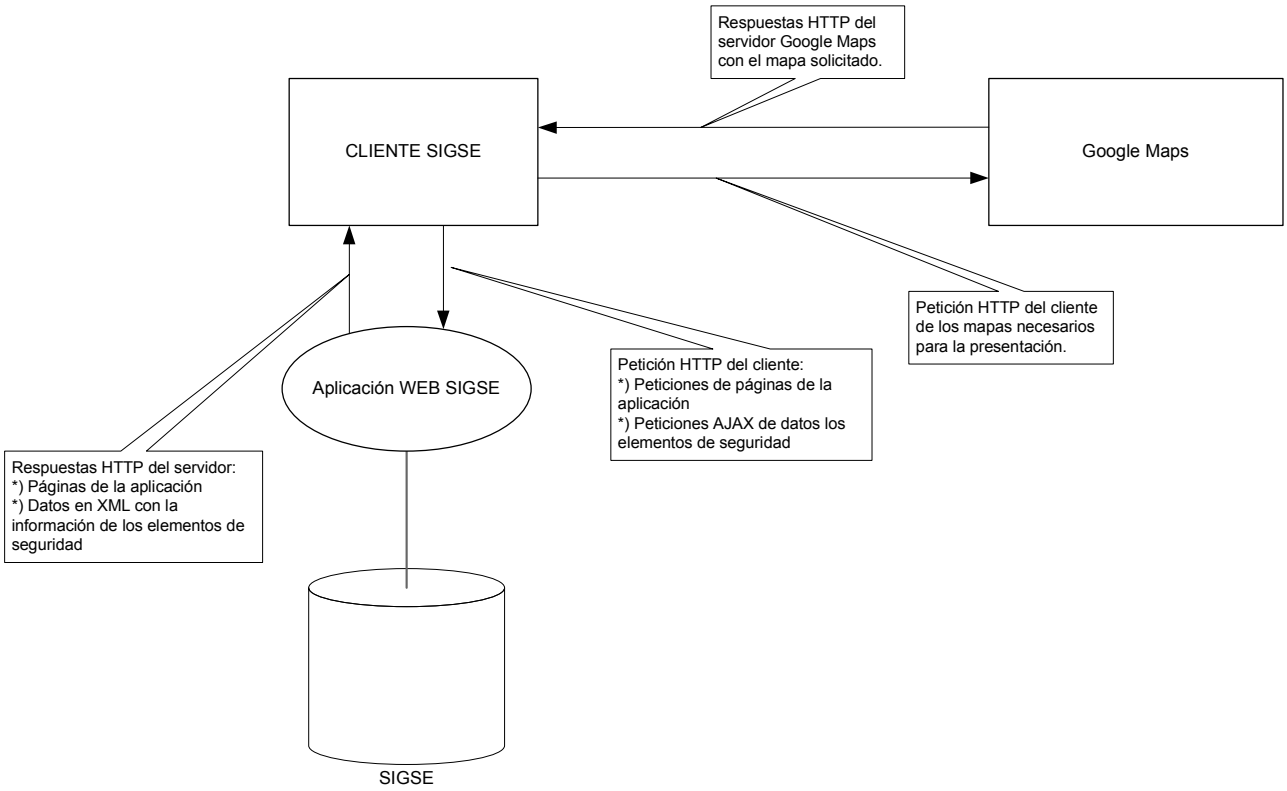

Figura 5.1 Arquitectura del sistema

[2] Para referirnos al sistema de localización de elementos de seguridad que construiremos para la empresa se seguridad utilizaremos la denominación de SIGSE

A continuación aclaramos la distribución del sistema con un diagrama que facilita la compresión de la arquitectura propuesta:

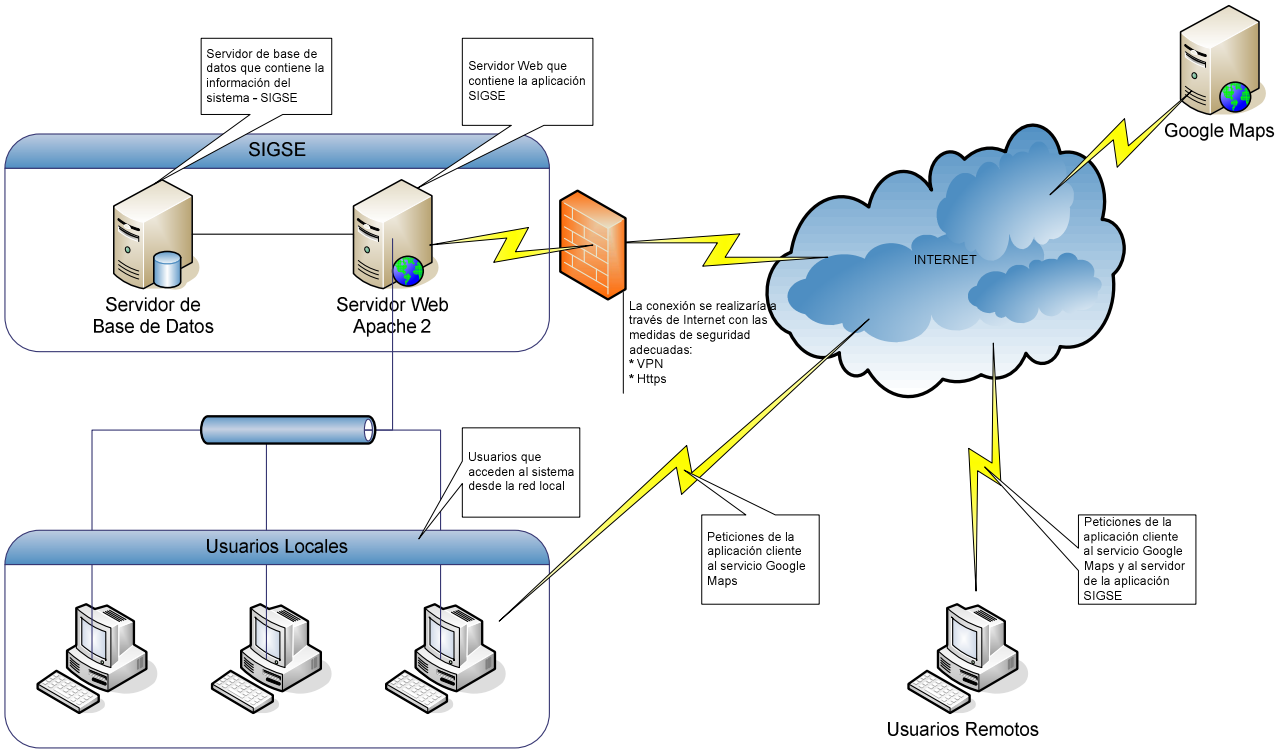

Figura 5.2 Distribución del sistema

5.4 Diseño

A continuación presentaremos el modelo de datos general que nos permitirá construir el módulo de localización de información geográfica. Este modelo lo ampliaremos cuando efectuemos el diseño del sistema de localización de elementos de seguridad, incorporando la información particular del sistema.

Como hemos introducido en el análisis de requisitos del sistema, necesitaremos las siguientes entidades:

- **Elemento**: Un elemento representa cualquier entidad que queremos posicionar en un mapa. Para nuestro sistema tendremos dos especializaciones de esta entidad que nos permitirán representar los elementos de seguridad: videocámaras y patrullas de vigilancia
- **Posicion**: Una posición representa las coordenadas de un elemento en un instante dado. Con estas coordenadas podremos establecer la última posición conocida de un elemento dado. Todo elemento tendrá al menos una posición y en el caso de los elementos móviles tendremos varias posiciones, siempre en instantes diferentes, que nos permitirán representar su movimiento y la ruta recorrida. Una posición identificará las coordenadas para la presentación del elemento en Google Maps. Para ello necesitaremos recoger la latitud y longitud del sistema
- **Ruta**: Una ruta está compuesta por un conjunto de posiciones que tienen que recorrer el elemento asociado. En nuestro sistema lo utilizaremos para conocer y representar las rutas de los elementos de seguridad

A continuación presentamos un diagrama con las entidades identificadas previamente:

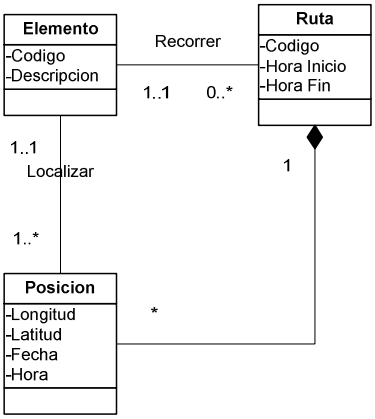

Figura 5.3 Modelo de datos GIS

Para la persistencia del modelo de datos que acabamos de presentar hemos determinado construir una base de datos que almacenará los datos del sistema. Para la construcción de la base de datos se han evaluado distintos sistemas gestores de base de datos como Oracle y SqlServer. Tal y como se adelantó en el plan de proyecto, para la construcción del sistema hemos optado por el gestor de base de datos MySQL. Este gestor dispone de las características necesarias para dar soporte de persistencia a los datos del sistema y no requiere la adquisición de licencias de utilización.

A continuación presentamos el modelo de datos de la base de datos con la que gestionaremos la persistencia de las entidades identificadas previamente:

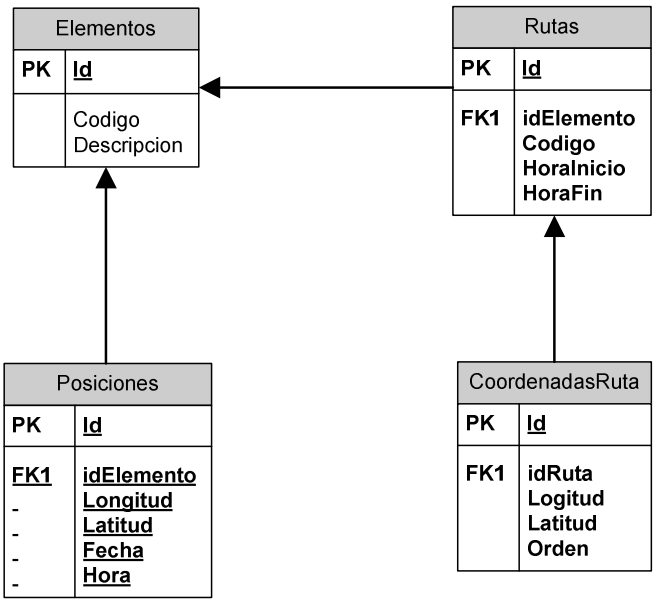

Figura 5.4 Diseño de tablas del GIS

A continuación presentamos algunos problemas que se nos presentan a la hora de diseñar el sistema de información geográfica:

- **Actualización de las posiciones de los elementos móviles**: Como hemos presentado en el los requisitos suponemos que los elementos móviles disponen de un sistema GPS que les permite obtener su posición geográfica. Al mismo tiempo suponemos que estos elementos móviles transmitirán por *GSM (Global System for Mobile communications)* o *GRPS (General Packet Radio Service)* sus coordenadas actualizadas a la base central. Dado que no disponemos de toda la infraestructura necesaria para un diseño real de esta parte del sistema, hemos optado por pre-cargar las posiciones geográficas de los distintos elementos. De esta forma inicializaremos el sistema con un número de posiciones asociadas a distintos momentos temporales con las que simularemos el mecanismo de actualización de posiciones descrito para los elementos móviles.

- **Falta de requisitos sobre los sistemas de seguridad**: No consideremos adecuado profundizar en los requisitos necesarios para la gestión de los distintos recursos que dispone una empresa de seguridad. El objetivo del diseño realizado es construir un sistema que permita localizar los elementos, y que permita en un futuro asociar a esta localización toda la información necesaria para la gestión de la empresa de seguridad y la incorporación de herramientas de análisis de la información desde el punto de vista geográfico (planificación de rutas, optimización de recursos, etc.).

- **Recogida de coordenadas de prueba**: Para las pruebas del sistema necesitaríamos recoger coordenadas de elementos reales con un sistema GPS que pudiéramos cargar en la base de datos. Ante la imposibilidad de contar con un sistema GPS construiremos una pequeña página que nos permita utilizar el servicio de Google Maps como identificador de coordenadas.

  A continuación presentamos una captura del sistema que hemos empleado para la recopilación de las coordenadas:

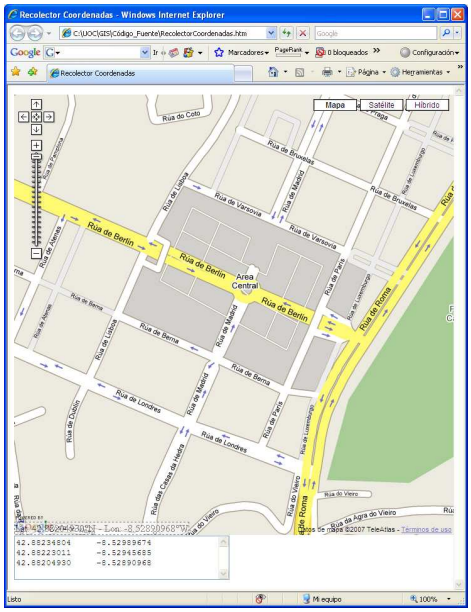

Figura 5.5 Sistema recolector de coordenadas

6 Sistema de localización de elementos de seguridad – SIGSE

En este apartado presentaremos una descripción completa del proceso que se ha realizado para diseñar el sistema para la localización de elementos de seguridad. De esta forma presentaremos desde los requisitos que tiene que cumplir el sistema hasta el diseño y su posterior implantación de la solución propuesta.

6.1 Introducción

El sistema de localización geográfica de elementos de seguridad, en adelante SIGSE, surge en respuesta a la necesidad de localizar geográficamente todos los recursos que dispone una empresa de seguridad.

Para ello construiremos un sistema de información con capacidades de representar información georreferenciada sobre mapas de las zonas sobre las que la empresa presta sus servicios de seguridad.

Los mapas que utilizaremos para la presentación de la información del sistema los obtendremos gracias a la utilización del servicio de mapas de Google Maps. Para la utilización de este servicio necesitamos una cuenta de cuenta de Google que nos permita obtener una clave de acceso al servicio de mapas de Google que utilizaremos en todo el desarrollo.

6.2 Captura de requisitos

Una empresa de seguridad nos transmite la problemática que tiene en la actualidad para la gestión de las localizaciones de los recursos de seguridad que dispone y que prestan los servicios que comercializa. Ante estos problemas creen que un sistema de información informático podría solventar gran parte de sus problemas, permitiendo incluso optimizar la gestión de sus recursos.

El sistema de información que necesitan tiene que permitir la visualización la distribución sobre un mapa de los distintos elementos de seguridad que dispone. En concreto la empresa dispone de los siguientes elementos de seguridad repartidos por el territorio:

- **Videocámaras**: Elemento estático que distribuyen por la geografía en función de los servicios de vigilancia que ofertan.
- **Elementos móviles**: La empresa de seguridad dispone de distintas patrullas que utilizan en los servicios de seguridad que obligan a cubrir grandes superficies.

Además de representar estos elementos sobre un mapa la empresa nos transmite que el sistema debe permitir los siguientes requisitos:

- Localizar los elementos de seguridad en un determinado momento sobre el territorio.
- Mostrar los movimientos que realizan los elementos móviles sobre el territorio.
- Permitir consultar las imágenes transmitidas por las cámaras de seguridad.
- Permitir consultar información adicional de los elementos que la aplicación muestra sobre los mapas.

Desde el punto de vista tecnológico la empresa no nos transmite ningún tipo de restricción. A la hora de seleccionar las tecnologías tendremos que tener en cuenta que se trata de una pequeña empresa y que por lo tanto será necesario seleccionar aquellas tecnologías que requieran menor inversión para la empresa.

6.3 Análisis

Partiendo de los requisitos que hemos obtenido para la construcción del sistema abordaremos la tarea de análisis de los requisitos del sistema. Para ello identificaremos los actores que intervienen en el sistema y los casos de uso que tiene que cubrir el sistema para satisfacer los requisitos capturados.

6.3.1 Actores

A la vista de los requisitos capturados en el apartado *6.2 Captura de requisitos*, identificamos los siguientes actores que intervendrán en mayor o menor medida en el sistema:

- **Gestor de seguridad**: Serán los usuarios reales del sistema que realizarán las peticiones de localización de los elementos de seguridad.
- **Actualizador o Timer**: Sistema autónomo que periódicamente realiza peticiones de actualización de las posiciones geográficas de determinados elementos al sistema.
- **GoogleMaps**: Servidor de mapas que utilizará el sistema para la localización de los elementos de seguridad.

En futuras versiones del sistema será necesario crear un módulo de administración que permita la gestión de los datos de los elementos de seguridad de la empresa. Por ello identificamos un actor adicional que será necesario para los casos de uso de este módulo de administración:

- **Administrador del sistema**: Este usuario será el encargado de crear los elementos de seguridad y cargar sus posiciones geográficas en el sistema.

6.3.2 Casos de uso

A continuación presentaremos los casos de uso que tiene que cubrir el sistema. Para cada caso de uso se indican la siguiente información:

- **Descripción**: Se efectúa una descripción indicando qué ocurre en el caso de uso.
- **Actores**: Se indican los actores que intervienen en el mismo.
- **Precondiciones**: Se indican las condiciones previas a cumplir.
- **Errores**: Se apuntan los errores que se pueden producir.
- **Entradas**: Se indican las entradas o parámetros del caso de uso.
- **Salidas:** Se indica cuáles son las salidas o resultados.
- **Pasos**: Finalmente, se enumeran los pasos a seguir para completar el caso.

6.3.2.1 Caso de uso 1: Localizar Videocámaras

Con este caso de uso ofertaremos al usuario la posibilidad de localizar las videocámaras de la empresa sobre un mapa e incorporando información adicional de la misma. A mayores se podrá visionar la imagen actualizada que está capturando la videocámara en cada momento.

A continuación sintetizamos la información que nos permite describir y analizar el caso de uso:

- **Descripción**: Un gestor de seguridad necesita averiguar la distribución de las videocámaras que tiene activas en un determinado momento.
- **Actores**: Gestor de seguridad, Google Maps.
- **Precondiciones**: Es necesario que se carguen en el sistema los datos de las videocámaras y sus posiciones.
- **Errores**: Coordenadas de la videocámara incorrectas.
- **Entradas**: No se requiere ningún tipo de entrada.
- **Salidas**: Ubicación geográfica de las distintas videocámaras.
- **Pasos**:
    1. El gestor de seguridad solicita la localización de las videocámaras
    2. El sistema consulta en base de datos las videocámaras y las últimas posiciones conocidas en el momento de la consulta. Para cada videocámara obtenemos los siguientes datos:
        - Código

- Descripción de la videocámara
- Últimas coordenadas conocidas: longitud y latitud
- Fecha y hora de la última actualización
- URL que nos permitirá consultar las imágenes capturadas por la cámara

3. El sistema solicita al servidor de Google el mapa de la zona en la que se posicionan las videocámaras
4. El sistema muestra la posición de las videocámaras sobre el mapa

A continuación presentamos un diagrama que presenta el caso de uso descrito anteriormente:

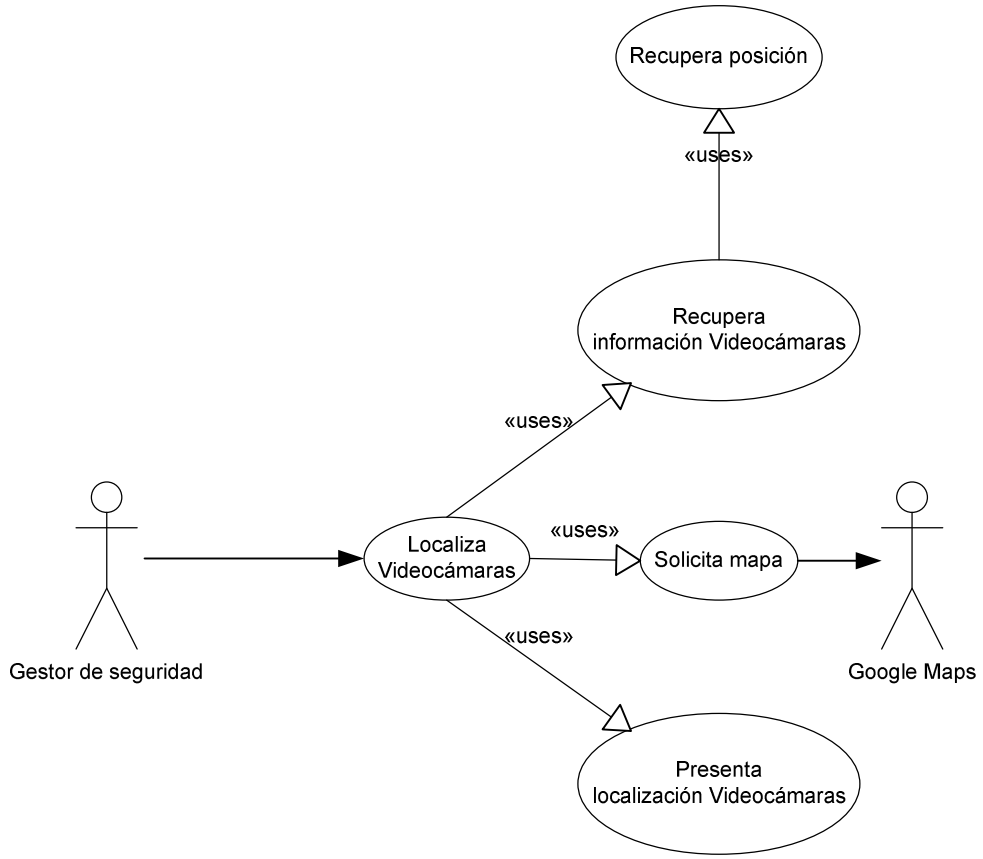

Figura 6.1 Caso de uso 1: Localizar videocámaras

6.3.2.2 Caso de uso 2: Localizar posición actualizada de los elementos móviles

Este caso de uso sintetiza la funcionalidad que resuelve el requisito identificado que demandaba la localización de los elementos móviles de la empresa. De esta forma el sistema permitirá posicionar los elementos móviles de la empresa sobre un mapa actualizando su posición de forma regular.

A continuación sintetizamos la información que nos permite describir y analizar el caso de uso:

- **Descripción**: Un gestor de seguridad necesita averiguar la distribución de las patrullas que tiene activas en un determinado momento, actualizando su posición de forma periódica. Los elementos móviles comunican por *GRPS* la última posición que obtienen con el sistema *GPS*. Esta comunicación se realiza a intervalos de tiempo iguales.
- **Actores**: Gestor de seguridad, Google Maps, Actualizador.
- **Precondiciones**: Es necesario que se carguen en el sistema los datos de las patrullas y sus posiciones. Dado que en esta versión del sistema no disponemos de infraestructura para que los elementos móviles comuniquen su información al sistema, tendremos que cargar en la base de datos un conjunto de posiciones ficticias asociadas a momentos temporales distintos.
- **Errores**: Coordenadas de la patrulla incorrectas.

- **Entradas**: No se requiere ningún tipo de entrada.
- **Salidas**: Ubicación geográfica actualizada de las distintas patrullas.
- **Pasos**:
    1. El gestor de seguridad solicita la localización de las patrullas
    2. El sistema consulta en base de datos los elementos móviles y las últimas posiciones conocidas en el momento de la consulta. Para cada elemento obtenemos los siguientes datos:
        - Código
        - Descripción del elemento
        - Últimas coordenadas conocidas: longitud y latitud
        - Fecha y hora de la última actualización
        - URL que nos permitirá consultar las imágenes capturadas por el elemento
    3. El sistema solicita al servidor de Google el mapa de la zona en la que se posicionan las patrullas
    4. El actualizador solicita periódicamente la actualización de las posiciones de los elementos móviles
    5. El sistema muestra la posición de los elementos móviles sobre el mapa

A continuación presentamos un diagrama que presenta el caso de uso descrito anteriormente:

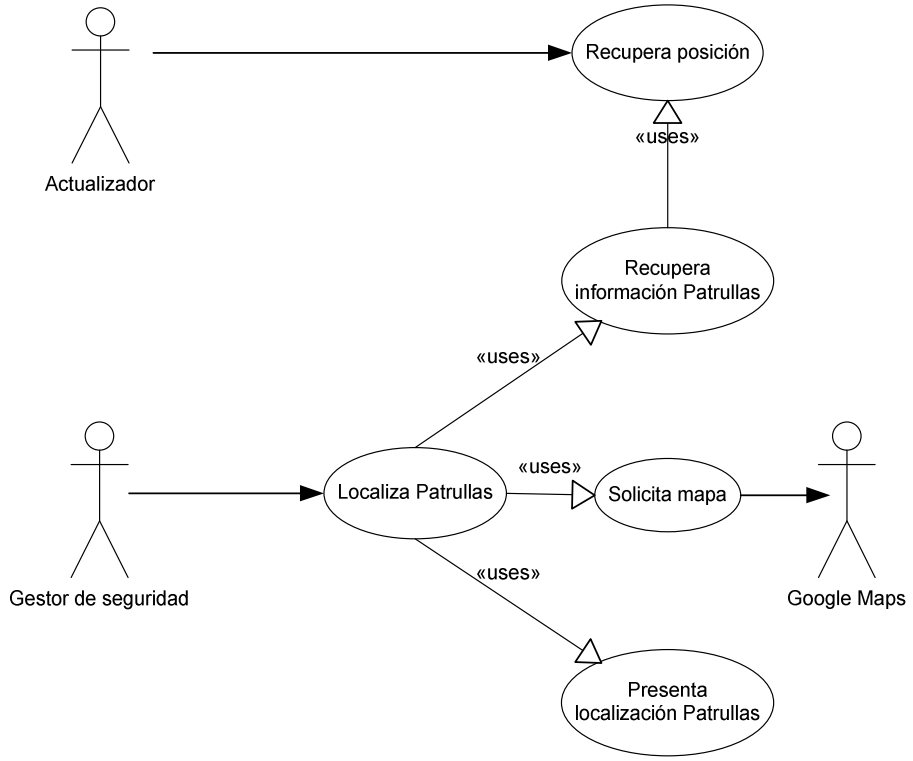

Figura 6.2 Caso de uso 2: Localizar posición actualizada de los elementos móviles

6.3.2.3 Caso de uso 3: Localizar la ruta asociada a un elemento móvil

Este caso de uso pretende dar respuesta a la necesidad de conocer las rutas que tienen asignados los elementos móviles de la empresa.

A continuación sintetizamos la información que nos permite describir y analizar el caso de uso:

- **Descripción**: Un gestor de seguridad necesita averiguar la ruta asignada a las patrullas que tiene activas en un determinado momento.
- **Actores**: Gestor de seguridad, Google Maps.
- **Precondiciones**: Es necesario que se carguen en el sistema los datos de las patrullas y las coordenadas de las rutas que tienen asignadas.
- **Errores**: Coordenadas de la ruta incorrectas.
- **Entradas**: Patrulla sobre la que se quiere consultar la ruta asignada.
- **Salidas**: Ubicación geográfica de la ruta asignada a la patrulla.
- **Pasos**:
    1. El gestor de seguridad solicita la localización de la ruta asignada a una patrulla
    2. El sistema consulta en base de datos los elementos móviles y las rutas asignadas a los mismos. Para cada elemento obtenemos los siguientes datos:
        - Código
        - Descripción del elemento
        - Últimas coordenadas conocidas: longitud y latitud
        - Fecha y hora de la última actualización
        - URL que nos permitirá consultar las imágenes capturadas por el elemento
        - Puntos que definen la ruta asociada. Cada punto está compuesto de las coordenadas que nos permitirán su localización y un campo de orden que nos permitirán establecer una secuencia dentro de la ruta definida.
    3. El sistema solicita al servidor de Google el mapa de la zona en la que se posicionan las rutas
    4. El sistema muestra la posición de las rutas sobre el mapa

A continuación presentamos un diagrama que presenta el caso de uso descrito anteriormente:

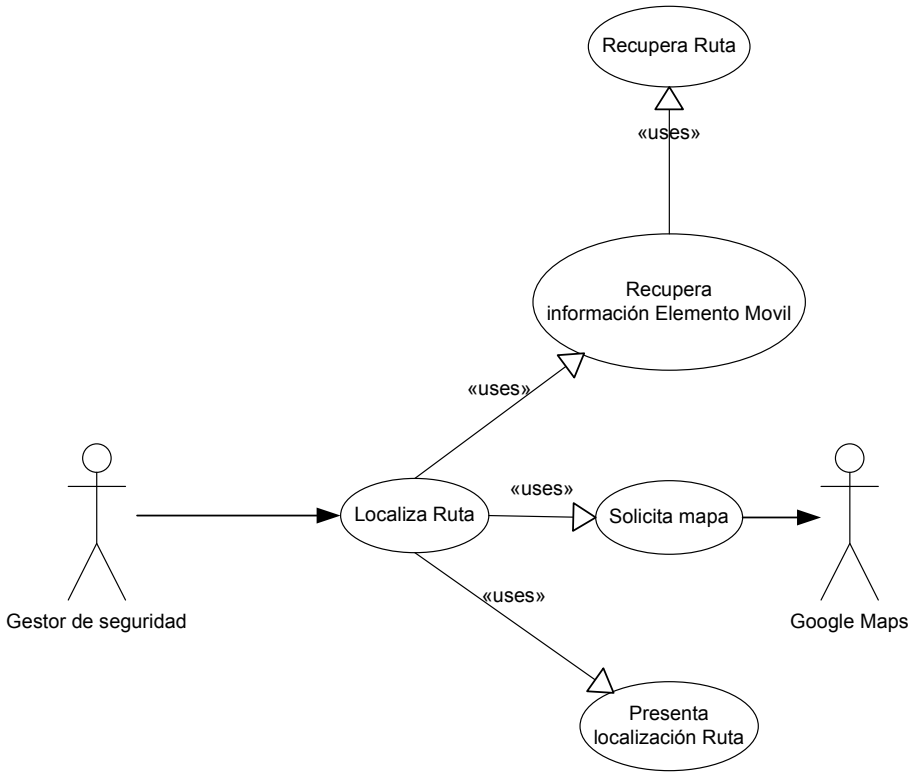

Figura 6.3 Caso de uso 3: Localizar la ruta asociada a un elemento móvil

6.3.3 Identificación de problemas

A continuación presentamos algunos problemas que se nos presentan a la hora de diseñar el sistema de localización de elementos de seguridad:

- **Actualización de las posiciones de los elementos móviles**: Como hemos presentado en el los requisitos suponemos que los elementos móviles disponen de un sistema GPS que les permite obtener su posición geográfica. Al mismo tiempo suponemos que estos elementos móviles transmitirán por GSM sus coordenadas actualizadas a la base central. Dado que no disponemos de toda la infraestructura necesaria para un diseño real de esta parte del sistema, hemos optado por pre-cargar las posiciones geográficas de los distintos elementos. De esta forma inicializaremos el sistema con un número de posiciones asociadas a distintos momentos temporales con las que simularemos el mecanismo de actualización de posiciones descrito para los elementos móviles.
- **Falta de requisitos sobre los sistemas de seguridad**: No consideremos adecuado profundizar en los requisitos necesarios para la gestión de los distintos recursos que dispone una empresa de seguridad. El objetivo del diseño realizado es construir un sistema que permita localizar los elementos, y que permita en un futuro asociar a esta localización toda la información necesaria para la gestión de la empresa de seguridad y la incorporación de herramientas de análisis de la información desde el punto de vista geográfico (planificación de rutas, optimización de recursos, etc.).
- **Presentación de imágenes de las videocámaras**: Los requisitos del sistema establecen que entre otra información, al localizar una videocámara en el mapa el usuario debe poder visualizar las imágenes que está capturando en ese momento el elemento de seguridad. Suponemos que las videocámaras están conectadas a un servicio web que provee las imágenes a través de Internet. Como no disponemos de este servicio utilizaremos videos de los servicios de Google Video y You Tube a modo de ejemplo. Cuando dispongamos del servidor de videos de las cámaras sólo tendremos que reemplazar las URLs asociadas a los elementos de seguridad.

6.4 Diseño

Partiendo del análisis realizado en el apartado anterior abordaremos la tarea de diseño del sistema. Para ello identificaremos entidades que intervienen en el sistema y definiremos el modelo de datos que nos permitirá gestionar la persistencia de las entidades identificadas. A mayores estableceremos las secuencias que tendremos que definir entre las entidades identificadas para implementar los casos de uso identificados durante el análisis.

6.4.1 Modelo de datos

A continuación presentaremos el modelo de datos general que nos permitirá construir el sistema que satisfaga los requisitos identificados previamente. En este modelo gestionaremos la información particular del sistema de gestión de información de elementos de seguridad y las entidades que hemos definido como necesarias para la gestión de la información geográfica (apartado *5.4 Diseño*).

Para la gestión de la información del sistema identificamos las siguientes entidades:

- **Elemento**: Un elemento representa cualquier entidad que queremos posicionar en un mapa. Para nuestro sistema tendremos dos especializaciones de esta entidad que nos permitirán representar los elementos de seguridad:
    - **Videocámara**: Representa a cualquiera de las videocámaras que posee la empresa de seguridad. Cada videocámara tiene asociada una URL en la que publican las imágenes que captura.
    - **Patrulla** de vigilancia: Representa a cualquiera de los elementos móviles que posee la empresa de seguridad. Cada patrulla tendrá asociada una ruta que determinará el recorrido que tienen asignado para prestar el servicio.
- **ServicioSeguridad**: Un servicio de seguridad representa a los distintos servicios de seguridad que presta la empresa con sus clientes. Aunque esta entidad no es necesaria para la construcción del sistema de localización de elementos de seguridad, hemos decidido incluirla para remarcar que este sistema seguramente estará encuadrado dentro de un sistema de información más completo que informatiza la totalidad de la gestión de la empresa de seguridad. Siguiendo este razonamiento la

entidad “ServicioSeguridad” es el punto de unión entre el sistema que estamos diseñando con el sistema de información completo: suponemos que las videocámaras y patrullas se distribuyen en el territorio para cumplir unos servicios de seguridad que presta la empresa.

- **Posicion**: Una posición representa las coordenadas de un elemento en un instante dado. Con estas coordenadas podremos establecer la última posición conocida de un elemento dado. Todo elemento tendrá al menos una posición y en el caso de los elementos móviles tendremos varias posiciones, siempre en instantes diferentes, que nos permitirán representar su movimiento y la ruta recorrida. Una posición identificará las coordenadas para la presentación del elemento en Google Maps. Para ello necesitaremos recoger la latitud y longitud del sistema.
- **Ruta**: Una ruta está compuesta por un conjunto de posiciones que tienen que recorrer el elemento asociado. En nuestro sistema lo utilizaremos para conocer y representar las rutas de los elementos de seguridad.

A continuación presentamos un diagrama en el que se pueden ver las relaciones entre las distintas entidades identificadas:

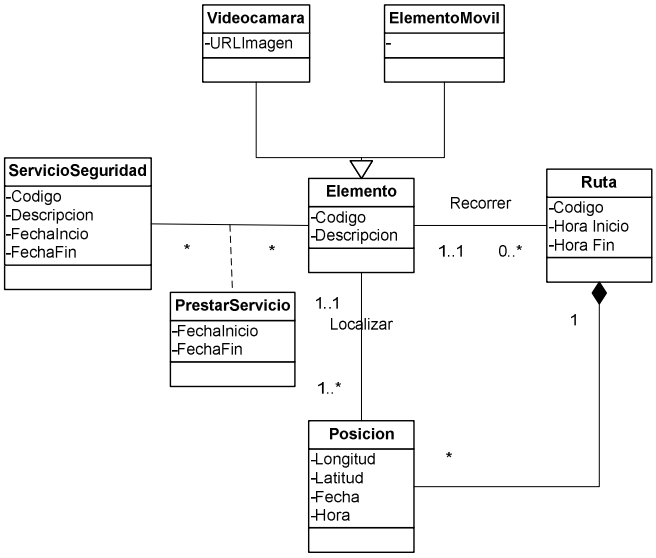

Figura 6.4 Diagrama de entidades del sistema

Para gestionar la persistencia del modelo de datos que acabamos de presentar diseñamos una base de datos que almacenará todos los datos del sistema. Para la construcción de la base de datos se han evaluado distintos sistemas gestores de base de datos como Oracle y SqlServer. De acuerdo con lo descripto en el apartado 5.4 hemos optado por el gestor de base de datos MySQL.

A continuación presentamos el modelo de datos de la base de datos con la que gestionaremos la persistencia de las entidades identificadas previamente:

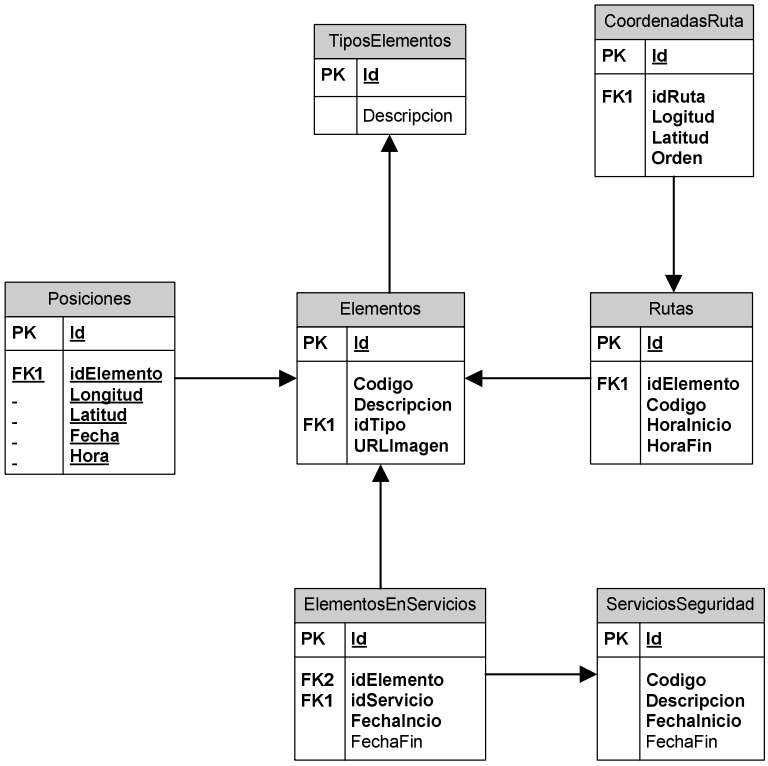

Figura 6.5 Diagrama relacional del sistema

En la siguiente tabla presentamos una descripción detallada de las distintas entidades y los atributos que definimos para las mismas:

Nombre	**TiposElementos**	
Descripción	Entidad que permite clasificar los elementos de seguridad. Inicialmente diferenciaremos dos tipos: Videocámaras y Elementos móviles (patrullas)	
Atributos	Nombre	Descripción
	Id	Identificador interno del tipo.
	Descripción	Denominación textual del tipo.
Nombre	**Posiciones**	
Descripción	Entidad que permite almacenar las posiciones asociadas a los elementos de seguridad. Cada elemento de seguridad puede tener asociada una o varias posiciones. En un instante temporal determinado un elemento sólo podrá tener asociada una posición.	
Atributos	Nombre	Descripción
	Id	Identificador interno de la posición.
	idElemento	Referencia al elemento al que está asociada la posición.
	Longitud	Componente longitud de la coordenada en la que está posicionado el elemento.
	Latitud	Componente latitud de la coordenada en la que está posicionado el elemento.
	Fecha	Fecha en la que se obtuvo la posición
	Hora	Hora en la que se obtuvo la posición

Nombre	**CoordenadasRutas**	
Descripción	Entidad que permite almacenar las coordenadas que definen la ruta que tiene asociado un elemento de seguridad móvil. Una ruta estará compuesta de dos o más coordenadas (puntos) con una determinada secuencia.	
Atributos	Nombre	Descripción
	Id	Identificador interno de la CoordenadaRuta.
	idRuta	Referencia a la ruta a la que está asociada la coordenada.
	Longitud	Componente longitud de la coordenada.
	Latitud	Componente latitud de la coordenada.
	Orden	Número que determina la secuencia de puntos que definen la ruta.
Nombre	**Rutas**	
Descripción	Entidad que permite almacenar las rutas que tienen asociados los elementos de seguridad móviles.	
Atributos	Nombre	Descripción
	Id	Identificador interno de la Ruta.
	idElemento	Referencia al elemento al que está asociada la ruta.
	Codigo	Código identificativo de la ruta.
	HoraInicio	Hora prevista en la que se inicia la ruta.
	HoraFin	Hora prevista en la que se finaliza la ruta.
Nombre	**Elementos**	
Descripción	Entidad que permite almacenar los elementos de seguridad. Los elementos de seguridad pueden ser de distintos tipos. Inicialmente se gestionan elementos de dos tipos: videocámaras y elementos móviles (patrullas).	
Atributos	Nombre	Descripción
	Id	Identificador interno del Elemento.
	idTipo	Referencia al tipo al que pertenece el elemento.
	Codigo	Código identificativo del elemento de seguridad.
	Descripcion	Descripción del elemento de seguridad.
	URLImagen	URL del servidor de imágenes que distribuye la señal de la videocámara.
Nombre	**ServiciosSeguridad**	
Descripción	Entidad que permite almacenar los servicios de seguridad que ofrece la empresa a sus clientes. Los servicios de seguridad pueden tener asociados varios elementos de seguridad que se utilizan para la prestación del servicio.	
Atributos	Nombre	Descripción
	Id	Identificador interno del Servicio.
	Codigo	Código identificativo del servicio.
	Descripcion	Descripción del servicio.
	FechaIncio	Fecha de inicio de la prestación del servicio.
	FechaFin	Fecha de finalización de la prestación del servicio.

Nombre	ElementosEnServicios	
Descripción	Entidad que permite almacenar la asociación de los elementos de seguridad a los servicios ofertados por la empresa.	
Atributos	Nombre	Descripción
	Id	Identificador interno del ElementoEnSeguridad.
	idElemento	Referencia al elemento de seguridad.
	idServicio	Referencia al servicio de seguridad.
	FechaIncio	Fecha de inicio de la asociación del elemento de seguridad al servicio.
	FechaFin	Fecha de finalización de la asociación del elemento de seguridad al servicio.

Tabla 3 Entidades del sistema SIGSE

6.4.2 Diagramas de secuencia

A continuación presentaremos los diagramas de secuencia de los distintos objetos de la aplicación. En el diagrama de secuencia se pueden observar las interacciones de los objetos a lo largo del tiempo. Presentamos un diagrama de secuencia para cada uno de los casos de uso que identificamos en el apartado *6.3.2 Casos de uso*.

6.4.2.1 Localizar Videocámaras

El siguiente diagrama resume el proceso de localización de las videocámaras en un mapa a partir de la acción del usuario:

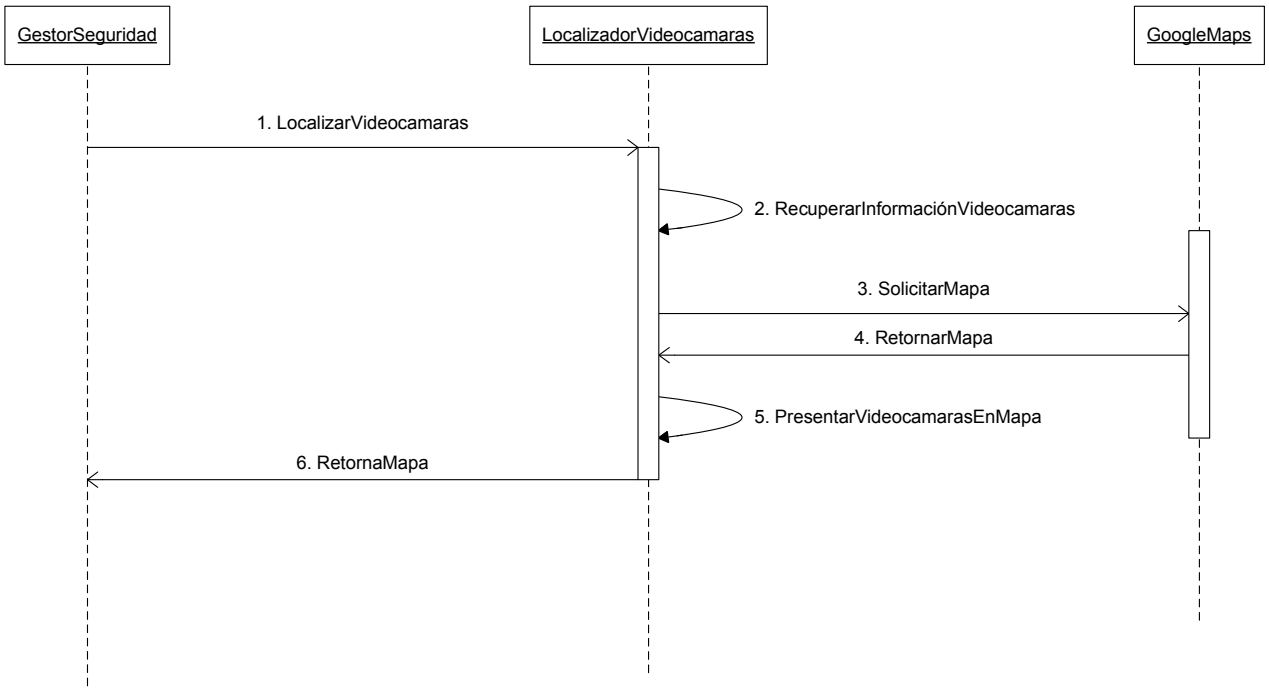

Figura 6.6 Diagrama de secuencia: Localizar Videocámaras

A continuación presentamos una breve descripción del diagrama:

1. **LocalizarVideocamara():** El usuario invoca la funcionalidad del sistema que le permite localizar las videocámaras de la empresa.

2. **RecuperaInformacionVideocamaras():** El sistema solicita al sistema gestor de base de datos que le retorne la información de las videocámaras de la empresa.
3. **SolocitarMapa():** El sistema genera una petición al servidor de mapas para que le retorne un mapa de la zona en la que se localizan las videocámaras.
4. **RetornarMapa():** El servidor de mapas retorna el mapa solicitado.
5. **RepresentarVideocamaras():** El sistema añade al mapa que le ha retornado el servidor de mapa la información de las videocámaras, utilizando para posicionarlas las últimas coordenadas conocidas
6. **RetornaMapa():** El sistema retorna al usuario el mapa en el que se presenta las videocámaras de la empresa.

6.4.2.2 Localizar posición actualizada de los elementos móviles

El siguiente diagrama resume el proceso de localización de los elementos móviles (patrullas) en un mapa a partir de la petición del usuario:

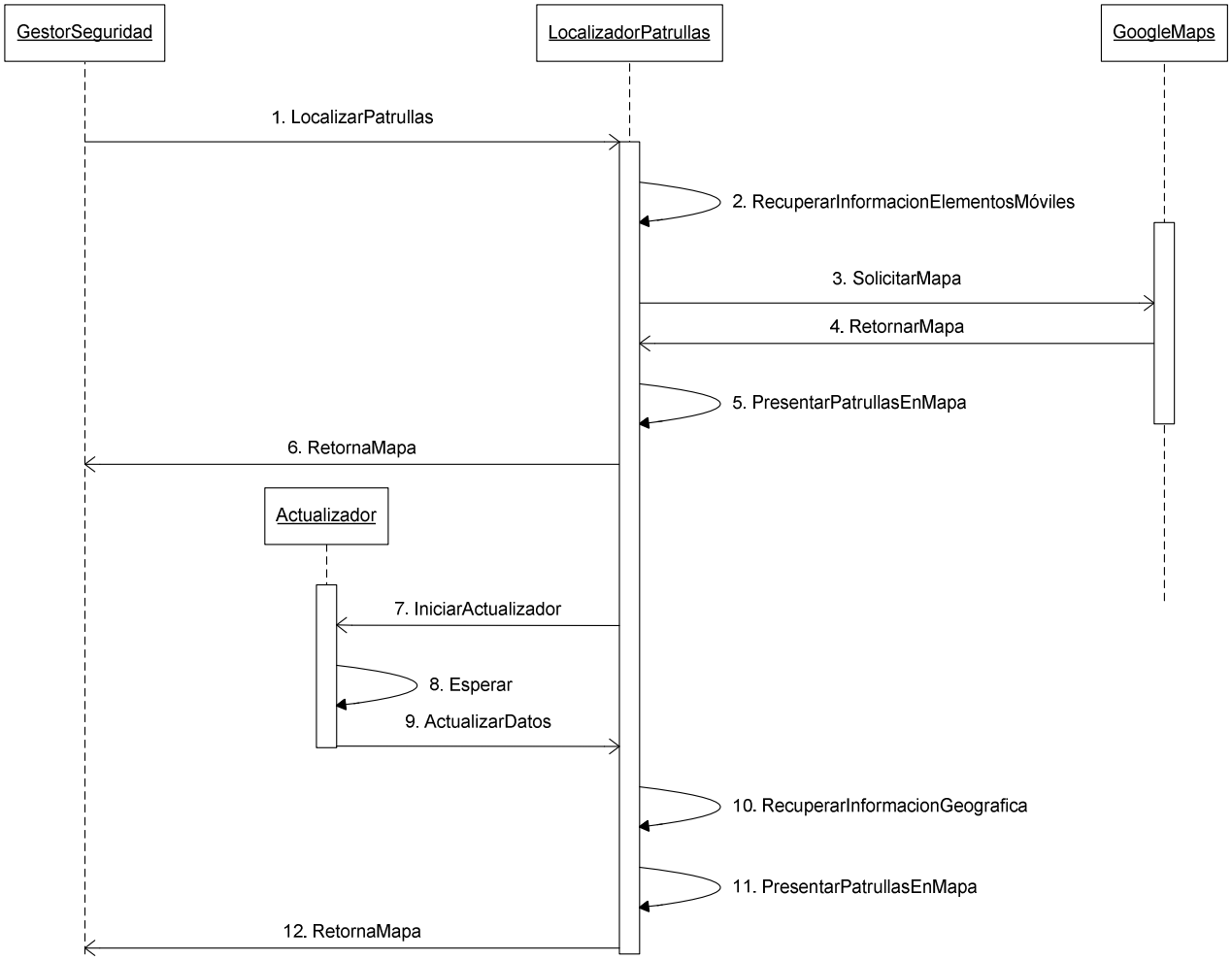

Figura 6.7 Diagrama de secuencia: Localizar posición actualizada de los elementos móviles

A continuación presentamos una breve descripción del diagrama:

1. **LocalizarPatrullas():** El usuario invoca la funcionalidad del sistema que le permite localizar las patrullas de la empresa.
2. **RecuperaInformacionElementosMoviles():** El sistema solicita al sistema gestor de base de datos que le retorne la información de los elementos móviles de la empresa.

3. **SolocitarMapa():** El sistema genera una petición al servidor de mapas para que le retorne un mapa de la zona en la que se localizan las patrullas.
4. **RetornarMapa():** El servidor de mapas retorna el mapa solicitado.
5. **RepresentarPatrullasEnMapa():** El sistema añade al mapa que le ha retornado el servidor de mapa la información de las patrullas, utilizando para posicionarlas las últimas coordenadas conocidas.
6. **RetornaMapa():** El sistema retorna al usuario el mapa en el que se presenta las patrullas de la empresa.
7. **IniciarActualizador():** Se inicia el proceso de actualización de posiciones que hará que se actualice cada cierto tiempo la posición de las patrullas.
8. **Esperar():** El actualizador espera hasta que llega el siguiente intervalo de actualización de posiciones.
9. **ActualizarDatos():** El actualizador solicita al sistema que actualice la posición de las patrullas en el mapa.
10. **RecuperaInformacionGeografica():** El sistema solicita al sistema gestor de base de datos que le retorne la última posición conocida de los elementos móviles de la empresa.
11. **RepresentarPatrullas():** El sistema añade al mapa que le ha retornado el servidor de mapa la información de las patrullas, utilizando para posicionarlas las últimas coordenadas conocidas.
12. **RetornaMapa():** El sistema retorna al usuario el mapa en el que se presenta las patrullas de la empresa.

6.4.2.3 Localizar la ruta asociada a un elemento móvil

El siguiente diagrama resume el proceso de localización de las rutas de los elementos móviles (patrullas) en un mapa a partir de la petición del usuario:

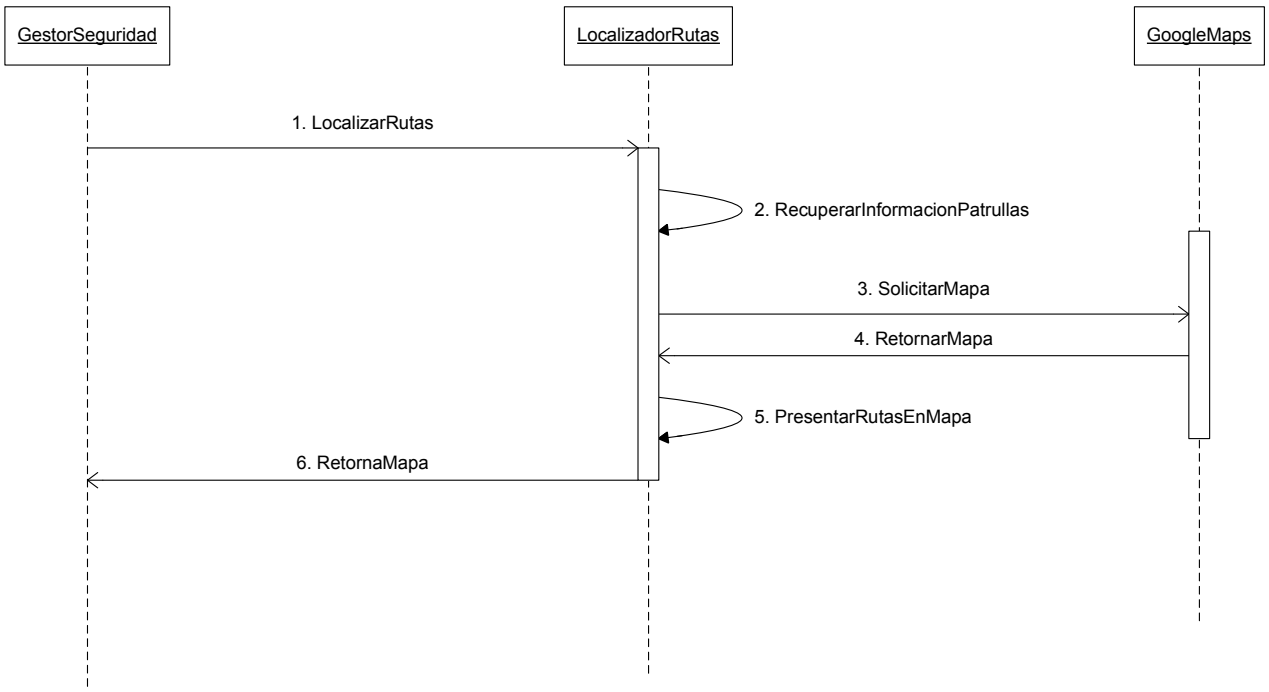

Figura 6.8 Diagrama de secuencia: Localizar la ruta asociada a un elemento móvil

A continuación presentamos una breve descripción del diagrama:

1. **LocalizarRutas():** El usuario invoca la funcionalidad del sistema que le permite localizar las rutas de los elementos móviles de la empresa.
2. **RecuperaInformacionPatrullas():** El sistema solicita al sistema gestor de base de datos que le retorne la información de las rutas de las patrullas de la empresa.

3. **SolocitarMapa():** El sistema genera una petición al servidor de mapas para que le retorne un mapa de la zona en la que se localizan las rutas.
4. **RetornarMapa():** El servidor de mapas retorna el mapa solicitado.
5. **RepresentarRutasEnMapa():** El sistema añade al mapa que le ha retornado el servidor de mapa la información de las rutas, utilizando para posicionarlas las coordenadas que definen la ruta.
6. **RetornaMapa():** El sistema retorna al usuario el mapa en el que se presenta las rutas de las patrullas de la empresa.

6.4.3 Arquitectura del sistema

La aplicación estará dividida en tres capas, cada una de las cuales estará claramente delimitada y con unos interfaces claros que permiten la comunicación entre dichas capas.

Las arquitecturas en capas representan una organización jerárquica de los elementos de un sistema, de modo que cada capa proporciona servicios a los elementos de la capa inmediatamente anterior, y se sirve de los servicios que le brindan los elementos de la capa inmediatamente siguiente. Esta arquitectura nos aporta las siguientes ventajas:

- **Niveles de abstracción**: Cada capa resuelve un problema concreto sin depender de las demás.
- **Facilita la reutilización de componentes**: Podemos realizar mejoras en las distintas capas sin que afecten a la integración con el resto de capas del sistema.

Las capas que identificamos en este sistema son:

- **Capa de presentación**: Responsable de la presentación de los resultados ofertados por la aplicación.
- **Capa de negocio**: Responsable de gestionar las reglas lógicas definidas para el sistema.
- **Capa de persistencia**: Responsable de gestionar la persistencia de las entidades de información empleados por el sistema.

Por otro lado el sistema tendrá una arquitectura cliente-servidor clara: la aplicación se modela como un conjunto de servicios o componentes de servidor, que ofrecen unos servicios y un conjunto de clientes que utilizan estos servicios. Desde este punto de vista los integrantes del sistema serán:

- **Servidor**: Servicios que alimentan la aplicación. En nuestro caso particular contaremos con los siguientes servicios: servicio de mapas, servicio de datos y servicio de lógica de negocio.
- **Cliente**: Son los solicitantes de los servicios ofertados por los servidores. En nuestro caso los clientes serán los responsables de ejecutar la capa de presentación del sistema.
- **Protocolo de comunicación**: Las comunicaciones entre los clientes y los servidores se efectuarán mediante el protocolo HTTP.

A Continuación presentamos una descripción de las distintas capas del sistema y de las decisiones arquitectónicas que hemos tomado en cada una de ellas.

6.4.3.1 Capa de presentación

La capa de presentación será la encargada de presentar los datos a los usuarios y por lo tanto es la encargada de localizar los elementos de seguridad en los mapas que nos provee el servicio de mapas de Google. La capa de presentación estará construida en PHP que retornarán al cliente paginas HTML con JavaScript que utilizaremos para generar el dinamismo necesario en la aplicación. En este sentido tenemos que remarcar que este JavaScript que “controla” la presentación de los elementos en pantalla no pertenece a la capa de presentación, aunque sí que se ejecuta del lado del cliente.

La capa de presentación realizará llamadas a la capa de negocio de la aplicación para que le provea de los datos necesarios para realizar la representación de la información. Por otro lado necesitaremos que la capa de negocio se comunique con un servicio externo que le provea los mapas. Este servicio será el servicio de mapas de Google.

6.4.3.2 Capa de negocio

La capa de negocio será la responsable de comunicar la capa de presentación, responsable de la presentación de la información, con la capa de persistencia, responsable de almacenar los datos del sistema. De esta forma la capa de presentación realizará peticiones a la capa de presentación cada vez que necesite presentar información de los elementos móviles. Esta capa a su vez se comunicará con la capa de persistencia para que le provea de los datos almacenados en la base de datos del sistema.

Por otro lado la capa de negocio será la responsable de establecer las reglas que controlan la presentación de los elementos de seguridad realizada por la capa de presentación.

Toda la capa de negocio estará construida en PHP y en JavaScript. Dado que la lógica de nuestra aplicación es muy básica, esta capa se limitará a trasladar la información que le retorne la capa de integración a una estructura que sea reconocida por la capa de presentación. Con el código JavaScript controlaremos la presentación de los elementos en pantalla (capa de presentación). Con esto observamos que la capa de negocio aunque tiene su origen enteramente en servidor, se ejecutará parcialmente del lado del cliente.

Toda información que se transmite desde la capa de negocio a la capa de presentación se transmitirá en XML con una estructura definida. Para realizar esto, se define un archivo XML en el que se almacena toda la información necesaria para la localización y presentación de la información de los elementos de seguridad. Cuando el usuario realiza una petición, la capa de negocio pedirá a la capa de persistencia que le retorne la información consultada. Con esta información la capa de negocio construirá un XML con la información necesaria para que la capa de presentación pueda localizar y presentar la información solicitada por el usuario

A continuación presentamos la estructura que tendrá el archivo XML que se intercambiarán las capas de negocio y presentación:

```
<elementos>
        <elemento latitud='' longitud = '' icono=''>
        <codigo></codigo>
        <descricion></descricion>
        <URLImagen></URLImagen>
        <ruta>
                <punto orden='' latitud='' longitud=''/>
        </ruta>
        </elemento>
</elementos>
```

A continuación presentamos un ejemplo de datos en formato XML intercambiado entre la capa de negocio y la capa de presentación:

```
<elementos>
        <elemento latitud='42.88244238' lonxitud = '-8.52835715' icono='Recursos/camara.png'>
                <codigo><![CDATA[P001]]></codigo>
                <descricion><![CDATA[Patrulla 1]]></descricion>
                <URLImagen><![CDATA[http://www.youtube.com/v/3JQnIPv3n7s]]></URLImagen>
                <ruta>
                        <punto orden='1' latitud='42.88244238' longitud='-8.52835715'/>
                        <punto orden='2' latitud='42.88319708' longitud='-8.52789044'/>
                        <punto orden='3' latitud='42.88287083' longitud='-8.52684975'/>
                        <punto orden='4' latitud='42.88212398' longitud='-8.52734327'/>
                </ruta>
        </elemento>
        <elemento latitud='42.88319708' longitud = '-8.52789044' icono='Recursos/camara.png'>
                <codigo><![CDATA[C001]]></codigo>
                <descricion><![CDATA[Camara 1]]></descricion>
                <URLImagen><![CDATA[http://www.youtube.com/v/3JQnIPv3n7s]]></URLImagen>
        </elemento>
        <elemento latitud='42.88287083' longitud = '-8.52684975' icono='Recursos/camara.png'>
                <codigo><![CDATA[C002]]></codigo>
                <descricion><![CDATA[Camara 2]]></descricion>
                <URLImagen><![CDATA[http://www.youtube.com/v/3JQnIPv3n7s]]></URLImagen>
        </elemento>
        <elemento latitud='42.88212398' longitud = '-8.52734327' icono='Recursos/patrulla.png'>
                <codigo><![CDATA[C003]]></codigo>
                <descricion><![CDATA[Camara 3]]></descricion>
                <URLImagen><![CDATA[http://www.youtube.com/v/3JQnIPv3n7s]]></URLImagen>
        </elemento>
</elementos>
```

En los mapas suministrados por el servicio de Google se representarán los siguientes elementos: marcadores (elementos de seguridad), polilíneas (rutas), y ventanas de información. Cada uno de los elementos del XML se identifica con uno de estos elementos de representación.

Elemento	Descripción
`elementos`	Este elemento es obligatorio y sirve para englobar al resto de elementos. La especificación de XML obliga a que exista un único elemento raíz.
`elemento`	Contiene los datos de representación de una determinada entidad (videocámara o patrulla). El número de estos elementos es variable. Si no hubiese ninguno significa que no habría que representar nada en el mapa. Este elemento se representará como un marcador dentro del mapa. Tiene tres atributos obligatorios: • `latitud`: valor de la latitud de la coordenada geográfica del elemento. • `lonxitud`: valor de la longitud de la coordenada geográfica del elemento. • `icono`: ruta de localización del icono que representará a la entidad. Cada uno de estos elementos debe contar de forma obligatorio con un elemento `descricion`, `codigo` y `URLImagen`.
`codigo`	Es obligatorio y sólo se puede incluir bajo una etiqueta titulada `elemento`. Sólo puede haber un elemento `codigo`. No tiene atributos y su contenido será el código del elemento de seguridad.
`descricion`	Es obligatorio y sólo se puede incluir bajo una etiqueta titulada `elemento`. Sólo puede haber un elemento `descricion`. No tiene atributos y su contenido será la descripción asociada al elemento de seguridad.
`URLImagen`	Es obligatorio y sólo se puede incluir bajo una etiqueta titulada `elemento`. Sólo puede haber un elemento `URLImagen`. No tiene atributos y su contenido será la URL del servicio que provee las imágenes que emite el elemento de seguridad.
`ruta`	Opcional. Sirve para representar la ruta asociado al elemento en el que está incluido (válido para patrullas). No tiene atributos.
`punto`	Representa a uno de los vértices de la ruta. Es obligatorio contar con al menos dos de estos elementos dentro del elemento `ruta`. Tiene tres atributos obligatorios: • `orden`: número que sirve para ordenar el conjunto de puntos. • `latitud`: valor de la latitud de la coordenada geográfica del punto. • `longitud`: valor de la longitud de la coordenada geográfica del punto.

Tabla 4 Definición de XML SIGSE

6.4.3.3 Capa de persistencia

La capa de persistencia será la responsable de gestionar la persistencia de la información en la base de datos del sistema.

Toda la capa de persistencia estará construida en PHP [63]. La capa de persistencia tendrá operaciones para consultar los datos almacenados en la base de datos, con el objetivo de proveer datos a la capa de negocio.

Como gestor de base de datos hemos seleccionado MySQL [61]. Para la recuperación de los datos utilizamos el driver de MySQL disponible para PHP.

6.4.3.4 Diagrama de capas

A continuación presentamos un diagrama en el que se puede observar las capas de la aplicación y las conexiones existentes entre ellas. También presentaremos la integración que tiene la capa de presentación con el servicio de mapas de Google.

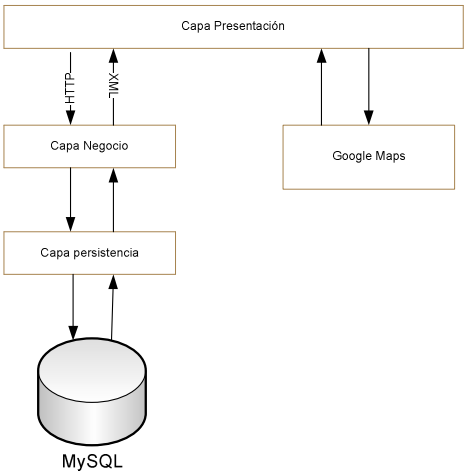

Figura 6.9 Capas del sistema

6.4.3.5 Secuencia de llamadas entre componentes

Como hemos comentado con anterioridad cada una de las capas tiene como misión resolver un problema particular de forma independiente al resto. Para conseguirlo es de suma importancia que todas las capas dispongan de interfaces claros de comunicación. Estos mecanismos de comunicación será los que utilizaremos para conseguir que las capas interaccionen entre ellas. Para presentar las interacciones entre los objetos de las distintas capas utilizamos un diagrama de secuencia en el que presentamos a los objetos y componentes involucrados en la secuencia que se nos pide y la capa a la que pertenecen.

El siguiente diagrama presenta las interacciones necesarias entre los objetos que componen cada una de las capas:

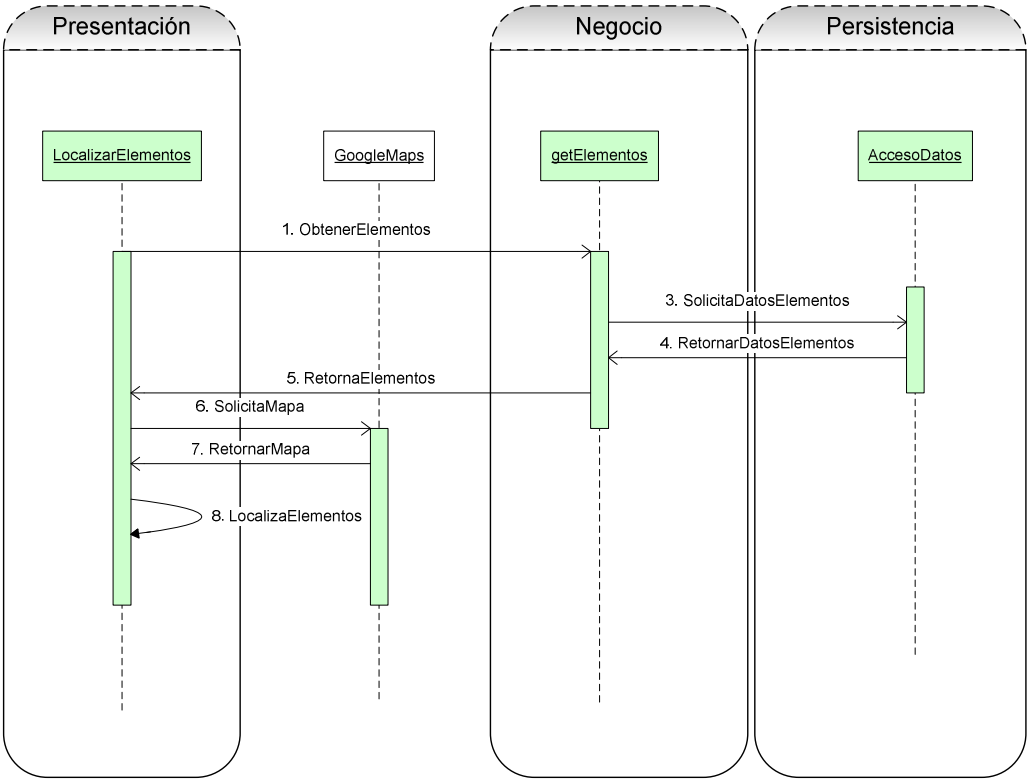

Figura 6.10 Diagrama de secuencia entre capas

En la secuencia se puede ver como una vez que el componente de la capa de presentación "*LocalizarElementos*" realiza una petición al componente de la capa de negocios "*getElementos*" para que de este actúe como fachada con la capa de persistencia solicitando a esta los datos necesarios para devolver los elementos de seguridad requeridos a la capa de presentación con sus posiciones geográficas. Una vez que la capa de presentación tiene los datos geográficos de los distintos elementos se realiza una petición al servicio externo de Google Maps para que retorne el mapa de la zona geográfica en la que se localizan los elementos de seguridad. Por último se posicionan los elementos de seguridad sobre el mapa utilizando para ello las coordenadas asociadas a cada uno de ellos.

6.5 Implementación

En los siguientes apartados comentaremos las decisiones tomadas en cada una de las partes en las que dividimos la aplicación. De esta forma presentaremos por un lado la implementación de la aplicación del lado del servidor web (capa de negocio, capa de negocio y capa de persistencia) y por otro lado la implementación del cliente (capa de presentación y capa de negocio). Como se puede observar la capa de negocio aunque se genera enteramente del lado del servidor, parte de ella se ejecuta en cliente (JavaScript de control de presentación de elementos). Por último haremos una breve exposición de la pequeña aplicación que hemos construido para la captura de las coordenadas que hemos utilizado para las pruebas del sistema.

Para cada sección abordaremos los siguientes apartados:

- **Decisiones de implementación**: Presentaremos los principales problemas que se han detectado y las decisiones que se han tomado a lo largo del desarrollo.
- **Interfaces**: Presentaremos las definiciones de las interfaces de la aplicación.
- **Funcionamiento**: Presentaremos una breve descripción de los procesos que se ejecutan y el funcionamiento operativo de la aplicación.
- **Código fuente**: Presentaremos la estructura del código que implementa cada uno de los componentes de la aplicación.

6.5.1 Aplicación Web

La aplicación del lado del servidor estará construida íntegramente en PHP. La aplicación web contendrá la capa de negocio, la capa de persistencia y parte de la capa de presentación (construcción de la ventana de información de los elementos). La base de datos se construirá sobre el gestor de base de datos MySQL 5.

6.5.1.1 Decisiones de implementación

Dado que la aplicación que tenemos que construir tiene una lógica de negocio y de persistencia muy pequeña hemos decidido integrarla en una única página PHP. De esta forma en **`getElementos.php`** tendremos tanto la lógica de acceso a base de datos como la lógica de negocio que determina las reglas que se aplican a las entidades del sistema.

La conexión con la base de datos está encapsulada un procedimiento denominado `ConectarBD`. En esta función se especifica el nombre del servidor, el nombre de la base de datos y el nombre del usuario con su clave de acceso.

6.5.1.2 Interfaces

El componente de servidor no dispone de interfaz de usuario. Desde el punto de vista de comunicación entre componentes, el cliente puede pasar distintos parámetros a la página `getElementos.php`. A continuación presentamos los parámetros admitidos:

- **tipo**: Indica el tipo de elementos a localizar.
- **bMostrarRuta**: Presentaremos las definiciones de las interfaces de la aplicación.

De acuerdo con lo expuesto mostramos un ejemplo de llamada: http://localhost/sigse/getElementos.php?tipo=2&bMostrarRuta=1.

Como hemos comentado en el apartado *6.4.3.2 Capa de negocio*, la respuesta que retornará el servidor a cualquier petición estará en formato XML de acuerdo a la estructura definida en dicho apartado del presente documento.

6.5.1.3 Funcionamiento

Todas las peticiones que se invocan desde distintas opciones disponibles en el interfaz del cliente. De esta forma tendremos una posible petición por cada uno de los casos de uso identificados en el apartado 6.3.2.

6.5.1.4 Código fuente

El código del servidor está estructurado en los siguientes ficheros:

- `getElementos.php`: Implementa toda la capa de persistencia de la aplicación. Por lo tanto es el encargado de recuperar de base de datos los elementos que se le solicitan (solicitados por parámetro) y retornar al cliente un XML con toda la información a presentar. Destacamos en el las siguientes funciones:
    - `ConectarBD`: Función responsable de crear la conexión con la base de datos.
    - `getSQLElemento`: Obtiene la consulta SQL que nos permitirá obtener los elementos solicitados por el usuario.
    - `getSQLRutaElemento`: Obtiene la consulta SQL que nos permitirá obtener la definición de la ruta del elemento consultado. Se le pasa por parámetro el identificador del elemento para el que se consulta la ruta.
- `utislUI.php`: Implementa una serie de funciones que encapsulan el código necesario para formar la pantalla de información de los elementos de seguridad. Destacamos en el las siguientes funciones:
    - `getHTMLHeader`: Función que obtiene la cabecera de la pantalla de información del elemento. Se le pasa por parámetro la información que se desea que aparezca en el título de la página.
    - `getHTMLFooter`: Función que obtienen el pie de la pantalla de información del elemento.
    - `getHTMLContent`: Cuerpo de información de la pantalla de información del elemento. El cuerpo estará compuesto por un objeto incrustado para visionado del video asociado a la cámara. A la función se le pasa la URL de servidor de imágenes. Para el prototipo hemos asociado a todos los elementos de seguridad un video del servicio de videos You Tube. En un caso real esta URL apuntaría al servidor de imágenes que nos retorne la captura de la videocámara seleccionada.
- `getInfoHTML.php`: Implementa parte de la capa de presentación de la aplicación. En concreto es responsable de retornar al cliente (a través de una petición AJAX) el HTML con el que formará la ventana de información asociada al elemento. Por lo tanto aglutina una serie de funciones que permiten formar el HTML de la ventana de información del elemento con los datos que le pasa el cliente por parámetro. Esta página incluye `utislUI.php` que es dónde se han encapsulado las funciones de construcción del HTML.

En los anexos de este documento (apartado 8.1) se puede consultar el código completo de las páginas comentadas anteriormente.

6.5.2 Cliente

El cliente de la aplicación está construido utilizando HTML (capa de presentación) y JavaScript (capa de negocio y presentación). Todo el aspecto gráfico está definido por dos hojas de estilo: una para el aspecto de la aplicación y otra para el aspecto de la pantalla de información de los elementos localizados en el mapa.

En las siguientes secciones explicamos las decisiones y detalles más representativos de la implementación.

6.5.2.1 Presentación personalizada de la información en el mapa

La API de Google Maps representa los marcadores, las ventanas de información, las polilíneas y demás elementos del mapa con una apariencia visual estándar. Esta apariencia puede ser personalizada para asemejarla lo máximo posible a la apariencia visual de la aplicación que haga uso de este servicio. Y esto es precisamente lo que se ha realizado para la integración con el sistema *SIGSE*. La personalización realizada de Google Maps abarcar los iconos de los marcadores, los polígonos y las ventanas de información.

Para personalizar los iconos de los marcadores, hay que crearlos como imágenes PNG con fondo transparente. Cada icono puede ir acompañado de una sombra, que en el caso de que todos los iconos tengan la misma forma sólo será necesario un archivo de imagen para ésta. Para establecer el icono de un marcador basta con pasar una instancia de la clase `GIcon`, que representa al icono deseado, al constructor del marcador:

```
var mapa = new GMap2(document.getElementById("mapa"));
var icono = new GIcon();
icono.image = "icono.png";
icono.iconSize = new GSize(20, 34);
icono.shadow = "sombra.png";
icono.shadowSize = new GSize(36, 34);
icono.iconAnchor = new GPoint(5, 34);
icono.infoWindowAnchor = new GPoint(5, 2);
var marcador = new GMarker(new GLatLng(42.87596410, -8.55834961), icono);
mapa.addOverlay(marcador);
```

Atributo	Descripción
`Image`	Ruta relativa o absoluta en la que se encuentra el archivo que contiene la imagen.
`iconSize`	Instancia de `GSize` que indica el tamaño en píxeles de la imagen.
`shadow`	Es opcional e indica la ruta relativa o absoluta en la que se encuentra el archivo que contiene la sombra de la imagen.
`shadowSize`	Instancia de `GSize` que indica el tamaño en píxeles de la sombra.
`iconAnchor`	Instancia de la clase `GPoint`, que indica el píxel de la imagen que coincidirá exactamente con la localización del punto geográfico.
`infoWindowAnchor`	Instancia de `GPoint`, que indica el píxel sobre el que se mostrará la ventana de información (infoWindow).

Tabla 5 Atributos de pantalla de información Google Maps

Para la representación de las rutas hemos simulado su representación a base de polilíneas. Estas polilíneas formarían unas zonas acotadas en el mapa que servirían para identificar a aquellas entidades de tipo “ruta”. Para poder mostrar al usuario la información asociada a esta zona o perímetro, se ha añadido un marcador, que al hacer clic sobre él abrirá una pequeña ventana con dicha información. La siguiente imagen muestra un ejemplo de esto.

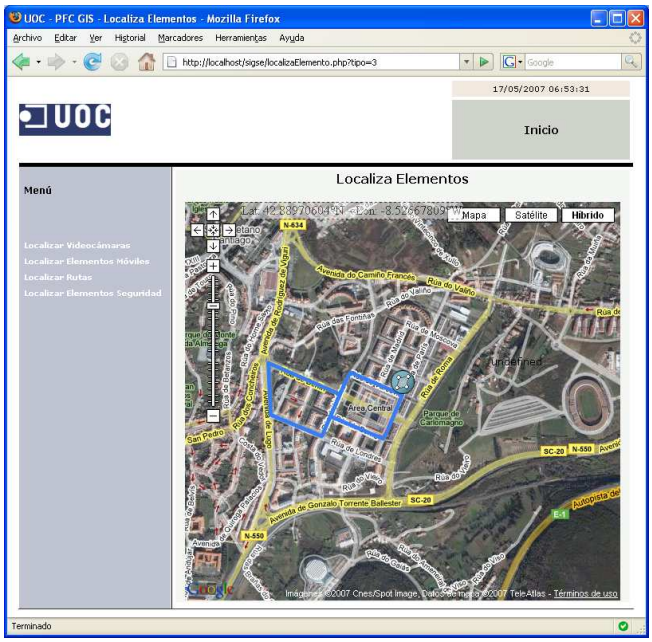

Figura 6.11 Localización de rutas SIGSE

Por último, se han personalizado las ventanas de información. En lugar de mostrar la clásica ventana con forma de bocadillo tipo cómic, se muestra un cuadro personalizado al estilo utilizado en toda la aplicación:

Figura 6.12 Ventana de información por defecto

Figura 6.13 Ventana de información personalizada

El código que maneja esta ventana personalizada se ha implementado como una clase de JavaScript (`VentanaInfo.js`). Esta clase extiende la clase `Overlay` de la API de Google Maps. Esta capacidad de crear "overlays" personalizados fue introducida en la versión 2 de la API. Un overlay es cualquier objeto representado en un mapa y que está asociado a una determinada coordenada geográfica. La clase `VentanaInfo` define los cuatro métodos requeridos por el interfaz `Overlay`: `initialize()`, que se encarga de crear la estructura DOM que representa al overlay; `remove()`, que elimina todos los elementos del overlay del mapa; `copy()`, para obtener una copia del overlay; y `redraw()`, que posiciona el overlay en el mapa teniendo en cuenta la coordenada geográfica del marcador al que está asociada. Todos los elementos DOM que forman el overlay `VentanaInfo`, se colocan en la capa `G_MAP_FLOAT_PANE`, es decir, en la capa más externa, siempre por encima de los marcadores y de las polilíneas. Si no fuese así, podría ocurrir la desagradable aparición de una ventana de información con un icono por encima o cortada por una poli línea. A continuación se muestra un ejemplo de cómo mostrar una ventana personalizada de tipo VentanaInfo.

```
var mapa = new GMap2(document.getElementById("mapa"));
var marcador = new GMarker(new GLatLng(42.87596410, -8.55834961));
mapa.addOverlay(marcador);
var html = ...<CODIGO HTML DE LA VENTANA>...;
GEvent.addListener(marcador, 'click', function() {
      ventanaInfo = new VentanaInfo(marker, html, "VentanaInfo")
      mapa.addOverlay(ventanaInfo)
});
```

El constructor de `VentanaInfo` toma tres argumentos:

- `marcador`: objeto de tipo GMarker, al que va asociada la ventana.
- `html`: código HTML para presentar en el interior de la ventana.
- `claseCss`: nombre de la clase CSS aplicada a la ventana.

Internamente, una `VentanaInfo` es una capa de HTML (elemento `div`), cuyo contenido es enviado a través de su constructor en forma de código HTML. En este código HTML se puede incluir casi cualquier elemento: imágenes, tablas, otras capas, etc., pero teniendo en cuenta que el tamaño de estas ventanas es muy limitado. El nombre del estilo CSS de dicho elemento `div` se obtiene también del constructor (argumento `claseCss`). El encargado de enviar el contenido de la ventana es el código JavaScript de la página que hace uso de `VentanaInfo`, y esta a su vez lo construirá en función de la información obtenida del XML (elementos `elemento` del XML) que recuperaremos del servidor mediante una petición AJAX.

6.5.2.2 Interfaces

Todos los interfaces están construidos con HTML y aplicando hojas de estilo que permiten cambiar el aspecto de los mismos de forma sencilla.

La aplicación se compone de una pantalla principal desde la que podemos acceder a las distintas opciones del sistema. A continuación presentamos la pantalla principal del sistema:

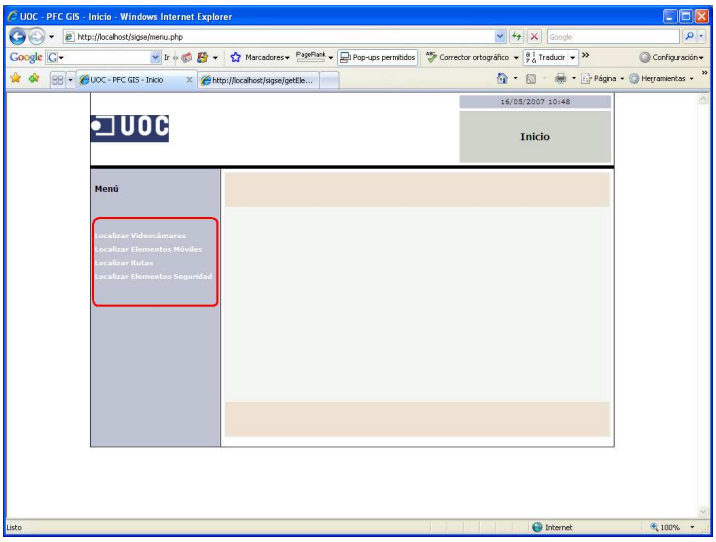

Figura 6.14 Menú SIGSE

En la zona izquierda de la pantalla se pueden observar las distintas opciones de la aplicación. Pinchando en las distintas opciones podremos visionar las páginas que localizan los elementos en los mapas que suministra el servicio de Google.

Todas las pantallas de localización de elementos tienen un aspecto similar. En la parte central derecha de la pantalla se presentará el mapa suministrado de Google, y en función de la opción selecciona se nos mostrarán los elementos de seguridad solicitados.

A continuación presentamos el aspecto de la pantalla de localización de elementos de seguridad:

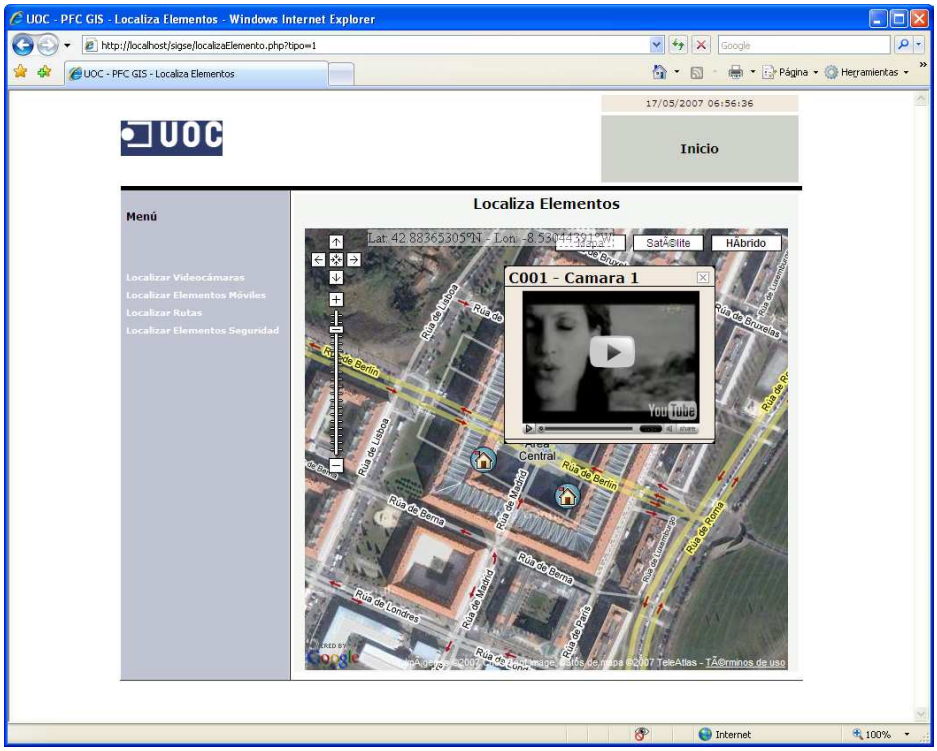

Figura 6.15 Pantalla de localización de elementos SIGSE

6.5.2.3 Funcionamiento

El usuario accede a la aplicación poniendo en su navegador la página inicial del sistema (http://localhost/sigse/index.php). Pulsando sobre el enlace inicio accedemos a la pantalla de menú del sistema que hemos presentado en la *figura 6.13*. Desde la pantalla de menú el usuario tiene acceso a todas las opciones del sistema: localizar videocámaras, localizar elementos móviles, localizar rutas o localizar todos los elementos de seguridad de la empresa.

Una vez que el usuario entra en la pantalla de localización puede regresar al menú principal (pulsando en el enlace superior de la pantalla) o acceder a otras funciones del sistema (el menú lateral). A continuación presentamos la pantalla de localización de elementos resaltando los accesos explicados anteriormente:

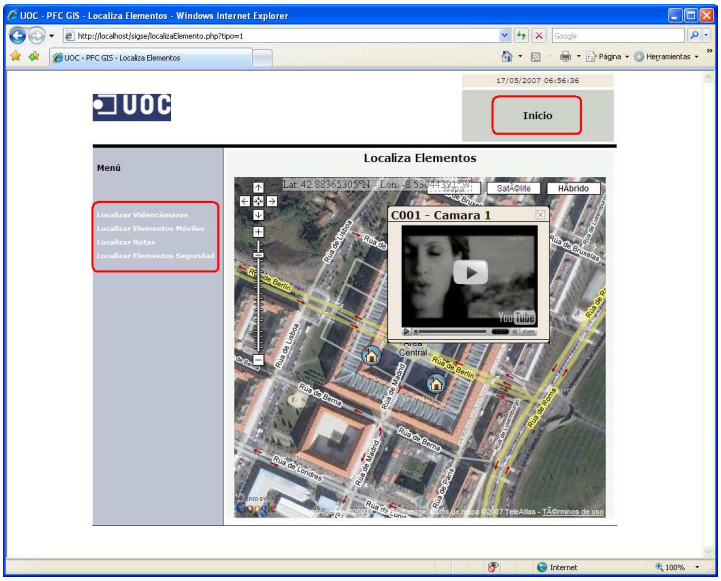

Figura 6.16 Ejemplo navegación en la aplicación SIGSE

6.5.2.4 Código fuente

El código fuente del cliente está implementado en dos ficheros.

- `VentanaInfo.js`: Implementa en JavaScript la clase `VentanaInfo`. Esta clase se encarga de gestionar la presentación de la capa con el contenido personalizado (ventana de información) asociada al elemento. Como hemos comentado con anterioridad, para inicializar la clase necesitamos indicarle los siguientes parámetros:
    - `marcador`: objeto de tipo GMarker, al que va asociada la ventana.
    - `html`: código HTML para presentar en el interior de la ventana.
    - `claseCss`: nombre de la clase CSS aplicada a la ventana.
- `localizaElemento.php`: Esta página construida en PHP y JavaScript es la responsable de presentar los mapas de localización y de posicionar los elementos de seguridad. A continuación identificamos los elementos estructurales más importantes de esta página:
    - El contenido HTML de la es muy sencillo y se limita a presentar la tabla del menú en la parte izquierda de la pantalla y el mapa en la parte central-derecha. En la parte inferior de la página se han incluido una serie de cajas de texto ocultas para facilitar la depuración de la página.
    - Del contenido JavaScript destacamos las siguientes secciones:
        - Al principio de la página incluimos el código JavaScript necesario para incorporar al navegador Firefox el soporte de las funciones de navegación por contenido XML selectNodes y selectSingleNode.
        - `Load`: Solicita el mapa al servicio de Google centrado en las coordenadas de interés para el sistema. También se encarga de invocar la carga de los elementos solicitados por el usuario.

- `cargaElementos`. Función que responsable de invocar a la capa de negocio solicitando los elementos de seguridad necesarios para responder a la solicitud del usuario.
- `engadeElementosXML`: Función responsable de procesar el XML recibido desde el servidor con la información de los elementos de seguridad a presentar.
- `engadeElemento`: Función responsable de posicionar un elemento concreto en el mapa solicitado a Google.

- `menu.php` e `index.php`: Estas páginas contiene el código HTML de la pantalla de inicio y del menú principal de la aplicación.

En los anexos de este documento (apartado 8.1) se puede consultar el código completo de las páginas comentadas anteriormente.

6.5.3 Aplicación para la recolección de coordenadas

Para la facilitar las pruebas del sistema hemos construido una pequeña página que utilizando los mapas de Google nos permite averiguar las coordenadas de los puntos que necesitemos. Para ello hemos construido una página en la que incluimos un mapa y un `textarea` en el que se nos retornan las coordenadas seleccionadas:

```
  <body onload="load()" onunload="GUnload()">
    <div id="map" style="width: 700px; height: 700px;align:center"></div>
    <div id="_coords" width="1px" height="1px" style="background: #ffffff; border: dashed 1px #000000; opacity: 0.50; filter:
alpha(opacity=50);; visibility: hidden;"></div>
    <input type="hidden" id="txtLoad" onclick="load()"/>
    <textarea id="txtCoordenadas" rows=4 cols=40></textarea>
  </body>
```

Para recoger las coordenadas realizamos una pequeña personalización sobre el funcionamiento de Google Map. En primer lugar creamos un `div` en el que posicionamos un cuadro en que se nos mostrarán las coordenadas a medida que movemos el ratón sobre el mapa:

```
    //muestra cuadro de coordenadas actules en la posición indicada
    function showCoordinates() {
                    //Obtemos controles
                    var coords = document.getElementById("_coords");
                    var google_map = document.getElementById("map");
                    google_map.appendChild(coords);


                    //Actualizamos coordenadas para tener una aproximación del ancho/alto del control coords
                    var center = mapa.getCenter();

                    //Actualizamos coordenadas
                    updateCoordinates(center.lat(), center.lng());

                    centerCoordinates();

    }
```

Con el siguiente código conseguimos enganchar el evento de movimiento del ratón con la actualización de las coordenadas:

```
          //añadimos manipulador de eventos para que actualice las coordenadas almover el raton
          GEvent.addListener(mapa, 'mousemove', function(point) {
                              updateCoordinates(point.lat().toFixed(8), point.lng().toFixed(8));
                    });
```

A continuación presentamos el código que actualiza las coordenadas en el `div` de coordenadas:

```
    //Actualiza las coodenadas que aparecen en el cuadro de coordenadas
    function updateCoordinates(lat, lon) {
                    var coords = document.getElementById("_coords");
                    coords.innerHTML = "Lat: " + lat + "&deg;N - Lon: " + lon + "&deg;W";
    }
```

Por último presentamos el manipulador que nos permite cambiar la acción asociada al evento de click sobre el mapa, de forma que llame a una función que vuelque las coordenadas actuales a la caja de texto creada para tal efecto:

```
          //añadimos manipulador de eventos para que ponga las coordenadas del punto sobre el que hacemos click en la caja de texto
          GEvent.addListener(mapa, 'click', function(marker, point) {
                              engadeCoordenadas(point.lat().toFixed(8), point.lng().toFixed(8));
                    });
```

A continuación mostramos la función que actualiza el contenido de la caja de coordenadas:

```
function engadeCoordenadas(lat, lon){
                var txt = document.getElementById("txtCoordenadas");
                if (!txt) {
                   return;
                }
                txt.value += lat + '\t' + lon + '\r\n';
}
```

6.5.3.1 Funcionamiento

Para obtener las coordenadas de las posiciones que necesitamos para la aplicación sólo tenemos que abrir con un navegador la página `RecolectorCoordenadas.htm`. Una vez que se nos presenta el mapa de Google sólo tendremos que desplazarnos sobre hasta localizar las posiciones que son de nuestro interés. Para obtener la posición de un punto determinado sólo tenemos que hacer click con el ratón sobre el mismo. El JavaScript de la página se encargará de ponernos en la zona inferior de la pantalla las coordenadas separadas por tabuladores (para facilitar su posterior importación). En la figura 5.3 podemos observar el aspecto del recolector de coordenadas.

6.6 Requisitos hardware y software

En el siguiente apartado determinaremos los requisitos hardware y software necesarios tanto para la fase de construcción del sistema como para la fase de puesta en explotación del mismo.

6.6.1 Entorno de desarrollo

Para la construcción del sistema ha sido necesario contar con un equipo informático con las siguientes características técnicas: Procesador Pentium IV con 2 GB de memoria RAM y con un espacio disponible en disco duro de al menos 100 MB.

A mayores ha sido necesario disponer de una conexión Internet que ha posibilitado la consulta de las distintas referencias bibliográficas y la realización de las pruebas de peticiones al servicio Google Maps.

Desde el punto de vista del software han sido necesarias las siguientes herramientas:

- Servidor Web Apache 2. Utilizado para las pruebas de la aplicación web.
- Servidor de base de datos MySQL 5 y la configuración necesaria en PHP para el acceso al gestor de base de datos.
- Editor de texto Notepad++ para la codificación de los ficheros de la aplicación.
- Microsoft Word para la construcción de toda la documentación del sistema.
- Microsoft Project para la construcción de los diagramas de Gantt.
- Microsoft Visio para la construcción de los diagramas utilizados en la fase de análisis y diseño de la aplicación.

6.6.2 Entorno de explotación

Para una correcta instalación de la aplicación es necesario disponer de un servidor que cuente de los siguientes elementos software:

- Servidor web Apache 2 configurado para la interpretación de PHP 5.
- Servidor de base de datos MySQL 5 y la configuración necesaria en PHP para el acceso al gestor de base de datos.

Como requisitos hardware del servidor consideramos que para la instalación de la versión actual de SIGSE es suficiente con un PC estándar con las siguientes especificaciones mínimas (para una puesta en explotación de la aplicación sería necesario aumentar las especificaciones en función del volumen de datos y del número de usuarios):

- Procesador Pentium IV o equivalente
- 512 MB de memoria RAM
- 100 MB de espacio libre en disco

- Tarjeta de red Ethernet
- Sistema operativo Windows XP o Linux

La máquina cliente que utilizarán los usuarios de la aplicación tiene que ser un ordenador personal que disponga de un navegador Web y conexión a Internet.

7 Conclusiones

La utilización de las especificaciones establecidas por el OGC abre una nueva puerta para la construcción de GIS y la colaboración entre distintos sistemas. Nos encontramos en pleno auge de los sistemas interconectados, la distribución de sistemas y de información, y las especificaciones de OGC sientan la base para que todos los GIS que se desarrollen de acuerdo a ellos puedan interconectarse y compartir información.

Dentro de las especificaciones definidas por el OGC destacamos dos por su importancia y grado de implantación en el mundo de los GIS:

- Web Map Service (WMS): Estos servicios web son especialmente útiles para proveer mapas en formato imagen para que pueda ser consultada por otros SIG u otros sistemas de información.
- Web Feature Service (WFS): Este tipo de servicios son adecuados para proveer mapas en formato vectorial, posibilitando la georreferenciación de información de forma precisa en el espacio. Algunos tipos de WFS (transaccionales) permiten la edición de información, dejando de ser simples servicios de consulta a servicios que permiten gestionar la información georreferenciada.

Dentro de los servicios de mapas estudiamos el caso del Google Maps. Este servicio ofrecido por Google (que no cumple las especificaciones de WMS) permite de forma sencilla la localización de información sobre mapas o combinaciones de imágenes fotográficas y mapas. Gracias a la publicación de la API de programación de Google Maps, cualquier sistema puede utilizar los servicios de Google Maps para presentar información de cualquier tipo sobre los mapas ofrecidos por el servicio.

En todo caso tenemos que destacar que Google Maps es una aplicación comercial de Google cuyo objetivo final es construir un sistema con el mayor número de usuarios sobre los cuales explotar sus servicios de publicidad (fuente de ingresos de la compañía). Para ello (conseguir mayor número de clientes) está realizando fuertes inversiones encaminadas en ofrecer cada vez más funcionalidades sobre su servicio de mapas (mayor precisión cartográfica, imágenes a pie de calle, consulta de tráfico, etc.).

Después de estudiar los distintos brevemente los GIS y las especificaciones OpenGIS, creemos que Google Maps puede ser un servicio adecuado para construir el sistema que permita la gestión en el territorio de una empresa de seguridad (objetivo práctico de este proyecto).

En general podemos concluir que Google Maps es una perfecta infraestructura para incorporar a sistemas de información la capacidad de presentar información localizada sobre el territorio (mapas o fotos), siempre y cuando no busquemos crear presentaciones complejas. Destacamos que las funcionalidades de representación que podemos añadir a nuestros sistemas con la versión actual de la API de Google son limitadas, aunque suficientes para la mayoría de los sistemas (incorporación de puntos de información, creación de líneas sobre el mapa, etc.).

7.1 Líneas futuras de trabajo

Podrían efectuarse ampliaciones en el proyecto en diversos ámbitos, básicamente orientadas a completar el prototipo para una empresa de seguridad real:

- **Completar el modelo**: añadiendo todas las entidades que son necesarias para la gestión de una empresa de seguridad: empleados de seguridad, incidencias, partes de trabajo, turnos, etc.
- **Tecnología**: realizando un uso real de GPS y GPRS. Para ello sería necesario implementar un servicio de actualización de posiciones. Por otro lado también sería interesante disponer de conexión directa con las cámaras de vigilancia a través del protocolo de comunicaciones IP. Desde el punto de vista de la aplicación sería interesante cambiar el intercambio de datos que se realiza en XML por intercambio en JSON. Este último formato es mucho más óptimo que XML, ya que reduce el tamaño de los datos transmitidos.
- **Conectar con un servicio WMS**: dado que Google Maps es un servicio propietario con ciertas restricciones de licencia, sería muy interesante para futuras versiones del sistema sustituir las

peticiones del sistema al servidor de mapas de Google por peticiones a un servidor WMS estándar que provea mapas de la zona de interés para nuestro sistema.

- **Incorporar nuevas funcionalidades**: la localización de los elementos de seguridad puede ser una gran herramienta de análisis para los gestores de la empresa. Para ello se podrían crear funcionalidades que permitieran analizar sobre el mapa parámetros como:
    - Número incidencias: detección de zonas conflictivas.
    - Desplazamientos de las patrullas: optimización de los recursos móviles reduciendo desplazamientos.
    - Localización de servicios más costosos (desplazamientos, recursos, etc.).

8 Anexo

8.1 Código fuente - servidor

A continuación mostramos el código de la página `getElementos.php`:

```
<?php
/*******************************************************************************/
/* getElementos.php                                                          */
/*                                                                             */
/* Esta página dará respuesta a la petición del cliente para que le retorne los*/
/* datos de los elementos de seguridad solicitados por parámetro*/
/* Parámetros:  */
/*  -TIPO: Tipo de elemento solicitado: */
/*             1 - Videocámaras                                              */
/*             2 - Patrullas                                                  */
/*             nada - Todos los elementos*/
/*  -bMostrarRuta: En los casos en los que se quiere representar la ruta*/
/*                                                                             */
/*******************************************************************************/

    //Configuramos para que no cahee la página
    header("Cache-Control: no-cache, must-revalidate"); // HTTP/1.1
    header("Expires: Mon, 26 Jul 1997 05:00:00 GMT"); // fecha pasada

    /**
      * Función que realiza la conexión con la base de datos
      */
    function ConectarBD()
    {
        if (!($link=mysql_connect("localhost","USUARIO","CLAVE")))
        {
            //error en la conexión con el servidor
            echo "Error conectando a la base de datos.";
            exit();
        }
        if (!mysql_select_db("ejbtest",$link))
        {
            //error con la selección de la base de datos
            echo "Error seleccionando la base de datos.";
            exit();
        }
        return $link;
    }

    /**
      *generamos SQL que recupera la información de los elementos
      */
    function getSQLElemento()
    {
        //Abrimos los elementos y su última posición
        $strSELECT = "select e.Id, e.Codigo as Codigo, e.idTipo, e.Descripcion, e.URLImagen, p.Latitud, p.Longitud, p.Fecha, p.Hora";
        $strFROM = " from elementos e inner join posiciones p on e.id = p.idelemento";
        if(isset($_GET['tipo']))
        {
            $strWHERE = " where idTipo = ".$_GET['tipo'];

            if ($_GET['tipo'] == 2)
            {
                $strWHERE = $strWHERE." And second(hora) <= ".date("s");
                $strORDER = " ORDER BY Hora DESC LIMIT 1";
            }
            else
            {
                $strORDER = "";
            }
            $strSQL = $strSELECT.$strFROM.$strWHERE.$strORDER;
        }
        else
        {
            $strWHERE = " Where (idTipo = 2 And second(hora) <= ".date("s").")";
            $strORDER = " ORDER BY Hora DESC LIMIT 1";
            $strSQL = "SELECT * FROM (".$strSELECT.$strFROM.$strWHERE.$strORDER.") as Ele1";
            $strWHERE = " where idTipo = 1";
            $strSQL = $strSQL." UNION ".$strSELECT.$strFROM.$strWHERE;
        }

        return $strSQL;
    }

    /**
      *generamos SQL que recupera la información de la ruta de un elemento
      */
    function getSQLRutaElemento($lidElemento)
    {
        //Abrimos los elementos y su última posición
        $strSELECT = "SELECT c.*";
        $strFROM = " FROM RUTAS r INNER JOIN CoordenadasRutas c on r.id = c.idRuta";
        $strWHERE = "  WHERE r.idElemento = ".$lidElemento;
        $strORDER = " ORDER BY Orden";
        $strSQL = $strSELECT.$strFROM.$strWHERE.$strORDER;
```

```
        return $strSQL;
    }

    //Nos conectamos a la base de datos
    $link=ConectarBD();

    //Recuperamos la información de los elementos
    $rsElementos=mysql_query(getSQLElemento(),$link);

    //Construimos la cabecera del XML
    $strXML = "<?xml version=\"1.0\" encoding = \"utf-8\" ?>";
    $strXML = "<elementos>";

    while($row = mysql_fetch_array($rsElementos)) {

        //decidimos el icono que presentamos
        if(isset($_GET['bMostrarRuta']))
        {
            $strRutaIcono="images/ruta.png";
        }
        else
        {
            if ($_GET['tipo'] == 2)
            {
                $strRutaIcono="images/info.png";
            }
            else
            {
                $strRutaIcono="images/camara.png";
            }
        }

        //describimos el elemento actual
        $strXML = $strXML."<elemento latitud='".$row["Latitud"]."' longitud = '".$row["Longitud"]."' icono='".$strRutaIcono."'>";
        $strXML = $strXML."<codigo><![CDATA[".$row["Codigo"]."]]></codigo>";
        $strXML = $strXML."<descricion><![CDATA[".$row["Descripcion"]."]]></descricion>";
        $strXML = $strXML."<URLImagen><![CDATA[".$row["URLImagen"]."]]></URLImagen>";

        //Comprobamos si tenemos que presentar las rutas
        if(isset($_GET['bMostrarRuta']))
        {
            if ($_GET['bMostrarRuta']==1)
            {
                //recuperamos las rutas
                $rsRutas=mysql_query(getSQLRutaElemento($row["Id"]));

                //formamos la parte del XML que describe la ruta
                $strXML = $strXML."<ruta>";
                while($rowRutas = mysql_fetch_array($rsRutas)) {
                    $strXML = $strXML."<punto orden='".$rowRutas["Orden"]."' latitud='".$rowRutas["Latitud"]."' longitud =
'".$rowRutas["Longitud"]."'></punto>";
                }
                $strXML = $strXML."</ruta>";

                //cerramos la conexión con la BD
                mysql_free_result($rsRutas);
            }
        }

        $strXML = $strXML."</elemento>";

        $strDEBUG = $row["Latitud"];
    }

    //cerramos el XML
    $strXML = $strXML."</elementos>";

    echo $strXML;

    //cerramos las conesiones con la BD
    mysql_free_result($rsElementos);
    mysql_close($link);
?>
```

No incluimos el código de la página `utils.php` y `getInfoHTML.php` por limitaciones de espacio y porque su implementación es relativamente sencilla. Se puede consultar el código fuente completo en los ficheros adjuntos a esta memoria.

8.2 Código fuente - cliente

A continuación mostramos el código empleado para la creación de los interfaces de la aplicación. En primer lugar presentamos el código JavaScript de la clase `VentanaInfo`:

```
/*****************************************************************************/
/* VentanaInfo.js                                                            */
/*                                                                           */
/* VentanaInfo es un Overlay personalizado para mostrar en un mapa de Google  */
/* Maps. VentanaInfo toma como entrada un cadena de caracteres que contiene   */
/* un código HTML.                                                           */
/*                                                                           */
/*****************************************************************************/
```

```
function VentanaInfo(marcador, html, claseCss)
  {
   this.marcador = marcador;
   this.html     = html;
   this.claseCss = claseCss;
   this.mapa     = null;
   this.div      = null;
  }

VentanaInfo.prototype = new GOverlay();

/**
  * Método ejecutado por el mapa cuando el Overlay se añada al propio mapa.
  */
VentanaInfo.prototype.initialize = function(mapa)
  {
   var div;

   //La ventana será una capa.
   div = document.createElement("div");

   //Establecer los estilos de la ventana.
   div.className      = this.claseCss;
   div.id             = 'VentanaInfo';

   //Establecer el contenido de la ventana.
   div.innerHTML      = this.html;

   //Impedir que los eventos sobre el div vayan a parar al mapa.
   GEvent.addDomListener(div, "click"    , stopPropagation);
   GEvent.addDomListener(div, "mousedown", stopPropagation);
   GEvent.addDomListener(div, "dblclick" , stopPropagation);

   //La ventana se mostrará en el mismo panel que las ventanas InfoWindow de
   //Google, es decir, en la parte más superficial.
   mapa.getPane(G_MAP_FLOAT_PANE).appendChild(div);

   //Almacenar el mapa y la capa en las propiedades del objeto.
   this.mapa = mapa;
   this.div  = div;
  }

/**
  * Método llamado por el mapa cuando el overlay se tiene que eliminar.
  */
VentanaInfo.prototype.remove = function()
  {
   this.div.parentNode.removeChild(this.div);
  }

/**
  * Devuelve una copia no inicializada de sí mismo.
  */
VentanaInfo.prototype.copy = function()
  {
   return new VentanaInfo(this.marcador, this.html, this.claseCss);
  }

/**
  * Método invocado por el mapa cuando cambie su visualización. El argumento
  * forzado será true cuando cambie el sistema de coordenadas, por lo que
  * hay que redibujar el Overlay.
  */
VentanaInfo.prototype.redraw = function(forzado)
 {
  var c1;
  var c2;
  var off;
  var pix;
  var bounds;
  var desplazamiento;
  var punto;
  var icono;

  //Si el usuario no hace ni zoom ni mueve el mapa, entonces
  //no se vuelve a repintar.
  if (!forzado)
    {
     return;
    }

  //Recuperar el punto y el icono asociado al marcador.
  punto = this.marcador.getPoint();
  icono = this.marcador.getIcon();

  //Obtener las coordenadas del marcador.
  pix = this.mapa.fromLatLngToDivPixel(punto);
  off = new Array();
  off[1] = pix.y;
  off[0] = pix.x;

  //Desplazar la capa a la derecha por la mitad del tamaño del icono más un
  //pequeño margen de 2 píxeles.
  off[0] += (icono.iconSize.width/2 + 2);

  //Desplazar verticalmente la capa por la altura de esta.
  off[1] -= this.div.offsetHeight;

  //Desplazar verticalmente la capa por el tamaño del icono más un pequeño
  //margen de 2 píxeles.
  off[1] -= (icono.iconSize.height + 2);
```

```
  //Colocar la capa en la posición adecuada.
  this.div.style.left   = off[0] + 'px';
  this.div.style.top    = off[1] + 'px';

  //Obtener las dimensiones del mapa.
  var bounds = mapa.getBounds();
  var c1 = mapa.fromLatLngToDivPixel(bounds.getSouthWest());
  var c2 = mapa.fromLatLngToDivPixel(bounds.getNorthEast());

  //Averiguar si hay que desplazar el mapa para que se pueda ver la capa.
  desplazamiento = new Object();
  desplazamiento.x = 0;
  desplazamiento.y = 0;
  if (off[0] + this.div.offsetWidth > c2.x)
    {
     //Desplazar a la derecha. El 5 es para que no coincida exactamente con
     //el borde del mapa.
     desplazamiento.x = c2.x - off[0] - this.div.offsetWidth - 5;
    }
  if (off[1] < (c2.y + 30))
    {
     //Desplazar hacia arriba. El 30 es para que la capa no coincida con el
     //control del tipo de mapa.
     desplazamiento.y = c2.y - off[1] + 30;
    }

  //Desplazar el mapa.
  if ((desplazamiento.x != 0) || (desplazamiento.y != 0))
    {
     mapa.panBy(new GSize(desplazamiento.x, desplazamiento.y));
    }
 }

 /**
   * Función de uso interno para impedir la propagación de los eventos del
   * overlay sobre el mapa.
   */
function stopPropagation(event)
  {
   var type = event ? event.type : window.event.type;
   var phase = event ? event.eventPhase : "unknown";

   if (window.event) // IE
     {
                  window.event.cancelBubble = true;
     }
   else // no IE
     {
      if (event.eventPhase == 1)
        {
         window.alert("error: stop_propogation used for capturing phase");
        }
      else
        {
         event.stopPropagation();
        }
     }
  }
```

A continuación mostramos parcialmente el código de la pantalla de localización de los elementos (`localizaElemento.php`). Dada la sencillez del código HTML de la página, hemos decidido presentar sólo el contenido JavaScript de la misma. Se puede consultar el código fuente completo en los ficheros adjuntos a esta memoria.

```
<?php
/*****************************************************************************/
/* localizaElemento.php                                                      */
/*                                                                           */
/* Esta página presenta los elementos solicitados en pantalla                */
/* Parámetros:                                                               */
/*  -TIPO: Tipo de elemento solicitado:                                      */
/*              1 - Videocámaras                                             */
/*              2 - Patrullas                                                */
/*              3 - Rutas                                                    */
/*      nada - Todos los elementos                                           */
/*                                                                           */
/*****************************************************************************/
    //Configuramos para que no cahee la página
    header("Cache-Control: no-cache, must-revalidate"); // HTTP/1.1
    header("Expires: Mon, 26 Jul 1997 05:00:00 GMT"); // fecha pasada
?>

......

    <script type="text/javascript">
    //<![CDATA[
    var mapa;
    var ventanaInfo;

    function load() {

        //Comprobamos si ya tenemos cargado el mapa
        if (!mapa) {
            if (GBrowserIsCompatible()) {
                mapa = new GMap2(document.getElementById("map"));
                mapa.addControl(new GLargeMapControl());   //Muestra botones de desplazamiento y nivel de zoom
```

```
                mapa.addControl(new GMapTypeControl());     //Muestra tipo de mapa: normal, satélite, híbrido.
                //habilitamos scroll con rueda de ratón
                mapa.enableScrollWheelZoom();

                //Posicionamos el mapa aproximadamenteo en el centro de Galicia: 42.88266250       -8.52769732
                mapa.setCenter(new GLatLng(42.88266250, -8.52769732), 17);

                mapa.checkResize();

                //mostramos coordenadas
                showCoordinates();

                //añadimos manipulador de eventos para que actualize las coordenadas al mover el ratón
                GEvent.addListener(mapa, 'mousemove', function(point) {
                    updateCoordinates(point.lat().toFixed(8), point.lng().toFixed(8));
                });

                //Establecemos el tipo de mapa
                mapa.setMapType(G_HYBRID_MAP);

                //notificamos que remato la carga
                document.getElementById('txtDocumentState').value = "Loaded";

            }
        }
        //invocamos la carga de los elementos
<?php
    if(isset($_GET['tipo']))
    {
        if($_GET['tipo']==1 OR $_GET['tipo']==3)
        {
            //cargamos los datos de los elementos
            echo "cargaElementos();";
        }
        else
        {
            //cargamos los datos a intervalos de tiempo de 3 segundos
            echo "setInterval(cargaElementos, 3000);";
        }
    }
    else
    {
        //cargamos los datos a intervalos de tiempo de 3 segundos
        echo "setInterval(cargaElementos, 3000);";
    }
?>
    }

    //muestra cuadro de coordenadas actuales en la posición indicada
    function showCoordinates() {
       //Obtemos controles
       var coords = document.getElementById("_coords");
       var google_map = document.getElementById("map");
       google_map.appendChild(coords);

       //Actualizamos coordenadas para tener una aproximación del ancho/alto del control coords
       var center = mapa.getCenter();

       //Actualizamos coordenadas para que se dibuje el <DIV>
       updateCoordinates(center.lat(), center.lng());

       centerCoordinates();
      }

    function centerCoordinates() {
       var coords = document.getElementById("_coords");

       coords.style.zIndex     = 100;
       coords.style.position   = 'absolute';
       coords.style.top        = "0px";
       coords.style.left       = "70px";
       coords.style.visibility = 'visible';
      }

    //Actualiza las coodenadas que aparecen en el cuadro de coordenadas actuales
    function updateCoordinates(lat, lon) {
       var coords = document.getElementById("_coords");
       coords.innerHTML = "Lat: " + lat + "&deg;N - Lon: " + lon + "&deg;W";
      }

    //función que añade un elemento
    function engadeElemento(latitude, lonxitude, descricion, iconPath) {
        var point = new GLatLng(latitude, lonxitude);

        //Crear el marcador.
        var icon = getIcono(iconPath);
        var marker = new GMarker(point, icon);

        if (!mapa) {
            window.alert('No se cargo el mapa en engadeElemento.');
        }

        mapa.addOverlay(marker);

        //añadimos el enganche a la pantalla personalizada
        GEvent.addListener(marker, 'click', function() {
            //marker.openInfoWindowHtml("prueba");
            if (ventanaInfo != null)
                {
                ventanaInfo.remove();
                }
            ventanaInfo = new VentanaInfo(marker, descricion, "VentanaInfo");
```

```
            mapa.addOverlay(ventanaInfo);
            }
        );
    }

    //Actualiza el encuadre
    function updateBounds(bounds, latLng)
    {
        bounds.extend(latLng);
    }

    //añadimos los elementos desde el XML
    function engadeElementosXML(strXML) {

        //antes de pintar nuevos elementos, eliminamos los antiguos.
        mapa.clearOverlays();

        //debug();

        //creamos encuadre vacío
        var bounds = new GLatLngBounds();

        //convertimos string con XML en documento DOM
        var xml = GXml.parse(strXML);
        if (!xml.documentElement) {
            window.alert('La cadena XML con los elementos no es correcta. Asegurese de que no contiene caracteres no permitidos.
Cadena: ' + strXML);
            return;
        }

        //obtemos los elementos del XML
        var elementos = xml.documentElement.getElementsByTagName("elemento");

        //recorremos los elementos
        for (var iCounterElementos=0; iCounterElementos<elementos.length; iCounterElementos++) {
            var latitude = parseFloat(elementos[iCounterElementos].getAttribute("latitud"));
            var lonxitude = parseFloat(elementos[iCounterElementos].getAttribute("longitud"));
            var icono = elementos[iCounterElementos].getAttribute("icono");
            var codigo = elementos[iCounterElementos].childNodes[0].firstChild.nodeValue;
            var descricion = elementos[iCounterElementos].childNodes[1].firstChild.nodeValue;
            var URLImagen = elementos[iCounterElementos].childNodes[2].firstChild.nodeValue;

            var descricionHTML = requestSincrono("getInfoHTML.php?codigo=" + codigo + "&descricion=" + descricion + "&URLImagen=" +
URLImagen);

            //debug();
            engadeElemento(latitude, lonxitude, document.getElementById('txtInfoHTML').value, icono);

            //actualizamos encuadre
            updateBounds(bounds, new GLatLng(latitude, lonxitude));

            //recorremos las rutas
            var rutas = elementos[iCounterElementos].getElementsByTagName("ruta");

            for (var iCounterRutas=0; iCounterRutas<rutas.length ; iCounterRutas++)
            {
                //establecemos el aspecto de la ruta
                var color     = '#397df3';
                var ancho     = 5;
                var opacidade  = 1;

                //recuperamos los puntos de la ruta
                var puntos    = rutas[iCounterRutas].getElementsByTagName("punto");

                //crea un array con los puntos que forman la polilínea con la que representamos la ruta.
                var pts    = [];

                //recorremos los puntos de las rutas
                for (var j=0; j<puntos.length; j++)
                {
                    pts[j] = new GLatLng(parseFloat(puntos[j].getAttribute("latitud")),
parseFloat(puntos[j].getAttribute("longitud")));

                    //actualizamos encuadre
                    updateBounds(bounds, pts[j]);
                }
                //debug();

                //crea a polilínea
                var polilinea = new GPolyline(pts, color, ancho, opacidade);

                mapa.addOverlay(polilinea);
            }
        }

        //actualizamos encuadre para que muestre resultados
        setExtendedBounds(bounds);
    }

    //control de resize
    function invokeCheckResize() {
        if (!mapa) {
            window.alert('No se cargo mapa en invokeCheckResize.');
        }
        else {
            mapa.checkResize();
            centerCoordinates();
        }
    }

    //establecemos bordes con margen
    function setBounds(bounds)
```

```
{
    //Estabecemos el centro del mapa en el centro del encuadre
    mapa.setCenter(bounds.getCenter());

    //debug();

    //Nivel máximo de zoom antes de que se deje de ver imagenes = 18
    var zoomLevel;
    zoomLevel = mapa.getBoundsZoomLevel(bounds);

    if (zoomLevel>18) zoomLevel = 18;

    //Establecemos nivel de zoom que permita ver todo el rectangulo
    mapa.setZoom(zoomLevel);
}

//añadimos bounds con margen
function setExtendedBounds(bounds)
{
      var center = bounds.getCenter();

    //Ampliamos rectangulo de encuadre para que no queden elementos pegados a los bordes del mapa
    var sw = bounds.getSouthWest();
    var ne = bounds.getNorthEast();
    var latDelta = center.lat() - sw.lat(); //Diferencia entre el centro y el borde superior
    var lngDelta = center.lng() - sw.lng(); //Diferencia entre el centro y el borde oeste
    var boundsExtended = new GLatLngBounds(sw, ne);

    //ampliamos suroeste
    boundsExtended.extend(new GLatLng(sw.lat()-latDelta, sw.lng()-lngDelta));

    //ampliamos noreste
    boundsExtended.extend(new GLatLng(ne.lat()+latDelta, ne.lng()+lngDelta));

    setBounds(boundsExtended);
}

//Método que se ejecuta cuando el usuario pulsa sobre la X de la ventana de
//información para cerrarla.
function cerrarVentanaInfo() {
    if (ventanaInfo != null)
    {
        ventanaInfo.remove();
        ventanaInfo = null;
    }
}

//Devuelve un objeto de tipo GIcon a partir de un nombre de tipo de entidad.
function getIcono(iconPath, shadowPath)
{
    var icon = new GIcon();
    icon.image = iconPath;

    //se no nos pasaron imagen de la sombra, ponemos icono por defecto
    if (!shadowPath) {
        icon.shadow = shadowPath;
    } else {
        icon.shadow     = "recursos/sombra.png";
    }

    var imgIcon = new Image();
    imgIcon.src = iconPath;

    var imgShadow = new Image();
    imgShadow.src = icon.shadorw;

    //debug();
    //cargamos Imagen para ter ancho y alto
    icon.iconSize   = new GSize(imgIcon.width, imgIcon.height);
    icon.shadowSize = new GSize(imgShadow.width, imgShadow.height);

    icon.iconAnchor = new GPoint(imgIcon.width/2, imgIcon.height);
    icon.infoWindowAnchor = new GPoint(imgIcon.width/2, imgIcon.height/2)

    return icon;
}

//función que crea el objeto para realizar peticiones XMLHTTP en función de las capacidades del navegador
function getHTTPObject() {
    var xmlhttp;
    if (window.XMLHttpRequest) { // Mozilla/Safari/IE7+
        xmlhttp = new XMLHttpRequest();
    } else if (window.ActiveXObject) { // IE6-
        xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
    }
    return xmlhttp;
}

//petición síncrona al servidor
function requestSincrono(URL) {
    var http = getHTTPObject();
    http.open("GET", URL, false);
    http.onreadystatechange = function() {
        if (http.readyState == 4) {
            //retornamos la respuesta
            document.getElementById('txtInfoHTML').value = http.responseText;
            return document.getElementById('txtInfoHTML').value;
        }
    }
    http.send(null);
}
```

```
    //petición asíncrona de los elementos a representar
    function request(URL) {
        var http = getHTTPObject();
        http.open("GET", URL, true);
        http.onreadystatechange = function() {
            if (http.readyState == 4) {
                //metemos en la caja de texto los elementos.
                document.getElementById('txtXML').value = http.responseText;
                //cargamos los elementos
                engadeElementosXML(document.getElementById('txtXML').value);
            }
        }
        http.send(null);
    }

    //función que carga los elementos llamando a la URL adecuada
    function cargaElementos() {
        var strXML = "";
<?php
    if(isset($_GET['tipo']))
    {
        if($_GET['tipo']==3)
        {
            echo "request(\"getElementos.php?tipo=2&bMostrarRuta=1\");";
        }
        else
        {
            echo "request(\"getElementos.php?tipo=".$_GET['tipo']."\");";
        }
    }
    else
    {
        echo "request(\"getElementos.php\");";
    }
?>
    }

    //]]>
</SCRIPT>
```

9 Glosario

AJAX: Siglas de *Asynchronous JavaScript And XML* (JavaScript y XML asíncronos). Es una técnica de desarrollo web para crear aplicaciones interactivas. Éstas se ejecutan en el cliente, es decir, en el navegador del usuario, y mantiene comunicación asíncrona con el servidor en segundo plano.

API: Siglas de *Application Program Interface* (interfaz de programación). API define un conjunto de reglas y bibliotecas (clases y métodos) que pueden emplear los creadores de software para desarrollar aplicaciones.

Atributo: Propiedad o característica de las entidades de una base de datos.

Azimut: Dirección medida en grados, en el sentido de las agujas del reloj, respecto del Norte. Por ejemplo, el Sur tiene un acimut de 180º.

Base de datos: Conjunto de datos organizado e interrelacionado. Una base de datos suele estar formada por tablas que se componen de registros (filas) y campos (columnas).

CAD: Siglas de *Computer Assited Design* (Diseño asistido por ordenador). Sistema que permite dibujar y diseñar por ordenador. Permite diseñar en dos o tres dimensiones mediante geometría alámbrica, esto es, puntos, líneas, arcos, splines; superficies y sólidos para obtener un modelo numérico de un objeto o conjunto de ellos.

Capa: Sistema de clasificación de entidades de un sistema GIS o CAD que generalmente está asociado a un tipo de entidad, por ejemplo capa de infraestructuras o capa de ciudades.

Clave primaria: Atributo (simple o compuesto) que identifica, de forma unívoca, una entidad de datos.

Clave ajena: Atributo (simple o compuesto) que aparece en una entidad y que es a su vez clave primaria en otra entidad.

Cobertura (*coverage*): Organización topológica de entidades de un GIS que representa un tema.

Cartografía: La cartografía es la ciencia que trata de la representación de la Tierra sobre un mapa. Al ser la Tierra esférica ha de valerse de un sistema de proyecciones para pasar de la esfera al plano.

DATUM: Establece los parámetros que permiten relacionar un geoide (por ejemplo la tierra) y el elipsoide que se utiliza para representarlo.

Digitalización: Es la captura de datos gráficos en técnica vectorial, raster o incluso la captura de datos alfanuméricos.

Elipsoide: Figura geométrica que surge al revolucionar una elipse sobre su diámetro principal.

Entidad: Representación abstracta de un objeto con información propia y que normalmente tiene una correspondencia en el mundo real.

Formato: Estructura de un fichero que posibilita su reconocimiento por un programa o periférico.

Geodesia: La Geodesia es una rama de las Geociencias y una Ingeniería, que trata del levantamiento y de la representación de la forma y de la superficie de la Tierra, global y parcial, con sus formas naturales y artificiales.

Georreferenciación (*Geocoding*): Proceso por el cual se le asignan coordenadas a una entidad o bien se le asocia un dato a una entidad ya referenciada geográficamente.

GIS: Siglas de *Geographic Information System* (Sistema de información geográfica). Denominación inglesa de los sistemas de información geográfica.

GML: Siglas de *Geography Markup Language* (Lenguaje de Marcado Geográfico). Es un sublenguaje de XML descrito como una gramática en XML Schema para el modelaje, transporte y almacenamiento de información geográfica.

GPS: Siglas de *Global Positioning System* (Sistema de posicionamiento global). Sistema que permite determinar en todo el mundo la posición geográfica en la que se encuentra el receptor, con una desviación de cuatro metros.

GRPS: Siglas de *General Packet Radio Service*. Es una tecnología digital de telefonía móvil que proporciona altas velocidades de transferencia.

GSM: Siglas de *Global System for Mobile communications* (Sistema Global para las Comunicaciones Móviles). Estándar europeo para teléfonos móviles digitales.

HTTP: Siglas de *HyperText Transfer Protocol* (protocolo de transferencia de hipertexto). Es el protocolo usado en cada transacción de la Web. El hipertexto es el contenido de las páginas web, y el protocolo de transferencia es el sistema mediante el cual se envían las peticiones de acceso a una página y la respuesta con el contenido.

JSON: Siglas de *JavaScript Object Notation* (Notación de objetos en JavaScript). Es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de Javascript que requiere el uso de XML.

Longitud: Distancia angular horizontal, paralela al ecuador, entre el Meridiano de Greenwich en Londres y un determinado punto de la Tierra.

Latitud: Distancia angular, medida sobre un meridiano, entre una localización terrestre (o de cualquier otro planeta) y el ecuador.

Meridiano: Círculo máximo que proporciona la longitud o distancia Este-Oeste de la Tierra.

OGC: Siglas de *Open Geospatial Consortium*. Consorcio compuesto por 341 organizaciones públicas y privadas. Su fin es la definición de estándares abiertos e interoperables dentro de los Sistemas de Información Geográfica.

Paralelo: Línea resultante de la intersección de un plano paralelo al del Ecuador y la superficie terrestre.

Píxel: Un píxel es un elemento individual de una imagen (un punto) en un terminal o pantalla de ordenador. En ámbito de las imágenes raster se entiende por píxel a cada uno de las celdillas individuales que componen la imagen.

Polígono: Entidad vectorial básica de un GIS que permite representar un área determinada.

Punto: Entidad básica de un GIS que consta de unas coordenadas, un símbolo de representación y una referencia a la entidad asociada.

Sistema Gestor de Bases de Datos (SGBD): Los Sistemas Gestores de Bases de Datos son un tipo de software muy específico, dedicado a servir de interfaz entre las bases de datos y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta.

Sistema abierto: Un sistema informático que emplea normas públicamente disponibles, de forma que puede interactuar con otros sistemas, software o dispositivos que emplean la misma norma.

Sistema de información geográfico: Sistema informático que permite la gestión de grandes volúmenes de datos organizados en base de datos referenciados espacialmente y que pueden ser visualizados mediante mapas.

XML: Siglas de *eXtensible Markup Language* (lenguaje de marcas extensible). Es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades.

UTM: Siglas de *Universal Transversal Mercator*. Es un sistema de coordenadas basado en la proyección geográfica transversa de Mercator, que se construye como la proyección de Mercator normal, pero en vez de hacerla tangente al Ecuador, se la hace tangente a un meridiano. Esta proyección divide la esfera terrestre en 60 husos de proyección tangentes al ecuador. A España le corresponden los husos 29, 30 y 31.

Vectorial: En el modelo de GIS vectorial, el interés de las representaciones se centra en la precisión de localización de los elementos sobre el espacio. Para modelizar digitalmente las entidades del mundo real se utilizan tres objetos espaciales: el punto, la línea y el polígono.

WFS: Siglas de *Web Feature Service* (Servicio de entidades vectoriales). Este servicio proporciona la información relativa a la entidad almacenada en una capa vectorial (cobertura) que reúnen las características formuladas en la consulta.

WMS: Siglas de *Web Map Service* (Servicio de mapas). Servicio web que produce mapas en formato imagen a la demanda para ser visualizados por un navegador web o en un cliente simple.

10 Referencias

10.1 Sistemas de información geográfica

[1] **Joaquín Bosque Sendra. Sistemas de Información Geográfica**. 2ª edición. Ed. RIALP, 1997 (1ª ed. 1992).

[2] **Universidad de Alcalá**.
[http://www.geogra.uah.es/gisweb/1modulosespanyol/IntroduccionSIG/GISModule/GISTheory.doc, marzo 2007]

[3] **Universidad de Murcia. Sistemas de Información Geográfica**. Francisco Alonso Sarría.
[http://www.um.es/geograf/sigmur/sigpdf/temario_in.pdf, marzo 2007]

[4] **Centro de Supercomputación de Galicia**. [http://www.cesga.es/content/category/7/36/47/lang,es/, marzo 2007]

[5] **Kingstom Center For GIS. Introducting GIS**. [http://www.emich.edu/visit/Publications/Intro_to_GIS.ppt, marzo 2007]

[6] **Centro documental Monografias.com**.
[http://www.monografias.com/trabajos14/informageogra/informageogra.shtml, marzo 2007]

[7] **Instituto de Investigación de Recursos Biológicos Alexander Von Humboldt**.
[http://www.humboldt.org.co/humboldt/mostrarpagina.php?codpage=70001#9, marzo 2007]

[8] **Página personal de Gabriel Ortiz**. [http://recursos.gabrielortiz.com/index.asp?Info=012, marzo 2007]

[9] **GIS.com** (The guide to Geographic Information Systems). [http://www.gis.com/whatisgis/geographymatters.pdf, marzo 2007]

[10] **Dirección general del catastro** [http://www.catastro.minhac.es/, marzo 2007]

[11] **Un SIG para el INE. Revista Fuentes Estadísticas**. [http://www.ine.es/revistas/fuentes/numero38/pag15.htm, marzo 2007]

[12] **Mapping interactivo. Revista internacional de ciencias de la Tierra**.
[http://www.mappinginteractivo.com/plantilla-ante.asp?id_articulo=954, marzo 2007]

[13] **Universidad de Cádiz**. [http://www.uca.es/dept/filosofia/TEMA%206.pdf (y otros), marzo 2007]

[14] **SOPDE** (Sociedad de Planificación y Desarrollo de la Diputación Provincial de Málaga).
[http://gis.sopde.es/cursosgis/DHTML/que_2_2.html, marzo 2007]

[15] **Tipos de SIG y modelos de datos**. [http://recursos.gabrielortiz.com/index.asp?Info=012, marzo 2007]

[16] **Tendencias de los Sistemas de Información Geográfica** (Aart van Wingerden, Gerente Regional, ESRI Europa).
[http://www.elagrimensor.com.ar/elearning/lecturas/Trend_SIG1.pdf, marzo 2007]

10.2 Cartografía

[17] **Universidad de Valladolid. Apuntes de Cartografía**. [http://www.cartesia.org, marzo 2007]

[18] **Wikipedia. Acimut** [http://es.wikipedia.org/wiki/Acimut, marzo 2007]

[19] **Centro de Documentos, Publicaciones y Recursos Educativos. El Sistema de Referencia Geológico**. [http://www.monografias.com/trabajos16/geoide/geoide.shtml, marzo 2007]

[20] **Wikipedia. Proyecciones** [http://es.wikipedia.org/wiki/Proyecci%C3%B3n_geogr%C3%A1fica, marzo 2007]

[21] **Portal de Tecnologías de la Información Geográfica. Nociones de Topografía**. [http://www.nosolosig.com/, marzo 2007]

[22] **ESRI**. [http://www.esri.com/, marzo 2007]

[23] **Formato PNG**. [http://es.wikipedia.org/wiki/PNG, abril 2007]

[24] **Formato JPEG**. [http://es.wikipedia.org/wiki/JPEG, abril 2007]

[25] **Formato GIF**. [http://es.wikipedia.org/wiki/GIF, abril 2007]

[26] **Dirección General del Catastro**. [http://www.catastro.minhac.es, marzo 2007]

[27] **Instituto Cartográfico Valenciano**. [http://www.gva.es/icv/, marzo 2007]

10.3 OGC

[28] **Open Geospatial Consortium. Open Geospatial Consortium Specifications**. [http://www.opengeospatial.org/standards, marzo 2007]

[29] **Open Geospatial Consortium. Catalogue Service Implementation Specification**. [http://portal.opengeospatial.org/files/index.php?artifact_id=5929&version=2, marzo 2007]

[30] **Open Geospatial Consortium. Simple Feature Access Part 1**. [http://www.opengeospatial.org/standards/sfa, marzo 2007]

[31] **Open Geospatial Consortium. Simple Feature Access Part 2**. [http://www.opengeospatial.org/standards/sfs, marzo 2007]

[32] **Open Geospatial Consortium. Web Coverage Service (WCS) Implementation Specification**. [http://portal.opengeospatial.org/files/?artifact_id=18153, marzo 2007]

[33] **Open Geospatial Consortium. Coordinate Transformation Service Implementation Specification**. [http://portal.opengeospatial.org/files/?artifact_id=999, marzo 2007]

[34] **Open Geospatial Consortium. Geography Markup Language**. [http://portal.opengeospatial.org/files/?artifact_id=4700, marzo 2007]

[35] **Open Geospatial Consortium. Filter Encoding Implementation Specification**. [http://portal.opengeospatial.org/files/index.php?artifact_id=8340, marzo 2007]

[36] **Open Geospatial Consortium. Web Map Service (WMS) Implementation Specification**. [http://portal.opengeospatial.org/files/?artifact_id=14416, marzo 2007]

[37] **Open Geospatial Consortium. Web Feature Service (WFS) Implementation Specification**. [http://portal.opengeospatial.org/files/?artifact_id=8339, marzo 2007]

[38] **Open Geospatial Consortium. Styled Layer Descriptor (SLD) Implementation Specification**. [http://portal.opengeospatial.org/files/?artifact_id=1188, marzo 2007]

[39] **Universidad de La Coruña. Definición e implementación de un Servicio Web de Mapas Activos**. [http://rosalia.dc.fi.udc.es/PlainConsulta/ArchivosAdministracion/ficheros_publicaciones/Defeimpde.pdf, marzo 2007]

[40] **LatinGeo**. [http://www.latingeo.net/lab_geo/index.php?, marzo 2007]

[41] **Universidad Politécnica de Madrid. Grupo mercator**. [http://mapas.topografia.upm.es/grupomercator/, marzo 2007]

[42] **Puesta en marcha y explotación de geoservicios del OpenGeospatial**. [http://mapas.topografia.upm.es/geoserviciosOGC/Curso-OGC.htm, marzo 2007]

[43] **Consejo Superior Geográfico. Infraestructura de Datos Espaciales de España**. [http://www.idee.es/show.do?to=pideep_pidee.ES, marzo 2007]

[44] **Universidad de Alicante. Sistema de información geográfica de la Universidad de Alicante**. [http://www.sigua.ua.es/es/utilidades/phpwms/index.php, marzo 2007]

[45] **Wikipedia. Wep Map Service**. [http://es.wikipedia.org/wiki/Web_Map_Service, marzo 2007]

[46] **Xunta de Galicia. SITGA**. [http://sitga.xunta.es/wfs.asp, marzo 2007]

[47] **Dirección General de Catastro. Servicio WFS**. [http://ovc2.catastro.meh.es/cartografia/wfs/servidorwfs.aspx, marzo 2007]

[48] **Dirección General de Catastro. WFS**. [http://jidee06.uji.es/down/s75_catastro.pdf, marzo 2007]

[49] **Dirección General de Catastro. Servicio WMS**. [http://ovc.catastro.meh.es/Cartografia/WMS/ServidorWMS.aspx, marzo 2007]

[50] **Dirección General de Catastro. WMS**. [http://www.csi.map.es/csi/tecnimap/tecnimap_2006/03T_PDF/la%20cartografia%20catastral.pdf, marzo 2007]

10.4 API Google Maps

[51] **Google Maps. Términos y condiciones de Google Maps**. [http://maps.google.es/help/terms_maps.html, abril 2007]

[52] **API Google Maps**. [http://www.google.com/apis/maps/documentation, abril 2007]

[53] **Como usar maps de Google en tu web**. [http://www.mundogoogle.com/index.php?Ver=Articulo&Id=44, abril 2007]

[54] **Google Maps API Tutorial**. [http://www.econym.demon.co.uk/googlemaps/, abril 2007]

[55] **OpenLayers**. [http://www.openlayers.org/, abril 2007]

[56] **Nática Sistemas de Información, S.L**. [http://www.natica.com, abril 2007]

[57] **WikiLoc**. [http://www.wikiloc.com/wikiloc/spatialArtifacts.do, abril 2007]

[58] **Google Maps Mania**. [http://www.googlemapsmania.blogspot.com/, abril 2007]

[59] **Nichloas C. Zakas** [et al.]. AJAX. Ed. Anaya Multimedia, 2007.

[60] **Nichloas C. Zakas. JavaScript para desarrolladores web**. Ed. Anaya Multimedia, 2006.

[61] **Ian Gilfillan, La biblia MySQL**. Ed. Anaya Multimedia, 2003.

[62] **Mohammed J. Kabir. La biblia Servidor Apache 2**. Ed. Anaya Multimedia, 2003.

[63] **Coggeshall, John. La biblia PHP 5**. Ed. Anaya Multimedia, 2005.

[64] **O'Reilly. What Is Web 2.0**. [http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html, abril 2007]

[65] **Servicio de trenes del Reino Unido**. [http://www.traintimes.org.uk/map/?oxf, mayo 2007]