



**Escola Politècnica Superior
de Castelldefels**

UNIVERSITAT POLITÈCNICA DE CATALUNYA

TRABAJO DE FIN DE CARRERA

TITULO DEL TFC: Acceso a aplicaciones desde dispositivos móviles mediante servicios Web en función del posicionamiento geográfico.

TITULACIÓN: Ingeniería Técnica de Telecomunicación, especialidad Telemática

AUTORES: David Andrés López Nuevo, Rubén Cuadrado Sánchez

DIRECTOR: Dolors Royo Vallés

FECHA: 24 de Julio de 2007/08

Título: Acceso a aplicaciones desde dispositivos móviles mediante servicios Web en función del posicionamiento geográfico.

Autores: David Andrés López Nuevo, Rubén Cuadrado Sánchez

Director: Dolors Royo Vallés

Fecha: 24 de Julio de 2007/08

Resumen

La idea principal de este proyecto es ofrecer servicios a los alumnos de la universidad UPC. Los alumnos accederán a estos servicios a través de una aplicación para dispositivos móviles.

La UPC está formada por diversos campus, situados en localidades diferentes, y cada uno de ellos dispondrá de sus propios servidores. Para seleccionar el campus al cual el alumno se conectará, se hará uso de la tecnología *GPS*. El alumno accederá directamente al servidor que se encuentre más próximo a su posición. De esta manera se pretende conseguir que el tiempo de retardo entre el dispositivo y el servidor sea mínimo.

El servicio ofrecido es un *podcast*. Este servicio tiene la finalidad de permitir a los alumnos acceder al material docente ofrecido por la universidad, como por ejemplo, documentación en formato *PDF* de las asignaturas, colecciones de problemas y ejercicios, etc.

Para el desarrollo de este proyecto ha sido necesario el estudio de la tecnología *Microsoft .NET*. Esta tecnología ofrece un entorno de desarrollo que permite la programación de aplicaciones para dispositivos móviles, así como la implementación de servicios Web.

También se ha requerido el estudio de otros ámbitos que intervienen en este proyecto, como son los servidores Web y las bases de datos. Este proyecto hace uso de servidores Web para la publicación tanto de los *podcast*, como de las páginas y los servicios Web. También ha sido necesario el uso de una base de datos para almacenar información sobre los campus, los servidores y los *podcast*.

Finalmente se ha conseguido implementar una aplicación para dispositivos móviles capaz de recibir e interpretar la señal *GPS*, así como gestionar, descargar y reproducir archivos de *podcast*. Además esta aplicación hace uso de servicios Web para tratar con la información de la base de datos.

Title: Access to applications from mobil devices by means of Web services in function of the geographical positioning.

Authors: David Andrés López Nuevo, Rubén Cuadrado Sánchez

Director: Dolors Royo Vallés

Date: July, 24th of 2007/08

Overview

The main idea of this project is to offer service to students of the UPC university. The students will access to this services across an application for mobile devices.

The UPC is formed by several campuses. These campuses are situated in different localities and every of them have their own servers. GPS is used for select the campus that the student will connect. The student will connect directly to the closest server to his position. Thus, the lag between the mobile device and the server is minimal.

The offered service is a podcast. By means of this service the students can download the educational material offer by the university. For example, documentation of the subjects, collection of exercises, ...

To develop this project was necessary to study *.NET* technology. This technology offers a development environment that allows programming applications for mobile devices, as well as Web services implementation.

This project uses a Web server in order to publish podcast, Web services and Web pages. Also has been necessary a database to store information about the campuses, the servers and the podcast.

Finally, an application for mobile devices capable of receive and process the *GPS* signal as well as managing, downloading and playing podcast files was implemented successfully. In addition, this application invokes Web services in order to access to the information stored in the database.

ÍNDICE

CAPÍTULO 1. INTRODUCCIÓN	1
1.1. Motivación	1
1.2. Razón y contexto	2
1.3. Objetivos.....	2
1.3.1. Idea inicial.....	2
1.3.2. Evolución del proyecto	3
1.3.3. Objetivo final	4
CAPÍTULO 2. TECNOLOGÍAS UTILIZADAS.....	5
2.1. Tecnología Microsoft .NET.....	5
2.1.1. ¿Qué es Microsoft .NET?.....	5
2.1.2. Estructura y componentes de .NET	6
2.1.3. Visual Studio.....	8
2.1.4. MSDN (MicroSoft Developer Network)	9
2.1.5. Convenciones de programación	9
2.2. Servicios Web.....	9
2.2.1. ¿Qué es un servicio Web?	9
2.2.2. Funcionamiento	10
2.2.3. XML (eXtensible Markup Language).....	11
2.2.4. SOAP (Simple Object Access Protocol).....	12
2.2.5. WSDL (Web Services Description Language).....	14
2.2.6. UDDI (Universal Description, Discovery and Integration)	15
2.3. Base de datos	16
2.3.1. ¿Qué es una base de datos?	16
2.3.2. RDBMS (Relational DataBase Management System).....	16
2.3.3. Componentes.....	16
2.3.4. SQL (Structured Query Language)	17
2.4. Tecnología GPS (Global Positioning System).....	18
2.4.1. ¿Qué es GPS?.....	18
2.4.2. Componentes.....	18
2.4.3. Funcionamiento	18
2.4.4. Protocolo NMEA	19
2.5. Podcast	20
2.5.1. ¿Qué es un podcast?	20
2.5.2. Componentes.....	20
CAPÍTULO 3. DISEÑO DEL PROYECTO	23
3.1. Elección de software	23
3.1.1. Entorno de trabajo: Visual Studio .NET	23
3.1.2. Servidor: IIS (Internet Information Services)	25
3.1.3. Base de Datos: MySQL.....	28
3.1.4. GPS: software propio	32
3.1.5. Podcast: software propio.....	32
3.1.6. Diseño global	33

3.2. Diseño del modelo	34
3.2.1. Descripción del sistema	34
3.2.2. Gestión de la base de datos	36
3.2.3. Conexión del dispositivo cliente con el servidor de un campus	37
3.2.4. Servicio podcast.....	40
CAPÍTULO 4. INTERFAZ Y PRUEBAS	45
4.1. Interfaz gráfica del cliente	45
4.1.1. Formulario inicio.....	45
4.1.2. Formulario navegador	47
4.1.3. Formulario podcast	48
4.1.4. Formulario reproductor.....	52
4.1.5. Configuración.....	53
4.1.6. Salir	57
4.2. Pruebas realizadas	57
4.2.1. Consultas SQL través de un servicio Web.....	57
4.2.2. Cálculo de distancia entre cliente y campus	59
4.2.3. Pruebas finales	60
CAPÍTULO 5. CONCLUSIONES	68
5.1. Objetivos conseguidos	68
5.2. Ámbito de uso	68
5.3. Ampliaciones y mejoras	69
5.4. Conclusiones personales	69
ANEXO 1. IMPLEMENTACIÓN DEL DISEÑO	73
ANEXO 2. PORTAL WEB	80
ANEXO 3. DESCRIPCIÓN DE LOS MENSAJES DE LA APLICACIÓN	83

CAPÍTULO 1. INTRODUCCIÓN

En este capítulo se comentan los motivos y las razones para llevar a cabo este proyecto. También se explican sus objetivos principales.

En el segundo capítulo se explican las tecnologías que intervienen en este proyecto.

En el tercer capítulo se muestra el software que se ha decidido utilizar, justificando en cada caso la elección realizada. También se explica el diseño del sistema

En el siguiente capítulo se explican cada una de las diferentes interfaces gráficas de la aplicación que se mostrarán al usuario. A continuación se comentan algunas de las pruebas más relevantes realizadas.

En el quinto capítulo se muestran las conclusiones a las que se han llegado, comparando los resultados finales con los objetivos propuestos al inicio. Además se comentan posibles usos comerciales del proyecto, así como sus posibles ampliaciones futuras.

El último capítulo contiene la bibliografía utilizada para la realización de este proyecto.

En el anexo 1 se muestran dos diagramas relativos a la implementación de la aplicación, los diagramas de clases y de secuencia.

En el anexo 2 se explican las páginas Web que componen el portal Web.

El anexo 3, al final del documento, contiene información acerca de los mensajes de error, advertencia e información que puede mostrar la aplicación cliente.

1.1. Motivación

Existe una motivación personal para la elección de este proyecto. Los temas tratados en él nos parecen interesantes. La plataforma *.NET* y los servicios Web cada vez están cobrando más importancia en el mundo de la telemática. Además, otra de las motivaciones es el poder trabajar con dispositivos móviles y el poder diseñar una aplicación que trabaje sobre éstos. Con ello se ponen en práctica los conocimientos que se han ido adquiriendo durante la carrera y permite una aproximación a la realización de una aplicación real dentro del mundo de la telemática.

Además de lo comentado anteriormente, hay que añadir que el propio desarrollo y la gestión del proyecto, así como el estudio de las tecnologías que intervienen en él, es un reto que un estudiante de ingeniería ha de superar.

1.2. Razón y contexto

En la sociedad actual es muy importante el concepto de la movilidad, puesto que el día a día de las personas es cada vez más ajetreado y con más prisas, y eso conlleva unos desplazamientos diarios y un uso del tiempo lo más apurado posible. La tecnología tiene un papel muy importante dentro de este campo, puesto que las personas tienen la necesidad de conseguir información, ya sea por algún asunto personal o para el trabajo.

La tecnología, cada vez más, tiende a mejorar los aspectos de movilidad, es decir, poder utilizarla en diferentes lugares: mientras te mueves, viajas, en casa o en el trabajo.

Las tecnologías móviles están evolucionando a un ritmo realmente rápido puesto que es lo que demanda la sociedad actual. También se puede observar que, cada vez más, los teléfonos móviles ya tienen acceso a Internet y permiten, por ejemplo, ver el correo desde cualquier lugar, ya sea en la calle, en el tren...

Este TFC hace uso de estas tecnologías para ofrecer servicios destinados a los usuarios de dispositivos móviles.

1.3. Objetivos

1.3.1. Idea inicial

Uno de los objetivos de este proyecto era el desarrollo de un servicio, que ofrecerá la UPC, al cual podría acceder cualquier alumno de la universidad mediante cualquier tipo de dispositivo móvil.

Dicho servicio consistía en que el dispositivo realizaba una petición al servidor y éste le devolvía una página Web dinámica. Independientemente del tipo de dispositivo móvil utilizado por el usuario, un teléfono móvil o una PDA; la página tendría que visualizarse correctamente.

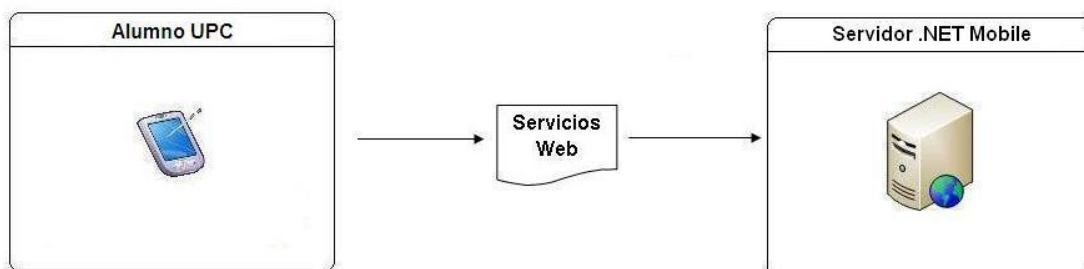


Fig. 1.1 Sistema inicial

Uno de los requisitos de este TFC era implementar el servicio usando la tecnología *.NET Mobile*. Dicha tecnología permite la programación de páginas Web para mostrarlas correctamente en dispositivos móviles.

Otro requisito del proyecto era el de utilizar servicios Web para realizar la comunicación entre el dispositivo cliente y el servidor.

1.3.2. Evolución del proyecto

1.3.2.1. Introducción del GPS

Como la universidad UPC dispone de diversos campus, se pensó en tener alojado el servicio en cada uno de ellos. De este modo, los alumnos podrían acceder al servicio a través de su propio campus.

Para averiguar el campus al cual el alumno deberá conectarse, era necesario conocer la posición geográfica de los campus y la del propio alumno. El alumno se conectará con el servidor del campus que tenga más próximo. Para realizar este proceso fue necesaria la introducción de algún mecanismo para conocer las posiciones de cada uno de los elementos del sistema: el *GPS*.

El uso de la tecnología *GPS* permite obtener las coordenadas (latitud y longitud) donde están ubicados el dispositivo del alumno y los diferentes campus que pertenecen a la UPC. Para poder usar esta tecnología, el dispositivo móvil deberá disponer de un receptor *GPS*.

En la figura 1.2 se muestra un ejemplo de cómo el dispositivo lee las coordenadas del satélite *GPS*. Mediante estas coordenadas, el dispositivo puede conocer su posición actual, que en este ejemplo es Castelldefels. Una vez conoce su posición, ya se puede conectar al campus más cercano.

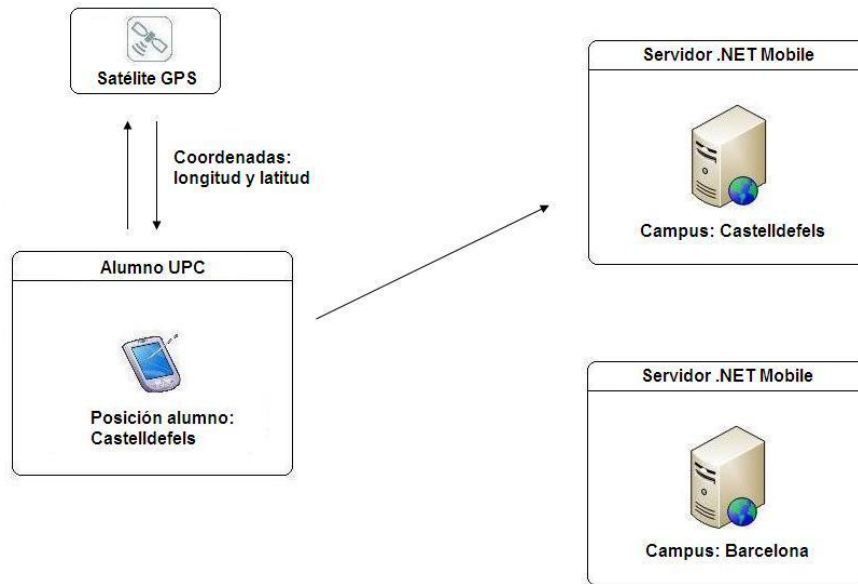


Fig. 1.2 Sistema con *GPS*

1.3.2.2. Incorporación de los servicios

Se decidió que los servicios ofrecidos por los servidores de cada campus fueran la suscripción a un *podcast* y el acceso a un portal Web.

Mediante el servicio de *podcast*, se pretende que los alumnos puedan descargarse a su dispositivo móvil el material docente que publique la universidad. El material ofrecido, por ejemplo, podría ser un archivo con la teoría de alguna asignatura en formato *PDF*, enunciados de prácticas, etc.

Por otro lado, el portal Web está formado por una serie de páginas Web que muestran al estudiante diferente información sobre la universidad.

1.3.3. Objetivo final

El objetivo final de este proyecto es implementar un servicio de *podcast* para los alumnos de la UPC. Estos alumnos accederán al servicio a través de una aplicación para dispositivos móviles.

Esta aplicación tiene que ser capaz de conectarse con el campus más próximo al alumno haciendo uso del *GPS* y de servicios Web. Una vez conectado a este campus, la aplicación podrá descargar los archivos *podcast* y acceder al portal Web.

CAPÍTULO 2. TECNOLOGÍAS UTILIZADAS

En este capítulo se hace una introducción a las tecnologías que se han utilizado en este proyecto.

2.1. Tecnología Microsoft .NET

2.1.1. ¿Qué es Microsoft .NET?

.NET es una tecnología desarrollada por *Microsoft*, es un nuevo tipo de arquitectura (no es la evolución de ninguna tecnología anterior) para desarrollar aplicaciones y distribuirlas como servicios Web. Esta arquitectura está pensada para entornos de red, como Internet, y permite que las aplicaciones puedan ser los más distribuidas posibles.

Internet ha llegado a convertirse en el principal entorno de trabajo para este tipo de aplicaciones. Un usuario puede acceder a Internet a través de dispositivos muy diferentes (como puede ser un ordenador, un teléfono, una PDA...).

Microsoft .NET ofrece herramientas para que las aplicaciones puedan funcionar sobre cualquier plataforma y puedan realizar correctamente la comunicación con el servidor.

Gracias a esta tecnología se consigue aumentar enormemente la flexibilidad de la programación de aplicaciones para Internet, ya que se pretende que los desarrolladores se dediquen al alquiler y a la publicación en Internet de las aplicaciones que ellos programan y que se puedan acceder a ellas mediante los servicios Web.

Los desarrolladores pueden hacer que sus aplicaciones utilicen los propios servicios Web que ellos han programado. Pero también pueden ofrecer esos servicios Web a otras aplicaciones e otros desarrolladores. De esta forma aparecen los Proveedores de servicios Web.

Por lo tanto, el principal objetivo de *.NET* es el siguiente:

- Proporcionar a los desarrolladores las herramientas necesarias para que sean capaces de programar aplicaciones basadas en servicios Web.
- Independencia de la plataforma.

2.1.2. Estructura y componentes de .NET

2.1.2.1. .NET Framework

.NET Framework es el elemento base de esta tecnología. Proporciona todas las herramientas necesarias para permitir la creación de las aplicaciones. Estas herramientas se pueden clasificar en tres grandes grupos. El *CLR (Common Language Runtime)*, la *BCL (Base Class Libraries)* y la creación de interfaces para el usuario.

El *CLR* es la parte más importante de *.NET Framework*, ya que se encarga de gestionar todo el código y la memoria.

La *BCL* es un conjunto de clases que forman una librería de servicios que pueden utilizar los programadores, por ejemplo, el acceso a datos o la gestión de hilos de ejecución.

Las interfaces de las aplicaciones para el usuario se pueden dividir en dos grupos dependiendo de la aplicación. Si la aplicación es una aplicación para la red, se utilizará *ASP.NET*, que proporciona los servicios Web y los formularios Web necesarios para programar este tipo de aplicaciones. Pero si en cambio, se quiere programar una aplicación para *Windows*, se utilizarán los *Windows Forms*, que permiten crear las aplicaciones para *Windows* gráficamente, mediante la utilización de controles.

2.1.2.2. CLR (Common Language Runtime)

El *CLR* es elemento más importante de *.NET Framework*, ya que se encarga de la gestión del código de las aplicaciones en tiempo de ejecución y proporciona servicios para facilitar su desarrollo.

Algunas de las funciones que proporciona el *CLR* son las siguientes:

- Administra el código en tiempo de ejecución y proporciona un recolector de basura (*garbage collector*) que se encarga de liberar los recursos que utiliza un objeto cuando éste deja de estar activo.
- Proporciona un sistema común de tipos, que permite la interoperabilidad entre diferentes objetos aunque estos estén escritos en diferentes lenguajes.
- También permite el control de excepciones entre lenguajes y gestiona la seguridad del código cuando se ejecuta.

A continuación se explican los elementos más importantes del *CLR* y como ejecuta el código.

2.1.2.3. CLR: CTS (*Common Type System*)

El CTS es el sistema común de tipos, que define como se crean los tipos en el CLR y le dice al CLR como quiere que se ejecute el código.

Un tipo es un elemento que se ejecuta dentro del CLR y que está implementado como una clase. Ese elemento es un objeto, por ejemplo, una instancia de una clase o una variable.

La funcionalidad más importante del CTS es la de proporcionar una serie de normas que permiten la comunicación entre objetos que estén escritos en diferentes lenguajes. Por ejemplo, desde un código escrito en *Visual Basic .NET*, se puede heredar de una clase escrita en *C#*.

2.1.2.4. CLR: Metadatos

Los metadatos son datos extra usados por la aplicación pero que no pertenecen al código ejecutable. Estos metadatos son creados por el compilador y describen los tipos, los miembros y las referencias del código. Estos metadatos los utiliza el CLR para cargar clases, colocar instancias en memoria, invocar a métodos o generar código nativo.

2.1.2.5. CLR: Lenguaje de programación

Es el lenguaje de programación con el cual se desarrollarán las aplicaciones. .NET permite utilizar diferentes lenguajes: Visual Basic .NET, C#, C++, J# entre otros. Todos estos lenguajes, para pertenecer a .NET Framework, han de cumplir una serie de características, definidas por el CLS.

2.1.2.6. CLR: CLS (*Common Language Specification*)

La especificación común de lenguajes (CLS) es un grupo de características que tienen que cumplir todos los lenguajes que forman parte de la plataforma .NET. El CLS hace posible que diferentes lenguajes de programación se puedan comunicar entre ellos y puedan entenderse.

Las finalidades del CLS son las siguientes:

- Independencia del lenguaje. Se puede escribir una aplicación en el lenguaje que decida el programador. El programador no se verá obligado a utilizar un lenguaje porque este le da funcionalidades que no le puede dar otro lenguaje, sino que ahora el programador podrá escribir la aplicación en el lenguaje que le sea más cómodo, sin que varíe el resultado final.
- Integración entre lenguajes. Siempre que se cumplan las reglas del lenguaje común, es posible que los diferentes lenguajes se puedan entender entre ellos.

- Integración de nuevos lenguajes. Al ser una plataforma abierta, cualquier grupo de programadores, y no sólo de Microsoft podrá añadir un lenguaje a *.NET Framework*, siempre que sigan las reglas del *CLS*. De esta forma se pueden ir añadiendo lenguajes a esta plataforma.

2.1.2.7. CLR: MSIL (*MicroSoft Intermediate Language*)

El compilador es un programa que se encarga de traducir el código que ha desarrollado el programador, no a lenguaje máquina (lenguaje binario) como harían la mayoría de compiladores, sino que genera un lenguaje intermedio llamado *MSIL*.

Este lenguaje intermedio es formado por una serie de instrucciones que serán interpretadas por el CLR en tiempo de ejecución.

Además es independiente del sistema operativo y del procesador donde vaya a ejecutarse el programa, por lo que no es un código ejecutable.

Dependiendo de la plataforma dónde se vaya a ejecutar la aplicación, un compilador *Just-in-time* se encarga de traducir el código *MSIL* a código máquina.

.NET Framework dispone de diferentes compiladores *JIT*. Se utilizará el compilador *JIT* apropiado para que el código *MSIL* se traduzca al código máquina de determinada plataforma.

Por ejemplo, para utilizar una aplicación en *Windows 98*, se utilizará un compilador *JIT* diferente de si la aplicación fuera para *Windows 2000*, pero el código *MSIL* es el mismo en ambos casos.

Gracias al *MSIL* se consigue la interacción entre los diferentes lenguajes, ya que sea cual sea el lenguaje en que esté programada la aplicación (*Visual Basic*, *C#*...) se obtendrá un lenguaje común para todos ellos y podrán comunicarse entre ellos.

2.1.2.8. BCL (*Base Class Libraries*)

Como su propio nombre indica, es la biblioteca de clases que utiliza el CLR. La *BCL* proporciona todos los tipos (objetos) necesarios para programar cualquier aplicación, ya sea una aplicación para *Windows*, o para la red.

2.1.3. Visual Studio

Es una herramienta gráfica para desarrollar aplicaciones utilizando la plataforma *.NET*. Utilizando *Visual Studio* se obtienen algunas ventajas, como es el poder programar dichas aplicaciones con gran rapidez y facilidad.

Gracias a Visual Studio podemos diseñar:

- Servicios Web
- Aplicaciones para *Windows*
- Aplicaciones para dispositivos móviles

2.1.4. MSDN (MicroSoft Developer Network)

Es un sitio Web de *Microsoft* muy útil para los desarrolladores de aplicaciones, ya que es donde se publica una extensa y detallada documentación sobre la plataforma *.NET* y sus lenguajes. Es una gran fuente de ayuda, se pueden encontrar explicaciones, consejos, ejemplos y partes de código que solucionan muchas de las dudas surgidas a la hora de programar.

Esta Web está en inglés, pero también en otros idiomas incluido el castellano.

2.1.5. Convenciones de programación

Las convenciones son una serie de “normas” que habría que seguir para que el código que se programe fuese lo más claro y legible posible.

Un programa realizado por un programador muy probablemente acabe llegando a las manos de otro programador, el cual habrá de realizar modificaciones sobre este. Que el código sea claro facilita la tarea de que otros programadores lo entiendan.

Hay que destacar que no es obligatorio seguir las convenciones, pero las empresas desarrolladoras aconsejan seguirlas, ya que son una buena manera de programar un código más limpio y además evitan que los programadores adquieran malos hábitos.

2.2. Servicios Web

2.2.1. ¿Qué es un servicio Web?

Un servicio Web es una aplicación capaz de interactuar con otras aplicaciones y/o usuarios y de hacerlo a través de Internet. El objetivo que hay detrás de esta idea es poder comunicar sistemas diferentes, tanto a nivel de hardware como de software. Cualquier cliente podrá acceder a un servicio Web independientemente del sistema operativo y la aplicación que utilice. Este acceso se consigue mediante el intercambio de mensajes *XML*.

2.2.2. Funcionamiento

Para el acceso a un servicio Web se utilizan las siguientes tecnologías:

- *XML (eXtensible Markup Language)*: Este es el metalenguaje (lenguaje basado en etiquetas) en el que están implementados los servicios Web.
- *SOAP (Simple Object Access Protocol)*: Este protocolo es el encargado de realizar la comunicación del cliente con el servicio Web. Para que la comunicación se realice correctamente es necesario conocer previamente una serie de parámetros.
- *WSDL (Web Services Definition Service)*: Lenguaje encargado de describir el servicio Web.
- *UDDI (Universal Description Discovery and Integration)*: Este protocolo permite la publicación y localización de los servicios. Se puede definir como una guía de servicios Web.
- *HTTP*: Habitualmente se utilizará este protocolo para realizar el transporte de los diferentes mensajes. No es obligatorio, por lo que también podemos utilizar otros protocolos para este transporte.

En la figura 2.1 se puede observar el acceso por parte de un cliente a un servicio Web.

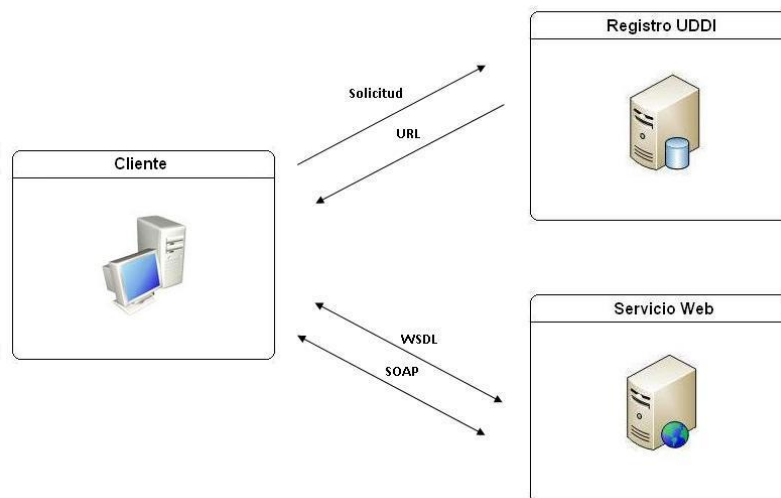


Fig. 2.1 Esquema de servicios Web

El primer paso para acceder a un servicio Web es la localización del servicio deseado a través del registro *UDDI*. Este nos deberá devolver la *URL* del servidor. Con la *URL* accedemos al servidor donde está alojado el servicio y se obtiene una descripción de él gracias a *WSDL*. Una vez se dispone de esta descripción ya podemos formar los mensajes *SOAP* adecuados para la comunicación.

2.2.3. XML (eXtensible Markup Language)

2.2.3.1. Introducción a XML

XML es un lenguaje de etiquetas desarrollado por el *W3C* (*World Wide Web Consortium*). Dispone de un conjunto de reglas para definir etiquetas que organizan y estructuran un documento. Aunque es similar al *HTML* hay grandes diferencias entre ellos. Se podría decir que *HTML* está especializado en la presentación de documentos mientras que *XML* lo está en la gestión de datos para la Web. Su importancia radica en que permite el intercambio de datos entre diferentes plataformas de una manera sencilla.

2.2.3.2. Historia de XML

XML proviene de *GML*, lenguaje creado por *IBM* en los años setenta. *GML*, siglas de *Generalized Markup Language*, surgió dada la necesidad que tenía *IBM* de almacenar grandes cantidades de información de diversos tipos. Este lenguaje fue normalizado por la *ISO* en 1986, dando lugar a *SGML* (*Standard Generalized Markup Language, ISO 8879*).

En el año 1989 nació la *World Wide Web* y junto a esta, también se creó el lenguaje *HTML*, derivado del *SGML*. El problema del *HTML* reside en no se requiere que el documento cumpla con la sintaxis propia del lenguaje *HTML* para ser presentado. Es por ello que desde 1996 el *W3C* se ha dedicado a poner orden en el *HTML* estableciendo reglas y etiquetas para que sea un estándar. Este mismo consorcio en 1998 desarrolló el *XML*. Con este lenguaje se pretende solucionar las carencias del *HTML*.

2.2.3.3. Objetivos y usos

Desde la aparición del *XML* ha habido cierta confusión con respecto a los usos reales que se le pueden dar. Por ello el *W3C* estableció diez objetivos principales:

- *XML* se debe poder usar directamente en Internet.
- *XML* debe admitir una gran variedad de aplicaciones.
- *XML* debe ser compatible con *SGML*.
- Debe ser fácil crear programas que procesen documentos *XML*.
- El número de funcionalidades opcionales de *XML* deberá mantenerse en un mínimo absoluto, preferiblemente cero.
- Los documentos *XML* deberán ser inteligibles para los humanos y razonablemente claros.
- El diseño de *XML* deberá prepararse rápidamente.
- El diseño de *XML* deberá ser formal y conciso.
- Los documentos *XML* deberán ser fáciles de generar.
- La concisión en los marcadores *XML* tiene una importancia mínima.

Además de estos objetivos es importante conocer para qué se utiliza realmente el *XML*.

Uno de sus usos más importantes es la comunicación de datos. Cualquier aplicación puede trabajar con *XML*, indiferentemente del lenguaje utilizado. Por ello para el intercambio de datos entre aplicaciones es más sencillo con la utilización de este lenguaje.

Otro uso es el de base de datos. Cuando se dispone de una base de datos sencilla es posible almacenarla dentro de un *XML*, ahorrando así recursos en el servidor.

También hay que destacar el uso de *XML* dentro del ámbito de los servicios Web. Cuando se invoca un servicio Web es necesaria la transmisión de ciertos parámetros, que se realiza mediante *SOAP*. *XML* es la base de *SOAP* y *WSDL*.

2.2.4. SOAP (Simple Object Access Protocol)

2.2.4.1. Introducción

SOAP es un protocolo que permite la comunicación entre aplicaciones a través de Internet. Es el núcleo de los servicios Web. Está basado en *XML*, es por ello que lo hace independiente tanto de la plataforma como del lenguaje utilizado. En sí mismo, un mensaje *SOAP* es un documento *XML*. Mediante el intercambio de estos mensajes se establece la comunicación entre cliente y servidor para el acceso a un servicio Web. Generalmente *SOAP* utiliza *HTTP* para su transporte aunque no es una exigencia y pueden usarse otros protocolos.

2.2.4.2. Ventajas

Las ventajas de utilizar *SOAP* son, principalmente, que no está ligado a ningún lenguaje de programación ni tampoco a ningún protocolo de transporte. A diferencia de otros protocolos de este tipo no utiliza código binario, sino que está basado en *XML*, por lo que lo hace más comprensible por parte de los programadores. Además, aprovecha los estándares ya afianzados, tal como *HTTP* o *XML*. De esta forma se consigue la interoperabilidad entre diferentes entornos y plataformas.

2.2.4.3. Formato de los mensajes SOAP

En la siguiente figura se puede observar el mensaje *SOAP* de ejemplo que envía un cliente a un servicio Web.

```
POST /ServicioAdministrador/Service.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://www.upc.es/InsertarServidor"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <InsertarServidor xmlns="http://www.upc.es/">
      <servidor>string</servidor>
      <url>string</url>
      <campus>string</campus>
    </InsertarServidor>
  </soap:Body>
</soap:Envelope>
```

Fig. 2.2 Mensaje petición SOAP

El servicio Web, al recibir la petición del cliente le envía a éste una respuesta, como muestra la figura 2.2.

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <InsertarServidorResponse xmlns="http://www.upc.es/">
      <InsertarServidorResult>string</InsertarServidorResult>
    </InsertarServidorResponse>
  </soap:Body>
</soap:Envelope>
```

Fig. 2.3 Mensaje respuesta SOAP

Podemos observar que tanto la solicitud como la respuesta siguen el mismo formato y además ambos son unos documentos XML totalmente válidos.

Veamos a continuación las partes que componen los mensajes SOAP.

- Cabecera *HTTP*. Indica a donde se envía la petición además del tipo de contenido, codificación y longitud del mensaje. Esta cabecera *HTTP* dispone de un campo extra para indicar que es una petición *SOAP*. Este campo se llama *SOAPAction*. También incluye la declaración del tipo de documento que contiene el mensaje *HTTP*, *Content-Type*, que en este caso es *XML*.
- Dentro de la información contenida en el mensaje, se indica en la primera línea, `<?xml version="1.0" encoding="utf-8"?>`, el tipo de datos. Todo documento *XML* ha de empezar con “?xml versión=1.0” (la codificación no es obligatoria ponerla).
- Elemento `<soap:Envelope>`. Es la raíz de un mensaje *SOAP* propiamente dicho. En este elemento es donde se almacena la información. En las figuras 2.2 y 2.3 encontramos dentro de este elemento el subelemento `<soap:Body>`. Es entre estas etiquetas donde se contiene la carga de datos del mensaje.
- Como se puede observar en la figura 2.2 dentro de `<soap:Body>` aparecen tres *strings*. Cada una de ellas es un parámetro de tipo cadena de texto que necesita el servicio.
- En la figura 2.3, en `<soap:Body>` podemos ver que el servicio retorna al cliente una cadena de texto como resultado.

Existen, además, dos elementos que son opcionales dentro de `<soap:Envelope>`, llamados `<soap:Header>` y `<soap:Fault>`. En `<soap:Header>` se puede almacenar información adicional sobre el mensaje. En caso de aparecer debe hacerlo antes que `<soap:Body>`. Y en `<soap:Fault>` se almacena información sobre errores. Este subelemento sólo puede aparecer una vez en un mensaje *SOAP*.

2.2.5. WSDL (Web Services Description Language)

WSDL es un lenguaje basado en *XML* que permite especificar las características y el funcionamiento de un servicio Web. Un documento *WSDL* detalla los parámetros necesarios para la comunicación entre cliente y un servicio Web. Para poder enviar los mensajes *SOAP*, es necesaria la utilización previa del *WSDL* para obtener una descripción del servicio Web al cual se quiere acceder. A continuación, en la figura 2.4, se muestra la estructura básica de un documento *WSDL*.

```
<definitions>
<types>
    Los tipos de datos.
</types>
<message>
    Las definiciones del mensaje.
</message>
<portType>
    Las definiciones de operación .
</portType>
<binding>
    Las definiciones de protocolo.
</binding>
</definitions>
```

Fig. 2.4 Estructura de un documento *WSDL*

Los elementos básicos de que componen cualquier documento *WSDL* son los siguientes:

- Puertos de *WSDL* (<portType>): Este elemento es el encargado de definir el servicio Web, así como las posibles operaciones y mensajes involucrados.
- Mensajes *WSDL* (<message>): En este elemento se define los tipos de datos que participan en una operación.
- Tipos de datos en *WSDL* (<types>): En este caso lo que se define son los tipos de datos utilizados en el servicio Web.
- Vínculos *WSDL* (<binding>): Especifica el formato del mensaje y protocolo para cada puerto

2.2.6. UDDI (Universal Description, Discovery and Integration)

Para tener acceso a un recurso de Internet es necesario conocer su ubicación. Lo mismo ocurre con los servicios Web. Dada la necesidad de tener un mecanismo para el descubrimiento de servicios Web apareció el *UDDI*. Es un registro público en el cual se puede buscar y publicar información sobre servicios Web. *UDDI* se encarga de describir estos servicios y muestra información asociada. Es como una “agenda de teléfonos” donde el cliente puede localizar el servicio en el que está interesado, y donde una empresa puede registrar los servicios que ofrece. El registro en *UDDI* consta de tres partes:

- Páginas blancas: información de contacto.

- Páginas amarillas: categoría industrial con el fin de poder organizar los servicios Web.
- Páginas verdes: información técnica relativa al servicio.

Una vez el cliente ha encontrado y seleccionado el servicio deseado dentro del registro *UDDI*, se le devuelve una o varias *URL*. Mediante estas *URL* se localizan los servicios Web, por lo que el cliente intenta obtener la descripción del servicio, es decir, el documento *WSDL*. Una vez obtenida esta descripción ya puede acceder al servicio mediante *SOAP*.

2.3. Base de datos

2.3.1. ¿Qué es una base de datos?

Una base de datos es un almacén de información. Es decir, un lugar donde se guardan un conjunto de datos que están relacionados entre ellos. Estos datos podrán ser recuperados en cualquier momento. Además, dispone de herramientas de software para el tratamiento de estos datos.

Se puede acceder una manera concurrente, ya sea por parte de usuarios como de aplicaciones. El software encargado de manejar la base de datos se le denomina *RDBMS*.

Hay diversos tipos de bases de datos, pero este documento se centra en las bases de datos relacionales, que son bases de datos que permiten establecer relaciones entre diversas tablas.

2.3.2. RDBMS (Relational DataBase Management System)

Las aplicaciones nunca manejan directamente la base de datos. Para ello utilizan un software intermedio llamado *RDBMS*. Es a este software donde las otras aplicaciones envían las peticiones y él se encarga de traducirlas en acciones sobre la base de datos. Un ejemplo de *RDBMS* sería *MySQL* u *Oracle*.

2.3.3. Componentes

2.3.3.1. Tablas

Es donde se guarda la información de la base de datos. Las tablas están formadas por filas y columnas. Las columnas son conocidas como campos. Cada campo ha de tener un nombre único y un tipo de datos asociado. A las filas se les llama registros, que es donde se almacena el valor de cada campo.

En cada tabla es obligatorio que haya un campo cuyo valor no puede estar repetido por ningún elemento de esa tabla. Este campo se conoce como *Primary Key* o Clave Primaria, y su función es la de actuar como un identificador único para cada elemento de la tabla. Esta clave primaria no puede tener un valor nulo.

2.3.3.2. Índices

Cuando se trabaja con bases de datos de gran tamaño no es óptimo recorrer toda la tabla para realizar una búsqueda. Generalmente los índices se usan en los campos donde más búsquedas se realicen, ya que su función es la de agilizar las búsquedas de los registros.

Un índice es una estructura de datos extra donde se almacena información de la columna indexada. A la hora de realizar las búsquedas ya no se recorrerá toda la tabla, sino que se hará uso de este índice para agilizar el proceso.

2.3.3.3. Vistas

Una vista es una consulta ya almacenada en la base de datos. Una vista puede ser tratada como si fuera una tabla. Por ello con una vista se puede insertar, borrar, actualizar y seleccionar datos. La diferencia con las tablas es que la vista no almacena los datos, los extrae de una o varias tablas en función de la consulta aplicada.

Las vistas actúan como una tabla virtual, y son útiles para mostrar datos de varias tablas de forma dinámica. También proporcionan seguridad ya que se pueden ocultar ciertos campos al usuario.

2.3.4. SQL (Structured Query Language)

2.3.4.1. Introducción

SQL es un lenguaje utilizado para el acceso a base de datos relacionales. La finalidad de este lenguaje es realizar consultas a la base de datos para obtener la información deseada. Es soportado por la mayoría de motores de bases de datos, aunque a veces existen pequeñas variaciones de este lenguaje según el sistema.

2.3.4.2. Tipos de comandos

Existen dos tipos de comandos SQL:

- *DDL (Data Description Language)*

Los comandos de este tipo se utilizan para definir y manipular estructuras de datos. Mediante sentencias de este tipo podemos crear nuevas tablas, borrarlas o bien modificarlas. Las operaciones básicas dentro de este grupo son: *CREATE*, *ALTER*, *DROP* y *TRUNCATE*.

- *DML (Data Manipulation Language)*

En este caso los comandos son usados para la manipulación de datos. No se puede modificar la estructura de la base de datos pero si la información contenida en ella. Con estos comandos se puede ver todo el contenido de una tabla o bien se puede filtrar. También se pueden añadir o eliminar datos. Los cuatro comandos más importantes dentro de este subconjunto son: *SELECT*, *INSERT*, *DELETE* y *UPDATE*.

2.4. Tecnología GPS (Global Positioning System)

2.4.1. ¿Qué es GPS?

El *GPS (Global Positioning System)* como su nombre indica, es un sistema de posicionamiento global, es decir, un sistema que nos proporciona la ubicación geográfica de un objeto (persona, coche, etc.) sobre la Tierra.

2.4.2. Componentes

Los componentes del Sistema de Posicionamiento Global son los siguientes:

- Red compuesta por 27 satélites, 24 de los cuales están en funcionamiento y 3 son de reserva.
- Estaciones terrestres que se encargan de controlar la trayectoria de los satélites y de las funciones de mantenimiento.
- Dispositivos de recepción GPS, que son los terminales de los cuales se llega a conocer la posición.

2.4.3. Funcionamiento

Para que el dispositivo *GPS* pueda obtener la posición en la que está ubicado, utiliza una red de satélites. Establecerá una comunicación con algunos de estos satélites, concretamente, los que sean visibles para el dispositivo.

Cuanto más elevado sea el número de satélites con los cuales establece una comunicación, la precisión de la posición también aumentará. Los satélites le enviarán al dispositivo la posición de cada uno de ellos. Además, el dispositivo se sincronizará con el *CLOCK* de cada uno de los satélites. Al saber la posición y al tener el *CLOCK* sincronizado, el dispositivo podrá obtener la distancia con cada uno de los satélites mediante el cálculo del retardo de la señal.

Una vez que el dispositivo sabe la distancia con cada uno de los satélites, por triangulación, es capaz de calcular la posición relativa en que se encuentra respecto a los tres satélites. Como el dispositivo también conoce las coordenadas de cada uno de los satélites, es capaz de calcular la posición absoluta en que se encuentra.

2.4.4. Protocolo NMEA

Es el protocolo desarrollado por la *National Marine Electronics Association* que utilizan los dispositivos *GPS* para comunicarse con un determinado programa, que estará instalado en un ordenador o en algún dispositivo móvil (*PDA*, teléfono móvil...).

Los dispositivos *GPS* envían un flujo de mensajes *NMEA* continuamente a la aplicación. Hay varios tipos de mensajes *NMEA*, de los cuales hay que destacar el *GGA* (*Global Positioning System Fix Data*), ya que este mensaje nos proporciona la información más relevante de la forma más abreviada posible, además también nos proporciona información sobre los satélites. A continuación podemos ver un ejemplo de mensaje *GGA*:

- \$GPGGA,080152.955,4116.5199,N,00159.1502,E,1,05,1.9,3.0,M,51.4,M,,0000*58

Los mensajes *NMEA* están formados por caracteres *ASCII* y todos tienen en común que empiezan con el símbolo "\$" seguido de "GP". Los mensajes de control, tienen en común que empiezan por "\$P" seguido de tres caracteres que varían en función del fabricante del chip del receptor. Seguidamente se encuentran tres caracteres para identificar el tipo de mensaje. Cada uno de los campos del mensaje que vienen a continuación está separado por una coma. El único caso en el que no se usa la coma es en el del *checksum*, donde se utiliza un asterisco y está situado al final del mensaje.

A continuación se detallan los campos más relevantes del ejemplo anterior.

- *GGA*: indica el tipo de mensaje, en este caso "Global Positioning System Fix Data".
- 080152.955: indica la hora en la que se recibe el mensaje. La hora correspondería al siguiente formato: 12:34:56.
- 4116.5199, N: indica la Latitud en grados (41) y minutos (16,5199) y la dirección (Norte).
- 00159.1502, E: indica la Longitud en grados (001) y minutos (59,1502) y la coordenada (Este).

En el caso de que el receptor no pueda recibir la señal correctamente, los mensajes *NMEA* que se reciben tendrían el siguiente formato:

- `$GPGGA,073935.026,,,,,0,00,,M,0.0,M,,0000*59`

Como se puede observar en la sentencia *GGA* anterior, los valores de algunos campos, como la latitud y la longitud, están vacíos.

2.5. Podcast

2.5.1. ¿Qué es un podcast?

Un *podcast* o canal *podcast* consiste en la creación, publicación y distribución de archivos a través de Internet. Los archivos ofrecidos habitualmente son de vídeo o audio, aunque no está limitado a ningún tipo de archivo por lo que también se pueden ofrecer, por ejemplo, *podcast* de imágenes o archivos *PDF*. Para la suscripción a un *podcast* es necesario un índice de contenidos (*RSS*) así como los contenidos propiamente dichos.

El *RSS* está almacenado en un servidor. El cliente necesita obtener este *RSS* para saber donde se encuentran los archivos del *podcast*, que no tienen porque estar publicados en el mismo servidor. Una vez el cliente obtiene el archivo *RSS* ya puede descargarse los archivos del *podcast*.

Estos archivos habrán sido creados anteriormente. Por ejemplo, grabar una conversación, y después editarla con algún programa de edición de MP3. Una vez creado el archivo, se alojará en algún servidor de Internet.

2.5.2. Componentes

2.5.2.1. RSS

Un *RSS* (*Really Simple Syndication*) no es más que un índice de contenidos, es un archivo *XML* que contiene las direcciones de Internet donde están alojados los contenidos que se quieren ofrecer. Este formato permite la sindicación de contenidos sin necesidad de utilizar un navegador Web, sino que utiliza un programa capaz de entender el formato *RSS*.

Para que los usuarios puedan acceder a un canal de *podcast*, éste se ha de dar a conocer de alguna manera. Se pueden encontrar en Internet una gran cantidad de páginas Web con diversos *podcast*, donde se permite la publicación y suscripción de *podcast*.

Los archivos *RSS* no tienen un estándar oficial, pero disponen de varias especificaciones. Las más utilizadas son la *0.91*, la *1.0* y la *2.0*.

Para un *RSS* sea válido ha de seguir alguna de las especificaciones anteriores. Estas especificaciones requieren que el archivo *XML* disponga de una serie de campos obligatorios.

```
<?xml version="1.0" encoding="UTF-8" ?>
<rss version="2.0">
<channel>
  <title>Xarxes i Serveis</title>
  <link>http://IP/RSS/XarxesiServeis.xml</link>
  <description>Documents docents sobre XS en format PDF</description>
  <item>
    <title>PDH</title>
    <link>http://IP/UPCMobile/Podcast/XarxesiServeis/PDH.pdf</link>
    <description>Transparencies de PDH</description>
  </item>
</channel>
</rss>
```

Fig. 2.5 Estructura de un *RSS*

En la figura anterior se puede observar un *podcast* que contiene un archivo *PDF* de una asignatura de la EPSC.

El *RSS* dispone de un encabezado *XML* seguido de la versión de la especificación de *RSS* que se ha utilizado, en este caso es la 2.0.

Según la especificación 2.0, cada archivo *RSS* está formado por una etiqueta *<channel>* que proporciona información acerca del *podcast*. Los elementos obligatorios que ha de contener el canal son los siguientes: el título del canal, el enlace donde se encuentra el *RSS* y una descripción del canal. Además por cada archivo que contenga este canal, habrá un elemento *<item>* dentro de la etiqueta *<channel>*.

Para cada *<item>* es obligatorio que éste contenga su título, el enlace donde se puede encontrar este archivo y una descripción.

Cuando se quiera añadir un nuevo elemento a un canal, bastará con modificar el archivo *RSS* añadiendo un *<item>* más y publicando el nuevo archivo en Internet.

2.5.2.2. Agregador

El agregador es un programa que se inicia en el dispositivo cliente y es capaz de gestionar uno o varios archivos *RSS* así como sus contenidos. Permite que el cliente se suscriba a los diferentes canales de *podcast*. Otra función del agregador es avisar al cliente cuando se produzcan actualizaciones en alguno de los canales a los cuales está suscrito. Para ello, el agregador realiza consultas al *RSS* y si detecta que se ha añadido algún nuevo elemento lo descargará para su posterior reproducción o almacenamiento. En cualquier

momento el usuario podrá cancelar la suscripción de un canal, simplemente tendrá que quitar del agregador dicho canal.

Tanto la actualización de los archivos *RSS* como la descarga de sus elementos se pueden hacer de forma manual o automática, dependiendo del software utilizado. Si es automática, será el propio *agregador* el que se encargue de descargar los nuevos contenidos cada vez que se publiquen.

CAPÍTULO 3. DISEÑO DEL PROYECTO

Este capítulo está estructurado en tres grandes apartados. En el primero de ellos se comenta el *software* utilizado para cada parte del sistema y la razón por la cual se ha escogido. Además, se comentan las diferentes alternativas.

En el siguiente apartado se muestra el diseño del modelo del sistema y su evolución a lo largo del desarrollo del proyecto.

En el anexo 1 se podrán encontrar los diagramas de clases y de secuencia de la aplicación y en el anexo 2 se explica el portal Web.

3.1. Elección de software

3.1.1. Entorno de trabajo: Visual Studio .NET

Uno de los requisitos de este trabajo es utilizar el entorno *.NET*. Por ello se han utilizado las opciones que la propia *Microsoft* ofrece con este fin: *.NET Frameworks* y *Visual Studio .NET*.

La versión de *Visual Studio* utilizada para la elaboración de este proyecto ha sido la 2005.

La ventaja que nos proporciona este *software* es que nos permite una programación fácil y rápida en el entorno *.NET*. Además, esta versión incluye las herramientas necesarias para trabajar con dispositivos móviles (*Compact Framework*).

Al inicio de este proyecto se utilizó la versión 2003 pero debido a problemas de conectividad con la base de datos migramos al 2005.

El lenguaje de programación utilizado para el desarrollo del proyecto ha sido *Visual Basic .NET*, ya que es el lenguaje más apropiado para trabajar con un entorno gráfico de ventanas y menús.

3.1.1.1. Instalación

Para la instalación de este entorno de trabajo es necesario cumplir con los siguientes requisitos:

- Tamaño en disco duro: 4Gb (con archivos de ayuda incluidos).
- Memoria *RAM*: 256 MB.
- Sistema operativo *Windows 2000 Service Pack 4*, *Windows XP Service Pack 2* o *Windows Server 2003*.

- *.NET Framework.*

Además del propio *Visual Studio* también se puede instalar una completa ayuda de este software. Aunque no es estrictamente necesaria su instalación se recomienda tenerla para cualquier consulta que se quiera realizar.

En este proyecto se ha optado por la instalación completa de todos los componentes.

3.1.1.2. Entorno de trabajo

Al iniciar *Visual Studio 2005* la primera pantalla que aparece es la página de inicio, en donde se puede escoger el proyecto que se desea abrir o bien crear uno nuevo.

En la figura 3.1 se muestran los diferentes elementos que componen el entorno de trabajo de *Visual Studio*.

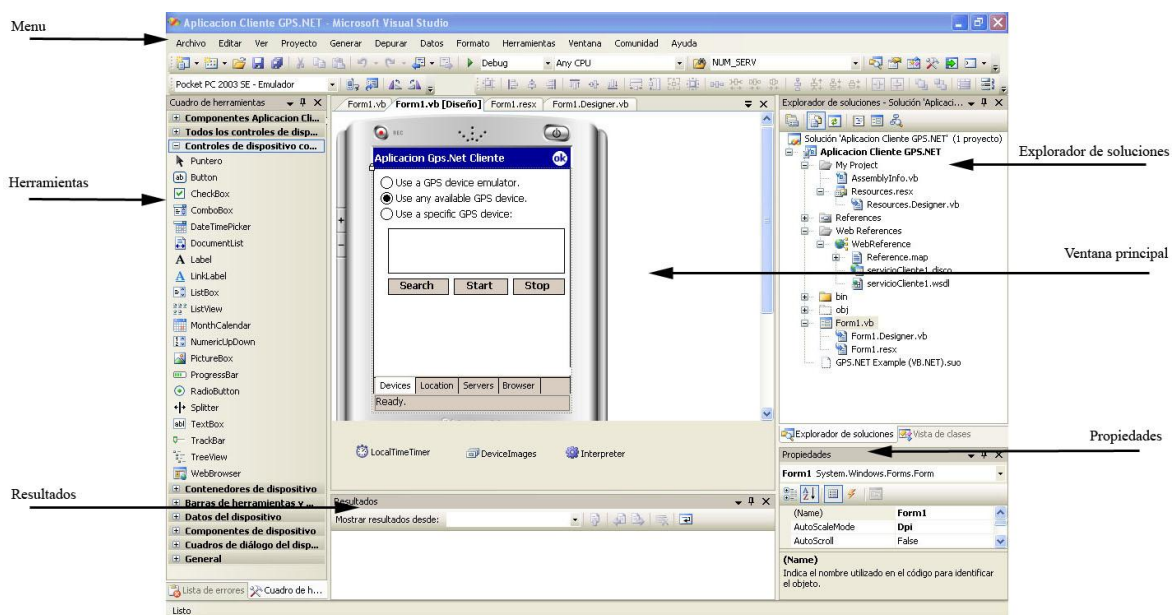


Fig. 3.1 Entorno de trabajo Visual Studio

- **Menú:** está compuesto por varios menús desplegables con las diversas opciones que dispone este *software* como por ejemplo *Archivo*, *Editar* o *Ayuda*.
- **Explorador de soluciones:** en esta ventana se muestra las diferentes clases, módulos, formularios, referencias Web, etc. del proyecto que está abierto. Además de los elementos que lo componen, también muestra como están estructurados.

- **Herramientas:** contiene los controles que se pueden agregar a una interfaz de usuario, ya sea un formulario o una página Web.
- **Propiedades:** muestra las propiedades de un elemento seleccionado. También permite modificar estas propiedades.
- **Resultados:** en esta ventana se puede observar el resultado de acciones diversas, tales como compilar o ejecutar.
- **Ventana principal:** zona principal de trabajo. Esta organizada en pestañas. Dentro de cada pestaña muestra el contenido de los diferentes archivos que forman parte del proyecto y que aparecen en el *explorador de soluciones*.

3.1.1.3. Alternativas

Existen varias alternativas a *Visual Studio .NET*. Fuera del ámbito de *Microsoft* se pueden destacar dos: *SharpDevelop* y *MonoDevelop*.

SharpDevelop es un entorno gráfico de desarrollo para la plataforma *.NET* ofrecido por *IC#Code*. Permite el desarrollo de software en *C#*, *Visual Basic .NET* y *Boo*. También se pueden importar proyectos creados con *Visual Studio .NET*. Para su instalación es necesario el sistema operativo *Windows* además de *.NET Framework 2.0*, aunque también soporta el *1.1*, pero puede ocasionar algunos problemas con este último.

MonoDevelop es un entorno gráfico para la plataforma *.NET* para *Linux*. Ha sido desarrollado por *Novell* y la Comunidad *Mono*. Los lenguajes que implementa son principalmente *C#* y también *Nemerle*, *Boo* y *Java*. Sólo puede ser ejecutado en sistemas operativos *Mac* o *Linux*.

La principal ventaja de estas dos opciones respecto a *Visual Studio* radica en que son totalmente gratuitas. En contra, hay que decir que no son tan completas como la herramienta de *Microsoft*.

Microsoft también ofrece la opción de utilizar un IDE gratuito. Se conoce como *Visual Studio Express Editions*. Estas ediciones vienen separadas en función del lenguaje de programación que se quiera utilizar. El período para su uso es de un mes e incluyen gran parte de las funcionalidades de la versión completa, aunque no todas.

3.1.2. Servidor: IIS (Internet Information Services)

IIS es el servidor Web de la plataforma *Windows*. Hoy en día es uno de los servidores más utilizados. Viene incluido en el sistema operativo *Windows (NT, 2000 Server, 2003 Server y XP Profesional)*. Además este servidor permite trabajar tanto con servicios Web como con *ASP.NET* (páginas y servicios Web).

3.1.2.1. Instalación

Para realizar la instalación del IIS en *Windows XP Profesional* es necesario ir a *Inicio/Panel de Control/Agregar o quitar programas*. Seguidamente hay que seleccionar *Agregar o quitar componentes de Windows*. A continuación se carga una nueva ventana con una lista donde se muestran los componentes disponibles, tanto los instalados como los que no. Aparecerá el componente con el nombre *Servicios de Internet Information Services (IIS)*. Después de seleccionarlo y presionar el botón *Siguiente* se procede a su instalación.

Se recomienda instalar el IIS en un sistema operativo *Windows Server 2003*. *Windows XP* limita el número de clientes que se pueden conectar al servidor IIS, mientras que *Windows Server 2003* no está limitado.

3.1.2.2. Directorios virtuales

Para una mejor gestión del servidor Web es recomendable disponer de directorios separados para los elementos que se quieran publicar, ya que si no todos los archivos estarán en la carpeta raíz del servidor Web (*wwwroot*).

Para crear nuevos directorios dentro de *wwwroot*, no basta con crear una nueva carpeta y publicar en ella los contenidos, sino que hay que informar al IIS que esta nueva carpeta es un directorio virtual. En caso contrario el cliente no podrá acceder a estos contenidos.

La configuración del IIS se podrá realizar desde *Inicio/Panel de control/Herramientas Administrativas/Servicios de Internet Information Server* como se puede ver en la figura 3.2.

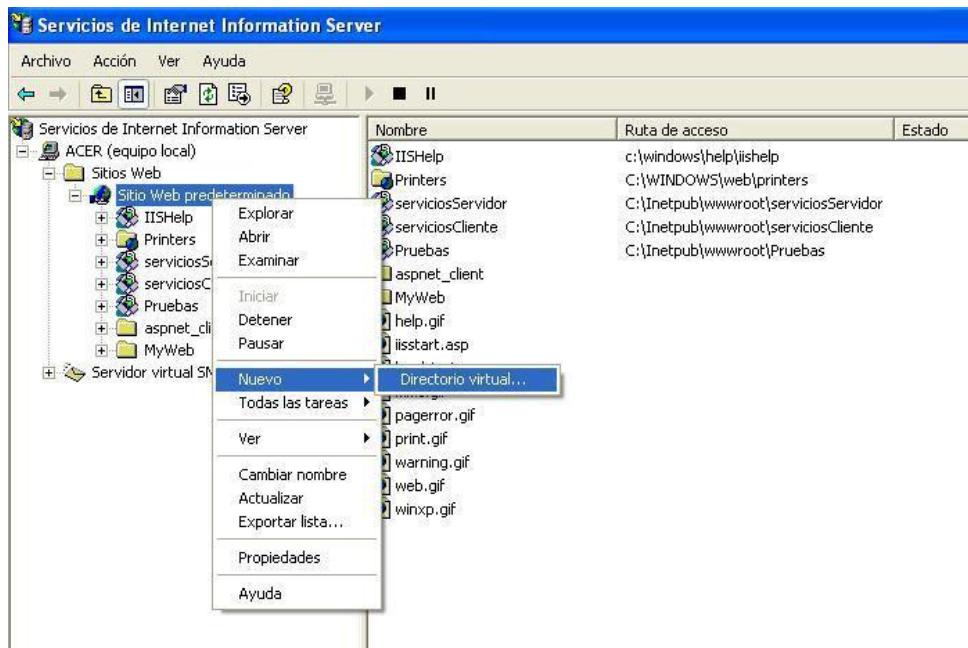


Fig. 3.2 Directorios virtuales

Para agregar un nuevo directorio virtual hay que hacer clic derecho sobre el sitio Web donde se quiera añadir el directorio, tal y como muestra la figura anterior.

Una vez creado el directorio virtual, ya podremos publicar en él los contenidos Web.

3.1.2.3. Alternativas

Como servidor Web alternativo a *IIS* podemos encontrar diversas posibilidades, pero si lo que se pretende es que este servidor interactúe con *.NET* las opciones se reducen a sólo una; el servidor *HTTP Apache*.

El servidor Web *Apache* ha sido desarrollado por *Apache Software Foundation*. Es de código abierto, multiplataforma y gratuito. Según una estadística publicada por *Netcraft* (<http://news.netcraft.com/>) es el servidor más utilizado hoy en día. Los resultados de esta estadística se muestran en la figura 3.3.

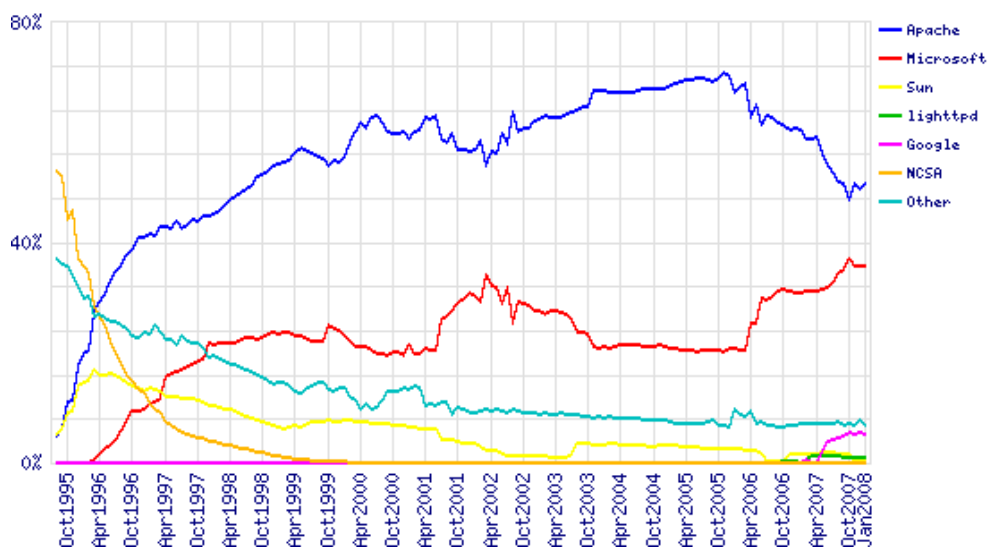


Fig. 3.3 Servidores más utilizados según *Netcraft*

También se puede observar que *IIS* de *Microsoft* es el segundo servidor más utilizado.

Otro estudio parecido ha sido realizado por *Google*. El artículo entero lo podemos encontrar en <http://googleonlinesecurity.blogspot.com/2007/06/web-server-software-and-malware.html>. En este caso los resultados obtenidos son los siguientes:

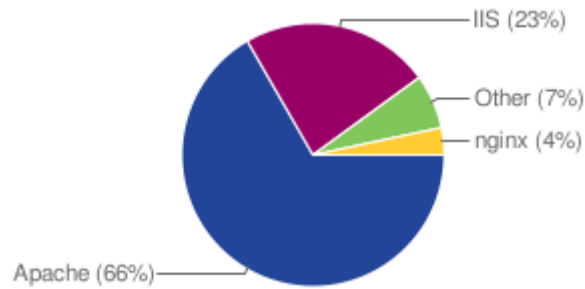


Fig. 3.4 Servidores más utilizados según Google

En este caso los porcentajes son similares a los de *Netcraft*. Se puede extraer la misma conclusión que en la estadística anterior, *Apache* es el servidor Web más utilizado seguido por *IIS*.

Dado que *Apache* es el servidor más extendido fue la primera opción que se evaluó para comprobar si se adecuaba a las necesidades del proyecto.

Por un lado es necesario el uso de servicios Web. *Apache Software Foundation* dispone de una serie de herramientas para implementar servicios Web llamada *Apache Axis*.

Por otro lado también se requiere el uso de páginas *ASP.NET* ya que se desea poder mostrar contenidos dinámicos en los dispositivos móviles. *Apache* dispone de un módulo llamado *mod_aspdonet*, pero este módulo ha sido retirado tal y como se indica en su página Web (http://httpd.apache.org/modules/#mod_aspdotnet), por lo que el uso de este servidor quedó descartado.

3.1.3. Base de Datos: MySQL

La base de datos que se va a utilizar en este proyecto va a ser pequeña, es decir con pocos registros. También se requiere que las consultas sean lo más rápidas posibles para evitar cualquier tipo de retardo no deseado en la aplicación cliente. Estas dos premisas son las que han condicionado la elección en este punto.

Se ha decidido utilizar *MySQL* como gestor de base de datos ya que cumple con todas las características necesarias. Este software ha sido desarrollado por *MySQL AB*. Este gestor es totalmente gratuito y además es uno de los más utilizados en el ámbito de aplicaciones Web.

3.1.3.1. Instalación

La descarga de este software se puede realizar desde su página oficial <http://www.mysql.com>.

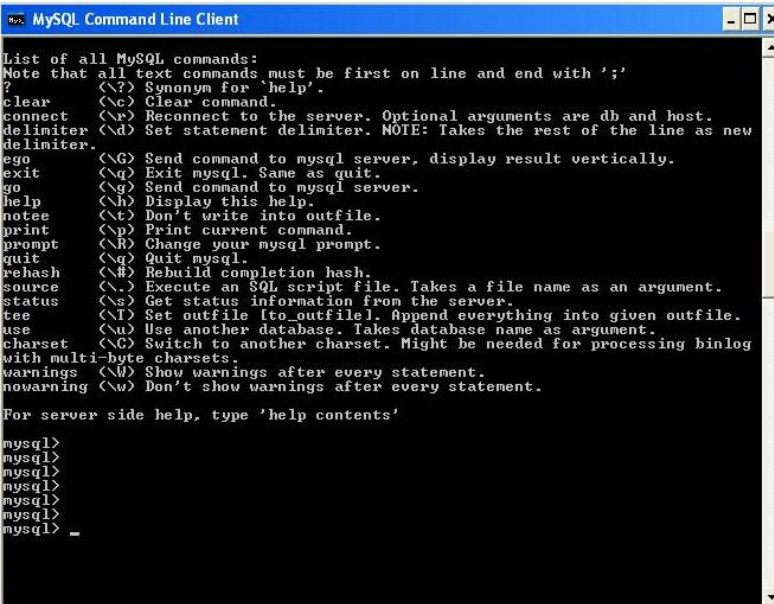
Para usar este gestor es necesaria la instalación de dos componentes.

- *MySQL Server*. Este programa es el servidor de base de datos.
- El *MySQL connector .NET* para poder trabajar con *.NET* y *MySQL* conjuntamente. Sin este conector, *Visual Studio* no puede realizar una conexión con una base de datos *MySQL*.

Opcionalmente se puede instalar una interfaz gráfica para la gestión de la base de datos que viene incluida en *MySQL GUI Tools*.

3.1.3.2. Entorno de trabajo

Si se ha instalado el *MySQL GUI Tools* se podrá utilizar una interfaz gráfica, *MySQL Query Browser*, para la gestión de la base de datos como se muestra en la figura 3.6. En caso de no querer utilizar esta interfaz, el entorno de trabajo sería por línea de comandos, como se muestra en la figura 3.5.



```
MySQL Command Line Client

List of all MySQL commands:
Note that all text commands must be first on line and end with ';'
?          (? ) Synonym for 'help'.
clear      (c)  Clear command.
connect    (r)  Reconnect to the server. Optional arguments are db and host.
delimiter (d)  Set statement delimiter. NOTE: Takes the rest of the line as new
delimiter.
ego        (G)  Send command to mysql server, display result vertically.
exit       (g)  Exit mysql. Same as quit.
go         (g)  Send command to mysql server.
help       (h)  Display this help.
notee     (t)  Don't write into outfile.
print      (p)  Print current command.
prompt     (R)  Change your mysql prompt.
quit       (q)  Quit mysql.
rehash     (#)  Rebuild completion hash.
source     (.)  Execute an SQL script file. Takes a file name as an argument.
status     (s)  Get status information from the server.
tee        (T)  Set outfile [to_outfile]. Append everything into given outfile.
use        (u)  Use another database. Takes database name as argument.
charset    (C)  Switch to another charset. Might be needed for processing binlog
with multi-byte charsets.
warnings   (W)  Show warnings after every statement.
nowarning  (w)  Don't show warnings after every statement.

For server side help, type 'help contents'

mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
```

Fig. 3.5 Línea de comandos de *MySQL Server*

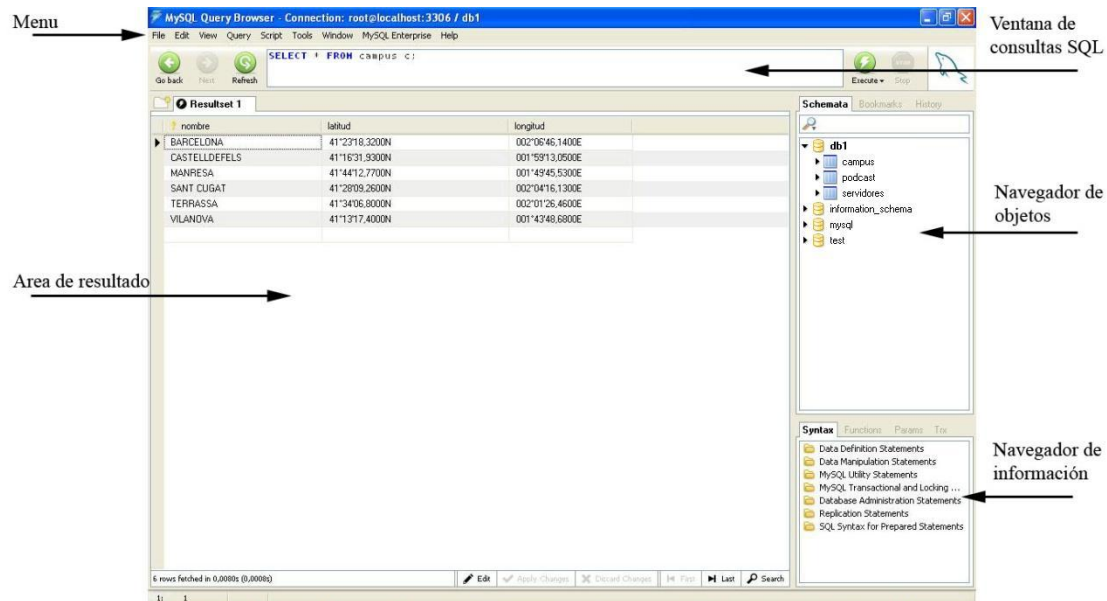


Fig. 3.6 MySQL Query Browser

En la ventana de consultas se escriben las sentencias SQL que se quieran realizar. Una vez ejecutadas, el resultado de las mismas se pueden observar en el *área de resultado*. Concretamente en la figura 3.6., la sentencia introducida ha sido “*SELECT * FROM campus c*” retornando como resultado toda la información contenida en la tabla *campus*.

En el *navegador de objetos* se muestran las diferentes bases de datos y las tablas que las componen. En la figura 3.6., se puede observar que la base de datos *db1* dispone de tres tablas, *campus*, *servidores* y *podcast*.

Por último, cabe destacar que dentro del navegador de información se puede hallar todo tipo de documentación relativa al lenguaje SQL, que incluye tanto la sintaxis como las funciones.

3.1.3.3. Creación de tablas

Para crear una nueva tabla hay que acceder al a la opción “*Create New Table*”, que aparecerá cuando se haga clic derecho en alguna de las bases de datos listadas por el *navegador de objetos*.

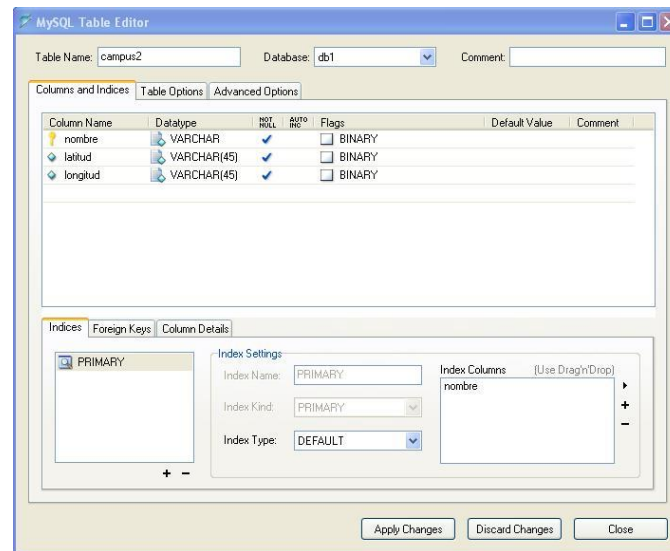


Fig. 3.7 Creación de una tabla mediante *MySQL Query Browser*

Para cada columna se especifica el nombre del campo y el tipo de dato que contendrá. En la figura 3.7., se han declarado todos los campos como cadenas de texto (*VARCHAR*). El campo *nombre* se ha establecido como clave primaria, por lo que ningún registro de esta tabla podrá tener el mismo nombre.

La sentencia SQL equivalente para crear esta tabla es la siguiente:

```
CREATE TABLE `db1`.`campus2` (`nombre` VARCHAR NOT NULL, `latitud` VARCHAR(45) NOT NULL, `longitud` VARCHAR(45) NOT NULL, PRIMARY KEY (`nombre`))
```

Mediante el uso de *MySQL Query Browser*, es más visual e intuitivo la creación y gestión de bases de datos que si se realizara mediante la línea de comandos.

3.1.3.4. Alternativas

Actualmente existen muchos gestores para bases de datos. Se pueden encontrar opciones muy completas, tanto dentro del *software* libre como en el de pago. Dentro del *software* de pago destacan *Microsoft SQL Server* y *Oracle*.

Otra alternativa muy diferente de las anteriores es el uso de *XML*. Los documentos *XML* permiten almacenar la información de forma estructurada, por lo que pueden hacer la función de base de datos. El problema de esta opción es la velocidad en la consulta. Según el tipo de consulta realizada puede variar mucho. Si se muestran los datos del mismo modo que están almacenados es muy rápido, más rápido que cualquier base de datos relacional. En caso de hacer una búsqueda por algún criterio y/o ordenar el resultado es cuando la velocidad de la consulta disminuye mucho, haciendo este sistema más lento que las otras alternativas.

3.1.4. GPS: software propio

La introducción del *GPS* en el proyecto implicaba el uso de un *software* capaz de interpretar la señal de *GPS* recibida. Este *software* además de poder comunicarse con el dispositivo *GPS*, debía de ser compatible con *.NET*.

La primera opción que se barajó fue el uso de *GPS.NET*. Incluso se llegó a realizar una versión de prueba de la aplicación cliente con este *software* integrado. Posteriormente se decidió que lo más adecuado era usar un *software* propio, ya que no se pretendía utilizar *software* de pago. Gracias al uso y estudio de *GPS.NET* adquirimos los conocimientos necesarios para desarrollar nuestro propio *software* para *GPS*.

3.1.4.1. Alternativas

La primera opción para interpretar los datos del receptor *GPS* fue utilizar *GPS.NET*, que es un componente para *Visual Studio .NET*. *GPS.NET* proporciona librerías que incluyen las clases y métodos necesarios para administrar y mostrar la información recibida de un satélite *GPS* en un dispositivo móvil. Estos métodos, también realizaban la conexión con el dispositivo *GPS* e interpretaba los mensajes *NMEA* automáticamente.

GPS.NET ha sido desarrollado por la empresa *GeoFrameworks*, por lo que es necesaria una licencia para hacerlo funcionar. Esta licencia hay que comprarla a la empresa *GeoFrameworks*. Aun así, en su Web se pueden encontrar versiones de prueba (*trial version*) que disponen de unas licencias que caducan al cabo de un mes desde que se descarga.

3.1.5. Podcast: software propio

Este proyecto requería que el dispositivo móvil cliente fuera capaz de gestionar un *podcast*. Para ello era necesario el uso de algún programa que le permitiera realizar esta gestión. Los requisitos deseados para este programa *podcast* son los siguientes:

- Gestión de canales.
- Actualización automática y manual de canales.
- Suscripción a canales.
- Descarga automática de contenidos.
- Descargas simultáneas.
- Gestión de contenidos descargados.
- Reproductor integrado.

Existen diversos programas de gestión de *podcast* para dispositivos móviles, algunos de los cuales son de libre distribución. Se han realizado pruebas con estas aplicaciones para determinar si se podría utilizar alguna de ellas en este proyecto.

Finalmente, dado que las aplicaciones estudiadas no se adecuaban a los requisitos anteriormente mencionados, se optó por el diseño e implementación de un *software* propio.

3.1.5.1. Alternativas

Se han encontrado dos programas que cumplían parte de los requisitos deseados, *ppcPodcast* y *AudioPod*.

El *software ppcPodcast* se puede encontrar en la siguiente dirección: <http://ppcpodcast.sourceforge.net/>. Funciona en dispositivos con *Windows CE* (Sistema operativo *Windows* para dispositivos móviles) tanto *Pocket PC* como *Smart Phones*. Esta aplicación permite la suscripción a canales *podcast*, así como gestionarlos y actualizarlos de forma automática. También dispone de hilos de ejecución simultáneos, que permiten varias descargas de contenidos al mismo tiempo. Los archivos se pueden descargar de forma manual o automática. En este último caso sólo para las novedades encontradas en un canal. No dispone de reproductor integrado por lo que es necesario depender de otra aplicación externa para la visualización de contenidos.

Por otro lado, tenemos *Audiopod*, disponible en <http://www.avaricum.net/>. Esta aplicación es para *Pocket PC*. Como en el caso anterior, permite la agregación y gestión de canales *podcast*. La actualización de las fuentes se realiza de manera automática. En cambio para la descarga de contenidos se ha de llevar a cabo de forma manual. Al igual de *ppcPodcast* no incorpora un reproductor integrado. La gestión de ficheros descargados no se puede realizar a través de esta aplicación, sino que es responsabilidad del usuario.

Aunque las dos alternativas comentadas son muy completas no acaban de cumplir los requisitos necesarios para este proyecto

En el caso de *ppcPodcast* encontramos que es una aplicación muy completa ya que se ajusta en gran medida a los requisitos deseados. El principal inconveniente es que depende de una aplicación externa ya que no incorpora un reproductor propio. Además este *software* presenta algunos fallos graves, como por ejemplo, no poder cancelar las descargas activas o bien se quedaba bloqueado en diferentes ocasiones. Por estos motivos fue descartado.

AudioPod no permite la descarga automática de contenidos. Esta aplicación está diseñada exclusivamente para archivos de audio. Tampoco se pueden gestionar los archivos descargados. Es por ello que también fue descartado.

3.1.6. Diseño global

Finalmente en lo que respecta al *software* utilizado se puede resumir en la siguiente lista:

- Marco de trabajo: *Visual Studio.NET 2005*
- Servidor: *Internet Information Services (IIS)*

- Base de datos: *MySQL*
- *GPS*: *Software propio*
- *Podcast*: *Software propio*
- Dispositivo móvil: *Windows CE con conexión inalámbrica*

Destacar que en todos los casos que ha sido posible se ha utilizado el software libre y/o propio con tal de reducir los costes de este proyecto con el fin de enfocarlo todo lo posible a un ámbito comercial.

Como era necesario diseñar una aplicación para el *GPS* y para *podcast*, se decidió implementarlo en una única aplicación.

Un requisito que deberá cumplir el dispositivo móvil para poder ejecutar la aplicación cliente, deberá ser que éste disponga de un sistema operativo *Windows CE*. Esto es necesario para que el dispositivo pueda ejecutar la aplicación ya que esta está programada en *.NET*. Además el cliente deberá de disponer de conexión a Internet y de un receptor *GPS*.

3.2. Diseño del modelo

En este capítulo se explica el modelo que se ha usado para realizar este TFC, es decir, cada una de las partes del proyecto y cómo interactúan entre ellas.

3.2.1. Descripción del sistema

El objetivo final de este proyecto es ofrecer un servicio a todos los alumnos de la UPC. Los estudiantes accederán a estos servicios a través de un servidor de su campus más próximo.

Los servicios ofrecidos serán el consumo de *podcast* y el acceso a un portal Web. Mediante estos servicios se pretende distribuir material docente relativo a la universidad UPC.

También ha sido necesario el desarrollo de una aplicación cliente capaz de conectarse al dispositivo *GPS* y gestionar la información relativa al servicio de *podcast*. Además, se han implementado unos servicios Web para que los administradores puedan gestionar la base de datos.

En la siguiente figura se muestra un esquema completo del sistema.

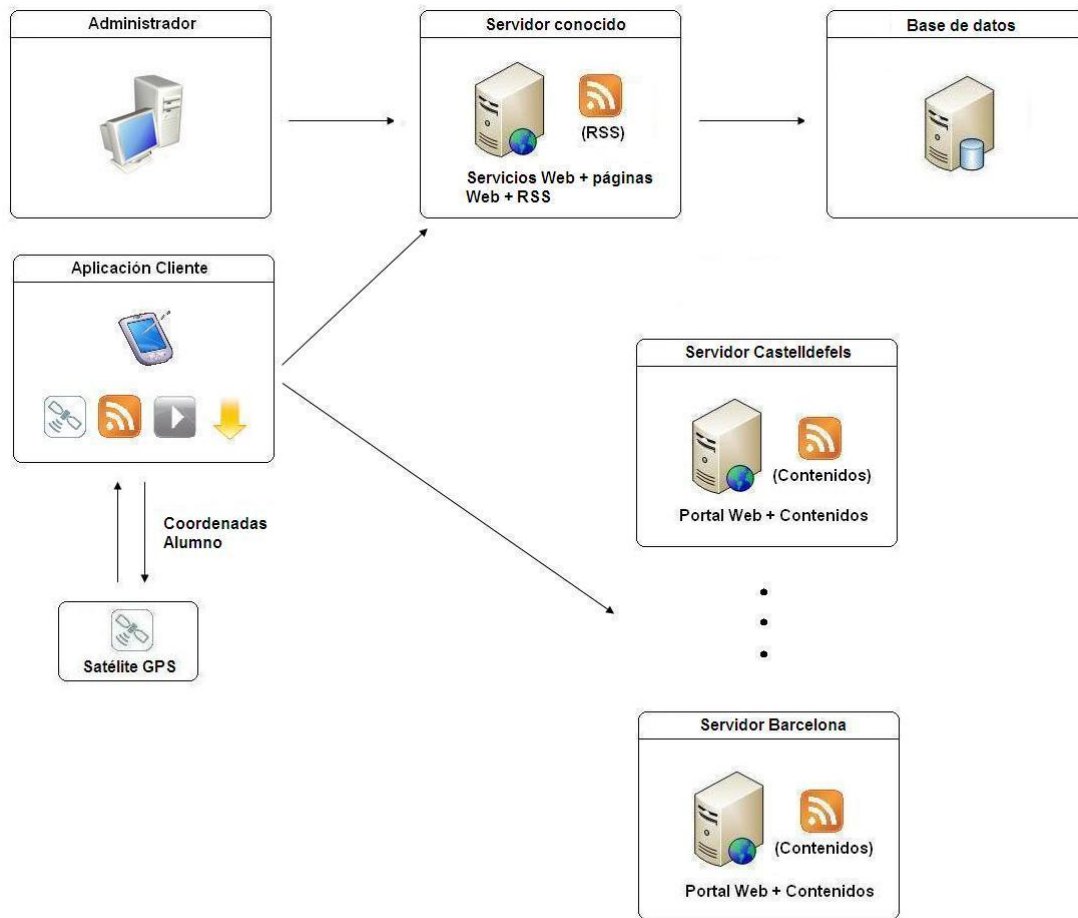


Fig. 3.8 Esquema del sistema

En el esquema anterior se pueden observar los elementos que van a formar parte de este sistema. Estos elementos son: un administrador, una *PDA* de un estudiante con la aplicación cliente instalada, los servidores de los diferentes campus, la base de datos y un servidor conocido.

El servidor conocido es un servidor que dispone de una dirección *IP* pública, que es conocida por los administradores y por la aplicación cliente.

El sistema se puede dividir en tres apartados.

- Gestión de la BD.
- Conexión del dispositivo cliente al servidor.
- Servicios cliente.

A continuación se explica el funcionamiento de cada una de las partes del sistema.

3.2.2. Gestión de la base de datos

El objetivo de este módulo es que los administradores puedan registrar y borrar servidores de los campus de la BD.

La base de datos que utilizan los administradores dispone de dos tablas. Una de ellas es para almacenar la información referente a los campus (nombre del campus y coordenadas geográficas) y la otra es para almacenar la información de los servidores que estén disponibles en cada campus (nombre del servidor, URL y campus al cual pertenece).



Fig. 3.9 Esquema de la gestión de la base de datos

La gestión de esta base de datos la realizan los administradores a través de unas páginas Web. Estas páginas llaman a un servicio Web llamado *ServicioAdministrador*, que se encarga de realizar las consultas pertinentes a la BD.

Estos servicios Web tienen la función de facilitar la gestión de la BD al administrador, ya que de esta forma no necesita conocer el lenguaje SQL para realizar modificaciones en las tablas. Las páginas Web tienen la función de mejorar la vista del servicio Web.

Las páginas Web están publicadas en un servidor conocido, por lo que los administradores pueden tener acceso a ellas desde cualquier ordenador. En la figura 3.10 se muestra la página de inicio a la que tendrán que acceder los administradores para poder realizar alguna acción sobre la base de datos.



Fig. 3.10 Página Web de inicio de los administradores

En la página de inicio se pueden encontrar cuatro enlaces a otras páginas Web. Dependiendo de la acción que el administrador quiera realizar, se dirigirá a una Web o a otra.

- Dar de alta Servidor

Mediante esta página Web el administrador puede dar de alta un servidor en el sistema. El administrador deberá introducir ciertos parámetros para dar de alta a un servidor. Estos parámetros son el nombre del servidor, la *URL* y el campus a que pertenece. Una vez introducidos estos parámetros se intentarán introducir estos datos en la BD. Si el proceso ha concluido con éxito, el servicio Web retornará un *XML* donde aparecerá la cadena de texto: "Proceso completado". Si no ha concluido con éxito, en el *XML* que retorna el servicio se podrá leer "Ha ocurrido un Error".

- Dar de baja Servidor

Si un administrador quiere dar de baja a un servidor, deberá realizar un proceso muy similar al de registro. El administrador deberá introducir el nombre del servidor que quiere dar de baja, su *URL* y el campus donde está ubicado. Como en el caso anterior, el servicio Web mostrará un mensaje indicando el resultado de la operación.

- Mostrar servidores

Desde la página de inicio, si un administrador pulsa sobre la opción "Mostrar Servidores", se cargará una nueva página Web. En este caso, el servicio Web le retornará un *XML* con todos los registros almacenados en la BD de la tabla "servidores". El administrador podrá visualizar el nombre de cada uno de los servidores que estén registrados, junto con su *URL* y su campus.

- Mostrar campus

El administrador podrá ver también los campus con sus respectivas coordenadas (latitud y longitud) mediante la página Web de "Mostrar Campus".

3.2.3. Conexión del dispositivo cliente con el servidor de un campus

La finalidad de este bloque es establecer una comunicación entre el dispositivo móvil del alumno y un servidor ubicado en alguno de los campus de la UPC.

Para ello, desde la aplicación cliente se ofrecen dos alternativas; conectarse al servidor más próximo geográficamente, o bien conectarse directamente a los servidores de un campus especificado por el usuario.

3.2.3.1. Conexión al servidor más cercano

La finalidad de esta sección es establecer una comunicación entre el dispositivo cliente y el servidor del campus más cercano al alumno.

En la figura 3.11 se muestra como se realizará esta conexión.

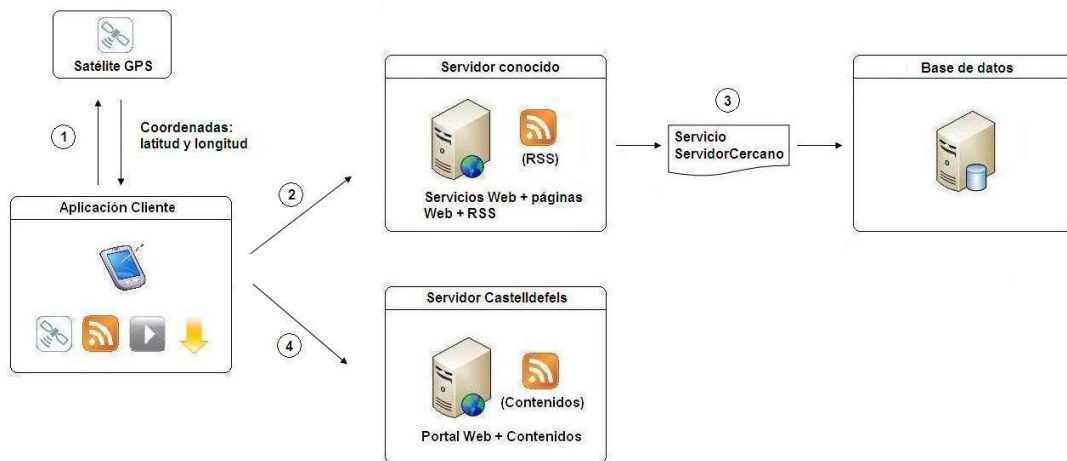


Fig. 3.11 Esquema de conexión con el servidor más próximo

El cliente es un dispositivo móvil, por lo que podrá moverse de un lado a otro y cambiar su posición. Lo primero que deberá hacer el cliente será conseguir su ubicación actual. Para ello necesitará disponer de un receptor *GPS*, ya sea interno (incluido en el propio dispositivo móvil) o externo (receptor *GPS* conectado al dispositivo cliente mediante un puerto *COM*).

La aplicación cliente se encargará de buscar el puerto por el que le llegan los mensajes *NMEA* que le envía el receptor *GPS*. Para ello escuchará por todos los puertos hasta que encuentre alguno por el que llegue la señal. Esto es necesario puesto que los mensajes *NMEA* no siempre se envían al mismo puerto.

Una vez descubierto el puerto, la aplicación será capaz de leer los mensajes *NMEA* y, por lo tanto podrá extraer la latitud y la longitud, es decir, conseguirá saber la posición del alumno.

Una vez que la aplicación conoce la posición del dispositivo, accederá a un servicio Web. Este servicio tiene el nombre de *ServicioSevidorCercano*. La aplicación cliente le pasará al *ServicioSevidorCercano* la latitud y la longitud del dispositivo. La aplicación es capaz de acceder a este servicio Web ya que está publicado en un ordenador el cual sabe su dirección *IP*.

Este servicio se conectará a la BD y mediante una consulta a la tabla "campus", conseguirá los nombres y las posiciones en la que están cada uno de los campus. Seguidamente calculará la distancia que separa al cliente de cada uno de los campus. Esto es posible ya que el servicio conoce las coordenadas de cada uno de ellos.

Una vez el servicio encuentra el campus más próximo al dispositivo, se volverá a conectar a la BD y realizará una consulta sobre la tabla "servidores", consiguiendo así una lista de los servidores ubicados en el campus más cercano al cliente.

El servicio le devolverá a la aplicación hasta un máximo de tres servidores activos. La aplicación iniciará la conexión con el primero de los servidores. En caso de que falle esta conexión, se intentará acceder al siguiente servidor de la lista.

Si no hay ningún servidor disponible en el campus más cercano al alumno, el *ServicioServidorCercano* le devolverá los servidores del segundo campus más cercano. Si no fuera posible conectarse a ninguno, la aplicación mostraría al usuario un mensaje de error.

Además cada cierto tiempo definido por el usuario, la aplicación realiza de nuevo el proceso descrito anteriormente. De esta forma, si la posición del estudiante varía de forma que el campus más cercano cambie, la aplicación se podrá conectar a este nuevo campus.

Una vez el cliente está conectado a un servidor podrá realizar las descargas de los archivos de *podcast* y también podrá acceder al portal Web.

Todas las descargas se realizarán a través del servidor más próximo. De esta forma el tiempo de descarga es mínimo. Por ello hasta que no se encuentre señal *GPS* no se podrán iniciar descargas.

3.2.3.2. *Conexión a un campus directamente*

En este caso el objetivo es establecer la comunicación entre el cliente y un servidor de un campus especificado por el alumno, sin hacer uso del *GPS*.

Si el dispositivo móvil no recibe la señal *GPS* correctamente, la aplicación se queda en espera hasta que encuentre señal. Es por ello que se ofrece la opción de conectar directamente con los servidores de un campus.

Si se utiliza la conexión directa, la conexión con el servidor se realizará antes que utilizando *GPS* ya que este último requiere un tiempo de configuración. En cambio, las descargas pueden tardar más en realizarse ya que el campus seleccionado por el estudiante puede que no sea el más cercano.

En la figura 3.12 se muestra como se realiza la conexión de manera directa cuando un estudiante selecciona el campus de Castelldefels.

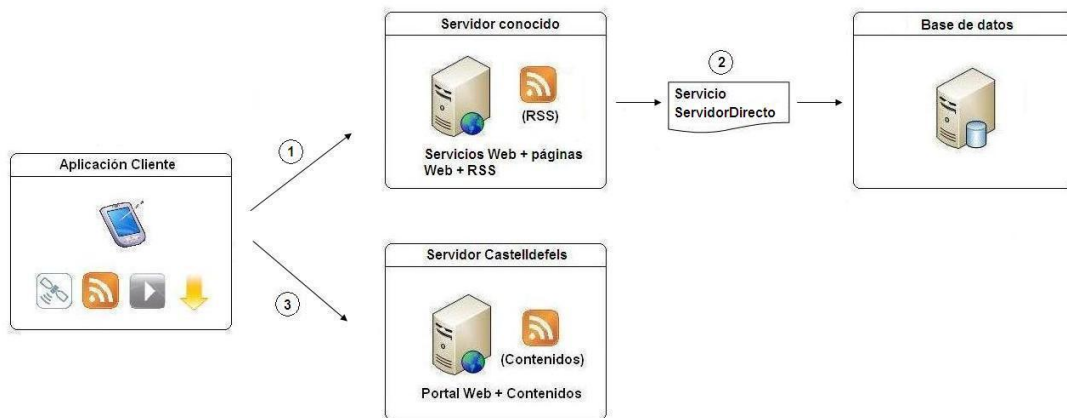


Fig. 3.12 Esquema de conexión directa con un servidor

Si el alumno decide conectarse directamente, la aplicación invocará a un servicio Web llamado *ServicioServidorDirecto*. La aplicación le pasará como parámetro a este servicio el nombre del campus al cual se desea conectar.

El *ServicioServidorDirecto* se conectará con la base de datos para averiguar los servidores activos relativos al campus seleccionado por el usuario. Estos servidores los consigue mediante una consulta a la tabla “servidores”.

Este servicio devolverá a la aplicación hasta un máximo de tres servidores. Al igual que en el caso anterior, se iniciará la conexión con el primero de los servidores, y en caso de que este falle probará con el siguiente servidor de la lista.

En el caso de no haber ningún servidor activo en el campus seleccionado por el usuario, la aplicación mostrará un mensaje de error al estudiante, haciéndole saber que no se puede conectar con el campus seleccionado.

3.2.4. Servicio podcast

Mediante el servicio de *podcast* se pretende que los alumnos de la UPC puedan suscribirse a un *podcast* y descargarse a sus dispositivos móviles el material publicado por la universidad. El uso de *podcast* permite que el material docente esté clasificado en diferentes canales según su temática. Se podrán encontrar *podcast* de diferentes asignaturas, como por ejemplo un *podcast* de “Xarxes i Serveis” u otro de la asignatura “Sistemes Operatius”. De esta forma el estudiante sólo se descargará los contenidos en los que esté interesado.

Ya que se quiere disponer de los mismos *podcast* en todos los campus, ha sido necesaria la replicación de los servidores. Todos los servidores de todos los campus contendrán exactamente los mismos archivos de *podcast*. Si un canal de *podcast* es modificado, se tendrán que realizar esas mismas modificaciones en todos los servidores de los otros campus.

Por lo tanto, cada servidor de cada campus tendrá los mismos archivos, pero los *RSS* de los canales estarán alojados en un único servidor conocido. Están centralizados y no replicados en todos los servidores de los campus. Esto facilita la gestión de los *RSS*. Como el archivo *RSS* tiene un tamaño muy reducido, la distancia entre el servidor y el dispositivo móvil no afecta al tiempo de descarga de dicho archivo.

El dispositivo cliente siempre se deberá conectar al servidor conocido cuando quiera descargarse los *RSS*. Supongamos que este servidor conocido está en Barcelona y el alumno en Castelldefels. Aunque su campus más cercano fuera Castelldefels, no se descarga el *RSS* desde allí.

3.2.4.1. Suscripción a un canal

El alumno podrá suscribirse a los *podcast* ofrecidos por la universidad. Las descargas de cada canal se harán desde el servidor del campus más próximo, en caso de usar el *GPS*.

Para realizar esta suscripción será necesario el uso de un servicio Web llamado *ServicioListaPodcast*. Este servicio es el encargado de conectar con la base de datos para conseguir una lista de los *podcast* disponibles e informar al usuario acerca de ellos. La llamada a este servicio se realizará al iniciar la aplicación.

Este servicio realiza una consulta sobre la tabla “*podcast*”, que es una tabla alojada en la BD que contiene información sobre los canales *podcast* disponibles en el sistema. Esta tabla almacena el nombre de canal, el tipo de archivos y el enlace al *RSS* del canal.

Una vez el cliente conoce los *podcast* del sistema, puede suscribirse a alguno de ellos. Cuando la aplicación se suscribe a un canal, se descarga el *RSS* del canal al cual se quiere suscribir y lo guarda en el directorio adecuado.

En esta aplicación ha sido necesario el diseño de unos directorios de carpetas para almacenar tanto los *RSS*, como los archivos descargados y los de configuración. En la figura 3.13 se pueden observar todas las carpetas que se crearán en el dispositivo cliente.

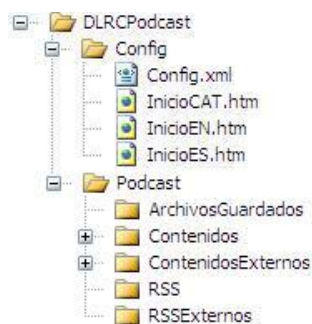


Fig. 3.13 Directorios de carpetas

Hay que mencionar que si la ruta donde se ha instalado la aplicación dispone algún espacio en blanco, se creará en el directorio raíz del dispositivo móvil una carpeta “Podcast” con las carpetas de contenidos y la carpeta “ArchivosGuardados”. Esto es necesario puesto que hay conflictos para reproducir archivos de audio cuya ruta tenga espacios en blanco.

Todos los *RSS* descargados del sistema se guardarán en la carpeta “RSS”. Por lo tanto, en esta carpeta se almacenarán una serie de documentos *XML*. Estos archivos se llamarán igual que el título del canal al cual correspondan.

Cuando se ha descargado el *RSS*, la aplicación lo guarda y lo interpreta, extrayendo la información que contiene. A continuación crea una carpeta dentro de “Contenidos”. Esta carpeta tendrá el mismo nombre que el título del canal al cual se ha suscrito, y contendrá los archivos que se descarguen de este canal.

Por lo tanto, la carpeta “Contenidos” contendrá tantas carpetas como número de canales a los que el usuario esté suscrito.

En la figura 3.14 se muestra como quedarían los directorios después de suscribirse al canal “Xarxes i Serveis” y haber descargado un elemento de dicho canal, “Enunciat_problemes.pdf”.

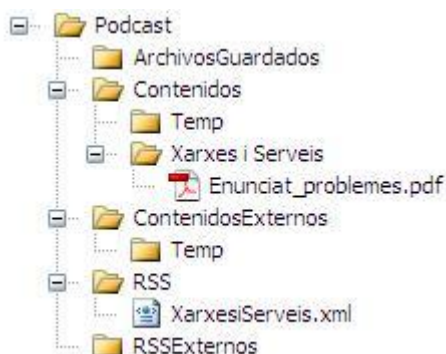


Fig. 3.14 Directorios de carpetas

También es posible la suscripción a cualquier otro *podcast* externo a este sistema, es decir, cualquier *podcast* publicado en Internet. El único requisito necesario es que el archivo *RSS* cumpla con la especificación del *RSS 2.0*. En este caso las descargas se realizarán como cualquier otro cliente *podcast*, es decir, la descarga se realizará directamente del servidor indicado en el *RSS*. A diferencia del caso anterior, los *RSS* se guardarán en la carpeta “RSSExternos” y los archivos descargados en “ContenidosExternos”.

3.2.4.2. *Darse de baja de un canal*

La aplicación permite al estudiante darse de baja de un canal al cual se ha suscrito previamente.

En este caso se eliminará el canal seleccionado, es decir, el archivo *RSS* correspondiente a ese canal será eliminado. Si el usuario decide borrar los archivos asociados al canal, se eliminará, en el directorio “Contenidos”, la carpeta correspondiente a ese canal y todos los archivos que contenga.

En caso de que decida conservar los archivos descargados previamente, sólo se eliminará el *RSS* correspondiente, y la carpeta en la que están los contenidos se moverá a la carpeta “ArchivosGuardados”.

3.2.4.3. *Actualización de un canal*

El usuario también podrá actualizar un canal al que esté suscrito. Para ello la aplicación comprueba si se han publicado elementos nuevos en el canal seleccionado.

Esta comprobación se realiza de la siguiente manera: primero la aplicación se descarga de nuevo el *RSS* del canal y compara sus elementos con los elementos del *RSS* que tiene guardado.

En caso de que encuentre algún elemento nuevo, la aplicación podrá descargarlo.

3.2.4.4. *Gestión de archivos*

La aplicación permite al estudiante gestionar los archivos descargados de los *podcast*. Las opciones que el usuario puede realizar sobre los archivos son: listarlos, reproducirlos y borrarlos.

La aplicación es capaz de listar todos los contenidos descargados de un canal seleccionado previamente. También es capaz de filtrar estos elementos en función del tipo de archivo. Para ello analiza su extensión, de esta forma conoce si son archivos de vídeo, de audio o de otro formato.

La aplicación dispone de un reproductor integrado que permite abrir archivos de audio. Este reproductor, de libre distribución, se denomina *GSPLAYER*.

Cuando la aplicación va a reproducir un archivo, ejecuta el *GSPLAYER* y lo incrusta dentro la aplicación.

Este reproductor no puede abrir archivos que contengan caracteres en blanco en su nombre. Como los archivos de los *podcast* pueden contener estos espacios, la aplicación realiza una comprobación del nombre del archivo antes de reproducirlo. Si el nombre del archivo contiene algún espacio en blanco, realizará una copia del archivo suprimiendo los espacios en blanco contenidos

en su nombre. Esta copia se guardará en una carpeta temporal. Al final de la reproducción, tanto la carpeta temporal como la copia del archivo son eliminadas.

Si el usuario desea reproducir un archivo que no sea de audio, será su responsabilidad disponer de un *software* para este fin. En este caso, el archivo se reproducirá con la aplicación que el usuario tenga seleccionada por defecto.

No se han podido integrar otros reproductores aparte del de audio debido a que surgían muchos problemas a la hora de incrustar estos programas en la aplicación.

3.2.4.5. Descarga de archivos

La aplicación sólo permite realizar descargas de los elementos que no estén descargados pertenecientes a un canal al cual el usuario está suscrito.

Para realizar esta acción, la aplicación compara los archivos de las carpetas de contenidos con los *ítems* de los archivos *RSS*. Si encuentra más *ítems* que archivos, listará los archivos que no se encuentran descargados.

Una vez listados los archivos no descargados, se podrá seleccionar uno de ellos para descargarlo. Para que la aplicación realice una descarga, necesitará conocer donde está alojado el archivo que desea descargar. Mediante el *RSS* del canal, la aplicación conoce la ruta donde están publicados los archivos.

Como no se desea que se descargue el archivo del servidor que indica el *RSS*, sino que se pretende que se descargue el archivo del servidor del campus más cercano, la aplicación modificará la ruta extraída del *RSS* de tal forma que estos archivos se descargarán del servidor al cual esté conectado el dispositivo.

En caso de un canal externo, los archivos se descargarán directamente desde la dirección indicada en su *RSS*, tal como lo hace cualquier otro cliente *podcast*.

Cuando se realiza la descarga de los elementos, éstos se guardarán en la carpeta del canal correspondiente en "Contenidos" o "ContenidosExternos", y tendrán el mismo nombre que el título del elemento del archivo *RSS*.

La aplicación permite realizar más de una descarga al mismo tiempo. Estas descargas se podrán pausar, reanudar o cancelar.

Además la descarga de archivos es robusta. Si durante la descarga de un archivo el servidor se cae y se interrumpe la descarga, la aplicación, después de un tiempo de espera, intentará descargarse ese mismo archivo de los servidores de reserva. En caso de que los servidores de reserva también fallen, se notifica al usuario que no se ha podido realizar la descarga.

CAPÍTULO 4. INTERFAZ Y PRUEBAS

4.1. Interfaz gráfica del cliente

En esta sección se mostrarán las diferentes interfaces con las que el usuario podrá interactuar.

Además, en el anexo 2 están explicados todos los mensajes que puede mostrar la aplicación cliente.

4.1.1. Formulario inicio

Al iniciar la aplicación, el programa cargará el siguiente formulario.



Fig. 4.1 Formulario inicio

En este formulario se puede ver un texto de bienvenida, seguido de dos etiquetas. Estas etiquetas variarán en función de si el uso del *GPS* está activado y del estado en que se encuentre la aplicación.

Si el *GPS* está activado, durante la detección y recepción de la señal, en las etiquetas de estado aparecerá el texto siguiente.

Espere unos minutos mientras se configura el GPS.

Estado: Iniciando...

Fig. 4.2 Etiquetas de estado al iniciar *GPS*

Cuando encuentre señal *GPS*, se intentará conectar a un servidor del campus más próximo. Si la conexión se realiza correctamente, las etiquetas de estado mostrarán el campus al cual el alumno está conectado. Además se mostrará la posición actual mediante las etiquetas de “Latitud” y “Longitud”.

En cambio, si por algún motivo no se ha podido establecer la conexión con el servidor, las etiquetas de estado cambiarán su texto.

Conectando con el servidor mediante GPS...

Conectando con el servidor mediante GPS...

Estado: No Conectado.

Estado: Conectado: BARCELONA

Fig. 4.3 Etiquetas de estado usando *GPS*.

En caso de que el *GPS* esté desactivado y se utilice la opción de conectar directamente con un servidor, las etiquetas de estado que se mostrarán son diferentes al caso anterior. Si se ha podido conectar correctamente, o bien si ocurre algún error, la etiquetas de estado se mostrarán como en la figura 4.4.

Conectando directamente con el servidor...

Conectando directamente con el servidor...

Estado: Conectado: BARCELONA

Estado: No Conectado.

Fig. 4.4 Etiquetas de estado sin utilizar *GPS*

Desde cualquier formulario tendremos acceso a un menú principal, que permitirá al usuario acceder a las diferentes secciones de la aplicación. Para ello, simplemente tendrá que presionar sobre “Menú” y aparecerá un menú desplegable como se puede ver en la figura 4.5.



Fig. 4.5 Menú principal de la aplicación

4.1.2. Formulario navegador

Al seleccionar desde el menú la opción de “Navegador” se accederá a dicho formulario. Como se puede observar en la figura 4.6, éste formulario tiene incrustado un navegador Web, aunque con algunas funciones deshabilitadas, como la barra de navegación y la barra de menú del navegador Web. En este navegador aparecerá una página por defecto indicando que no se está conectado a ningún servidor. Las páginas por defecto están situadas en la carpeta “Config” del dispositivo móvil. Cuando se conecte con algún campus, se cargará la página de inicio del servidor correspondiente.

En la figura 4.6 se puede observar el formulario “Navegador” con una página de inicio de un servidor de un campus.



Fig. 4.6 Página Web mostrada en el navegador

4.1.3. Formulario podcast

El formulario “Podcast” está formado por cuatro pestañas: “Canales”, “Suscripción”, “Archivos” y “Descargas”.

Por defecto, la primera vez que se acceda a este formulario desde el menú, se abrirá la pestaña “Canales”.

4.1.3.1. Pestaña canales

Esta pestaña, figura 4.7, muestra al usuario una lista con todos los canales a los que está suscrito. Si se selecciona un canal, automáticamente se mostrarán todos los elementos que contiene ese canal, tanto los que no están descargados como los que lo están. Estos elementos son los que vienen indicados en el archivo RSS del canal.



Fig. 4.7 Pestaña “Canales” del formulario “Podcast”

Esta pestaña también dispone de dos botones en su parte inferior. El primero de ellos, “Dar de baja”, cancela la suscripción del canal seleccionado. Si no hay ningún canal seleccionado y se pulsa sobre este botón, aparecerá un mensaje de error. Si hay un canal seleccionado, aparecerá un mensaje de advertencia que preguntará al usuario si desea borrar todos los archivos descargados asociados a ese canal.

El otro botón, “Actualizar”, al igual que con el botón anterior, requiere que un canal esté seleccionado. Si no hay ninguno seleccionado, aparecerá un mensaje de error. Al pulsar sobre este botón, la aplicación comprobará si se han añadido nuevos elementos a este canal, y si es así, preguntará al usuario si desea descargarlos o no. En caso afirmativo se añadirá automáticamente esta descarga a la pestaña “Descargas”.

4.1.3.2. Pestaña Suscripción

La pestaña suscripción se puede observar en la figura 4.8. En esta pestaña se muestra al usuario una lista con los diferentes canales de *podcast* que ha publicado la universidad.



Fig. 4.8 Pestaña “Suscripción” del formulario “Podcast”

Debajo del recuadro donde aparecen los diferentes canales, se puede encontrar el botón “Suscribirse”. Al pulsarlo pueden ocurrir diferentes sucesos; si no hay ningún canal seleccionado, se mostrará un mensaje de error. En cambio, si hay un canal seleccionado, aparecerá un mensaje de información que indicará al usuario si la suscripción se ha realizado correctamente. Si el estudiante ya estaba suscrito a ese canal, aparecerá un mensaje de error.

Si el usuario desea suscribirse a un *podcast* que esté publicado en Internet, lo podrá hacer presionando sobre el botón “Suscripción Externa”. Al pulsar sobre este botón, se mostrará al usuario una pantalla donde podrá introducir la *URL* del archivo *RSS* del canal al que se quiere suscribir, ver figura 4.9. Una vez introducida la *URL*, si se presiona sobre “Aceptar”, se volverá a mostrar la pestaña “Suscripción” y saldrá un aviso para informar al usuario si el proceso ha finalizado con éxito o no. Si se presiona sobre el botón “Cancelar” se cancelará la suscripción que está llevando a cabo.



Fig. 4.9 Caja de texto para suscripción externa

4.1.3.3. Pestaña archivos

Esta pestaña permite al usuario gestionar los archivos descargados de cada uno de los canales de los *podcast*, y tiene el siguiente aspecto.



Fig. 4.10 Pestaña “Archivos” del formulario “Podcast”

En esta pestaña el usuario podrá seleccionar un canal de los que aparecen en la lista desplegable. Una vez seleccionado un canal, si se pulsa sobre el botón “?” se mostrará un mensaje con la descripción de ese canal. Lo mismo sucede si se presiona el botón “?” de “Archivos disponibles”, pero en este caso aparecerá información relativa al archivo.

Si se presiona el botón “Aceptar”, en la lista inferior se mostrarán los archivos descargados del canal seleccionado, siempre que haya alguna opción de búsqueda seleccionada.

Las opciones de búsqueda se activan seleccionando los recuadros correspondientes. Si el usuario desea que sólo se listen archivos de audio, deberá seleccionar la opción “Audio”. Lo mismo deberá hacer para archivos de “Vídeo”. La opción “Otros” contempla todos los archivos que no sean ni de audio, ni de vídeo, como por ejemplo, archivos *PDF*. Si se marca la opción de “Todos”, se deseleccionaran automáticamente los criterios de búsqueda anteriores si éstos estaban activados.

La aplicación permite tener seleccionadas al mismo tiempo “Audio” y “Vídeo”, “Audio” y “Otros” o “Vídeo” y “Otros”. Si se seleccionan al mismo tiempo “Audio”, “Video” y “Otros”, automáticamente se deseleccionaran y se seleccionará la opción de “Todos”.

Una vez seleccionado un archivo de un canal, el usuario podrá abrirlo o borrarlo mediante los dos botones situados en la parte inferior del formulario. Si se presiona alguno de estos dos botones sin tener seleccionado un archivo, se mostrará un mensaje de error.

Al pulsar sobre “Reproducir”, si el archivo es de audio, se abrirá el formulario “Reproductor” con el reproductor integrado y se reproducirá el archivo. Si el archivo no es de audio, se intentará abrir con la aplicación que el dispositivo tenga asociada a la extensión de dicho archivo.

También se podrá borrar el archivo seleccionado pulsando sobre el botón “Borrar”. La aplicación mostrará un mensaje de advertencia para asegurarse que el usuario realmente quiere borrar el archivo.

4.1.3.4. *Pestaña descargas*

Ésta es la cuarta pestaña del formulario “Podcast”. Mediante esta pestaña el usuario podrá gestionar las descargas que realice.



Fig. 4.11 Pestaña “Descargas” del formulario “Podcast”

En esta pestaña se puede encontrar una lista desplegable donde el usuario deberá seleccionar un canal. Una vez seleccionado un canal, el usuario podrá pulsar sobre el botón “Aceptar” para que se muestren los archivos de ese canal que no están descargados. Para descargar uno de estos archivos, el usuario lo deberá seleccionar y posteriormente deberá pulsar sobre el botón “Descargar”.

A medida que progresa la descarga, la barra se irá llenando, y el tanto por ciento de la descarga irá aumentando hasta llegar al 100%. Una vez llega al 100%, este valor será sustituido por el texto “Completada”.

Mientras una descarga está en curso, el usuario la podrá pausar o cancelar. Para realizar una de estas acciones el usuario deberá seleccionar la descarga correspondiente, y después presionar sobre el botón “Pausar” o “Parar”. Para seleccionar todas las descargas que aparecen en la pantalla, el usuario podrá presionar sobre el botón “Selec. todo” o si quiere deseleccionar las descargas, podrá pulsar sobre “Desel. todo”.

Cuando se pausa una descarga, el tanto por ciento descargado cambiará por “Pausada.”, en cambio si se cancela, cambiará por “Cancelada”. Las descargas pausadas se podrán reanudar si éstas están seleccionadas y se pulsa sobre el botón “Reanudar”.

Las descargas que estén completadas o canceladas podrán quitarse de la pantalla si se presiona sobre el botón “Limpiar”.

4.1.4. Formulario reproductor

El formulario “Reproductor” contiene un reproductor *GSPLAYER* integrado, incrustado en el propio formulario, que permite abrir archivos de audio. Este reproductor no estará en funcionamiento desde que se inicia la aplicación, sino que se activará cuando el usuario decida reproducir un archivo de audio.

Cuando el reproductor no está activado, el formulario “Reproductor” tendrá una apariencia como en la figura 4.12.



Fig. 4.12 Formulario “Reproductor”

Cuando se abre un archivo de audio, este formulario se mostrará en primer plano y reproducirá dicho archivo. El usuario podrá pausar la reproducción del archivo, avanzar o retroceder y cambiar el volumen mediante controles del propio reproductor.

4.1.5. Configuración

El formulario “Configuración” tiene una apariencia como el de la figura 4.13.



Fig. 4.13 Formulario “Configuración”

Este formulario dispone de una etiqueta de título que lo identifica y de una barra de desplazamiento. A continuación muestra las diferentes opciones que el usuario podrá cambiar dependiendo de sus preferencias.

4.1.5.1. *Tiempo actualización*

Esta opción permite modificar el periodo de tiempo, en minutos, que pasará hasta que se realice una nueva llamada al servicio *ServicioServidorCercano*. Este servicio es el que le comunica a la aplicación que campus se encuentra más próximo al usuario y a qué servidor se conectará.

Cuanto mayor sea este tiempo, menos llamadas se realizarán al servicio, por lo que habrá menos comprobaciones sobre cuál será el servidor más próximo al dispositivo.

Para modificar este tiempo, bastará con desplazar hasta el índice deseado la *trackbar*. Al final de la barra hay una etiqueta que muestra el valor seleccionado, como se puede observar en la figura 4.14.

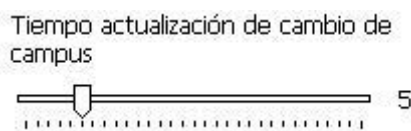


Fig. 4.14 Barra de tiempo de actualización

4.1.5.2. *Actualizaciones automáticas*

El siguiente parámetro que este formulario permite controlar al usuario es activar o no las actualizaciones automáticas. Para activar esta opción habrá que presionar en el recuadro correspondiente.



Fig. 4.15 Activación de actualizaciones automáticas

Si esta opción está activada al iniciar el programa, será la propia aplicación la que se encarga de consultar si se han colgado nuevos contenidos en los canales que el usuario está suscrito. Si es así, se descargarán automáticamente los elementos nuevos, añadiéndolos al panel de descargas de la pestaña "Descargas" del formulario "Podcast".

4.1.5.3. *Borrar archivos al darse de baja*

Cuando el usuario se da de baja de un canal, aparecerá una caja de texto que le preguntará si desea borrar los archivos asociados a ese canal, pero si esta opción está activada, no aparecerá dicha caja de texto, sino que automáticamente borrará los contenidos que el usuario se haya descargado de ese canal.

4.1.5.4. *Puerto GPS automático*

Cuando la aplicación realiza un escaneo de los puertos por los cuales llega la señal de *GPS*, podría darse el caso que llegara señal por más de un puerto si el dispositivo móvil dispone de más de un receptor *GPS*. En este caso, si esta opción está desactivada, se le preguntará al usuario que puerto desea utilizar para leer la señal. En cambio, si esta opción está activada, la aplicación utilizará el primer puerto que encuentre por el cual llegue la señal de *GPS*.

4.1.5.5. *Orientación de la pantalla*

Mediante esta opción el usuario podrá cambiar la orientación de la pantalla, es decir, podrá decidir si la posición de la pantalla será vertical, o si será apaisada. Si la orientación cambia, el tamaño de los controles se adaptará automáticamente a la nueva posición que tome la pantalla tal y como se puede observar en la figura 4.16.



Fig. 4.16 Pantalla apaisada

Hay que mencionar también que el cambio de la orientación no afectará a la posición que tenía el dispositivo antes de iniciar la aplicación. Por ejemplo, si antes de iniciar la aplicación la posición de la pantalla es vertical, se inicia la aplicación, se cambia mediante esta opción la orientación a horizontal y se cierra la aplicación, el dispositivo volverá a recuperar la posición inicial en que estaba antes de iniciar la aplicación, en este ejemplo, en vertical.

4.1.5.6. Idioma

Mediante esta opción el usuario podrá seleccionar el idioma en el cual quiere que se muestren todos los controles (etiquetas, textos de los botones...) y todos los mensajes que mostrará la aplicación tanto mensajes de error como advertencia e información.

Como muestra la figura 4.17, los idiomas disponibles son el castellano, el catalán y el inglés.

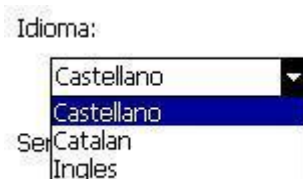


Fig. 4.17 Desplegable de selección de idioma

4.1.5.7. Servidor por defecto

Esta opción permite al usuario decidir a qué campus se conectará la aplicación, seleccionando el valor del desplegable y presionando sobre el botón "OK". Por ejemplo, si el usuario selecciona "Barcelona", la aplicación se conectará directamente a algún servidor del campus de Barcelona. También es posible seleccionar la opción "Auto". Si el desplegable está seleccionado con este valor, la aplicación utilizará el *GPS* para determinar el campus más cercano al dispositivo para conectarse a él.



Fig. 4.18 Desplegable de selección de campus

En cualquier momento el usuario podrá cambiar esta opción. Por ejemplo, si estaba seleccionada la opción de "Barcelona" y el usuario cambia a "Vilanova", aparecerá un mensaje de advertencia indicando al usuario que el campus ha cambiado y si desea conectarse al nuevo campus seleccionado.

4.1.5.8. Guardar Cambios

Al presionar el botón “Guardar Cambios”, se guardará en el archivo *config.xml* los valores que tienen las diferentes opciones. La próxima vez que se inicie el programa se cargarán las opciones con los valores guardados.

4.1.6. Salir

Al seleccionar esta opción del menú, la aplicación se finalizará, cerrando todos los formularios y cancelando todas las descargas activas de archivos que se estuvieran realizando en ese momento, así como parando el reproductor y cerrando los diferentes hilos de ejecución.

4.2. Pruebas realizadas

Para las pruebas descritas a continuación se han utilizado los siguientes elementos:

- Windows XP Professional SP2
- Visual Studio 2005 y .NET Framework 2.0 sp1
- Emulador Pocket PC 2003 y Active Sync 4.5
- Internet Information Server 5.1
- MySQL Server 6.0, MySQL Connector Net 5.1.6 y MySQL Tools
- Internet Explorer 6.0, 7.0.5730.13
- Mozilla Firefox, versiones 2.0.0.14 y 3.0
- Google Earth 4.2.0198.2451 (Beta)
- GPS.NET 2.0
- PDA Acer C530 (Con S.O. Windows CE 5.0)
- Aplicación cliente

4.2.1. Consultas SQL través de un servicio Web

El objetivo de esta prueba es comprobar el correcto funcionamiento de los servicios Web utilizados por los administradores.

Desde un ordenador cualquiera, actuamos como administradores y accedemos al servidor conocido (con dirección *IP 147.83.119.126*) donde están implementados los servicios Web y está alojada la BD. Abrimos un navegador Web e introducimos la siguiente *URL*:

<http://147.83.119.126/ServicioAdministrador/Service.asmx>

Comprobamos que el navegador muestra los *Web Methods* correspondientes.

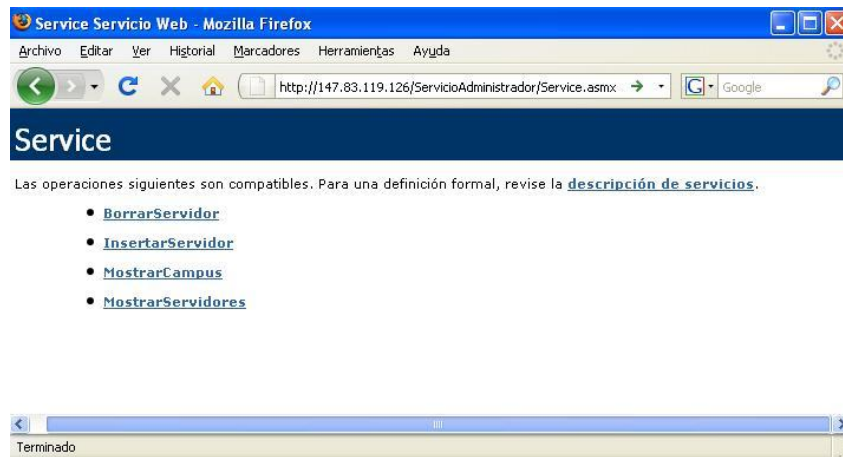


Figura 4.19 Métodos Web del servicio de administración

El servicio Web estará formado por cuatro *Web Methods*. Cada *Web Method* corresponde a una acción que se puede realizar mediante el servicio Web. Cuando se accede directamente a un servicio Web, lista todos los métodos Web que ofrece.

Para que el servicio Web pueda realizar una conexión con la base de datos de *MySQL*, es necesario tener instalado algún elemento intermedio que permita la comunicación entre el servicio y la BD. Este elemento es el *MySQL .NET Connector*, que debe estar instalado en la máquina donde se ejecuta el servicio Web.

Una vez el servicio puede comunicarse con la base de datos, ya puede ejecutar los cuatro métodos, que realizarán las consultas SQL.

Si pulsamos sobre “Insertar Servidor” nos aparece una pantalla en la cual introducimos los parámetros necesarios, tal como muestra la figura 4.20.

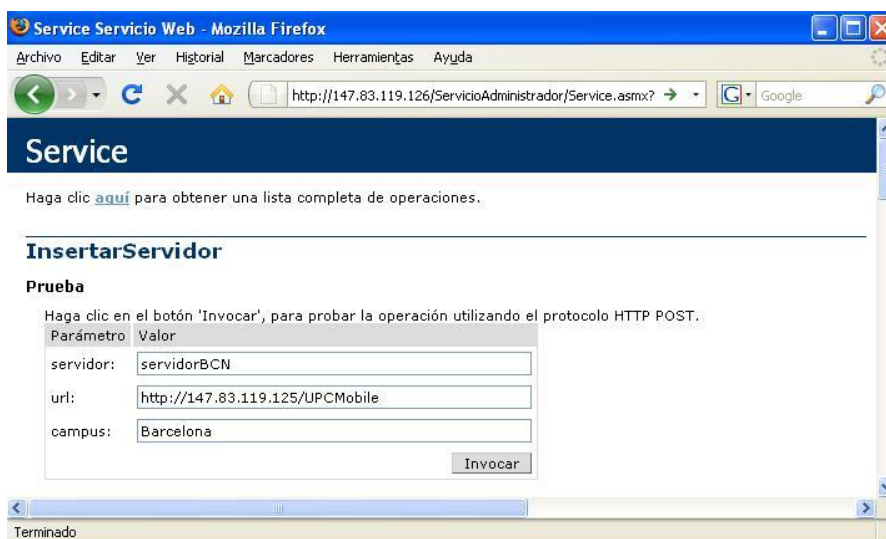


Figura 4.20 Invocar al método “InsertarServidor”

Una vez introducidos los parámetros correspondientes invocamos al servicio. Aparece un mensaje informando que todo ha ido correctamente.

Desde el ordenador “147.83.119.126”, abrimos la base de datos y observamos que ha introducido correctamente los datos, como muestra la figura 4.21.

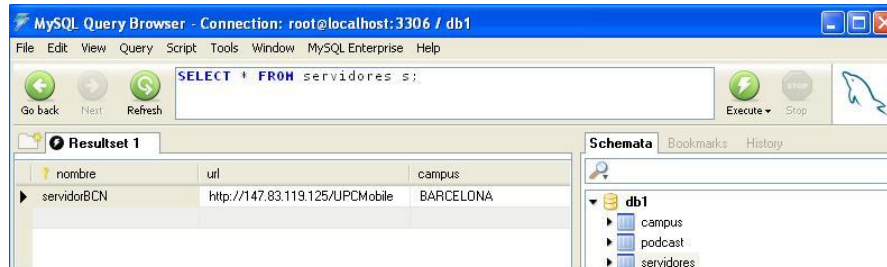


Figura 4.21 Comprobación de la BD

4.2.2. Cálculo de distancia entre cliente y campus

Dadas las coordenadas de un cliente y diferentes campus, se quiere calcular la distancia entre cada uno de ellos mediante un servicio Web.

Conociendo estas coordenadas, es posible obtener el campus más cercano al dispositivo móvil cliente, ya que se puede calcular la distancia del dispositivo con cada uno de los campus.

Para realizar este cálculo se ha simplificado el problema suponiendo que la Tierra es una esfera perfecta. Con esta suposición se introduce un pequeño error, pero a efectos prácticos no afecta al resultado final.

Se ha diseñado un servicio Web que recibe como parámetros las coordenadas del cliente. Además obtiene las coordenadas de los campus a través de una consulta a la base de datos.

Este servicio es el encargado de realizar el cálculo de distancias y escoger el campus con la distancia mínima respecto al cliente. Este cálculo lo realiza mediante la siguiente fórmula, obtenida de <http://mathforum.org/library/drmath/view/51711.html>.

$$\arccos(\cos(\text{latitud Cliente}) * \cos(\text{latitud Servidor}) * \cos(\text{longitud Cliente}) * \cos(\text{longitud Servidor}) + \cos(\text{latitud Cliente}) * \sin(\text{longitud Cliente}) * \cos(\text{latitud Servidor}) * \sin(\text{longitud Servidor}) + \sin(\text{latitud Cliente}) * \sin(\text{latitud Servidor})) * \text{radio de la Tierra.} \quad (4.1)$$

El radio de la Tierra se ha aproximado a 6378 Km.

Las funciones matemáticas del seno, el coseno y el arcoseno están implementadas en la librería *Math*, por lo tanto para usar estos métodos basta con importar esta librería.

Para comprobar su correcto funcionamiento, se ha procedido al cálculo de distancia de un cliente situado en el campus de Castelldefels respecto a este mismo campus y el campus de Barcelona.

La distancia obtenida del cliente con el campus de Barcelona es de 16,3779011758607 Km. Si se compara este resultado con el obtenido mediante la herramienta de medir distancias de *Google Earth*, se puede observar que son muy similares ya que el de *Google Earth* es 16,39 Km. La diferencia entre ambos resultados es mínima, además este tipo de imprecisiones no afectan al correcto funcionamiento del sistema puesto que no requiere tanta precisión.

La distancia obtenida para el campus de Castelldefels es de 0,000095039606094360352 Km. El resultado esperado sería de 0 Km, pero como se ha comentado anteriormente se introduce un pequeño error.

4.2.3. Pruebas finales

Estas pruebas se han realizado con el sistema en funcionamiento. Disponemos de un servidor en Castelldefels y otro en Barcelona.

El servidor de Barcelona se encuentra en Campus Nord y se puede acceder a él mediante la siguiente *URL*: *pfc.pc.ac.upc.edu*. Este servidor actuará como el servidor conocido.

En Barcelona está situada la base de datos y el servidor *Web IIS*, que contiene los servicios *Web* y los *RSS*, así como los archivos de los *podcast* y las páginas del portal *Web*. Este servidor está conectado a la red externa para que se pueda acceder a él mediante Internet.

En este servidor de Barcelona, se han publicado tres *RSS*, es decir, el sistema dispondrá de tres canales *podcast* diferentes. Cada uno de ellos ofrece un tipo de contenido diferente: vídeo, audio y *pdf*.

Se ha comprobado su sintaxis mediante el servicio validación de *RSS* que ofrece el *W3C* (Consortio Web). A dicho servicio se accede mediante la siguiente *URL*:

<http://validator.w3.org/feed/>

Todos los *RSS* utilizados para las pruebas son válidos. Cuando un *RSS* es correcto, el validador muestra la imagen de la figura 4.22.



Fig. 4.22 RSS válido

Hay que destacar que aunque en estas pruebas tanto los servicios Web como la base de datos y los RSS están en un mismo ordenador, esto no tiene porque ser así. Podría estar cada uno de estos elementos en ordenadores diferentes. Esta prueba se ha realizado de esta manera ya que únicamente disponíamos de un ordenador en Barcelona.

Por otro lado, en Castelldefels también se ha instalado un servidor Web IIS, con dirección IP: 147.83.119.126, pero en este caso solo contiene los archivos de los *podcast* y las páginas del portal Web. Estos contenidos son los mismos que se pueden encontrar en el servidor de Barcelona, ya que están replicados.

Una vez instalada la aplicación cliente en la *PDA ACER*, se comprueba el correcto funcionamiento del programa. Para ello, se realizan diferentes pruebas.

En el momento de realizar las pruebas siguientes el dispositivo estaba ubicado en Castelldefels.

4.2.3.1. Pruebas interfaz

Una vez instalada la aplicación en la *PDA*, se comprueba que ha creado correctamente todos los directorios y los archivos de configuración.

Además se comprueba el funcionamiento de todos los controles de los formularios de la aplicación.

4.2.3.2. Pruebas funcionamiento del GPS

Desde el formulario “Configuración”, se activa la opción de utilizar el *GPS* para conectarse al campus, es decir, se selecciona “Auto” en “Campus por defecto”.

Tras unos instantes se puede comprobar que en el formulario “Inicio” las etiquetas de estado han cambiado correctamente, mostrando las coordenadas y el campus al cual se ha conectado. Como se puede observar, este campus es el de Castelldefels. Las coordenadas que aparecen también son correctas, ya que se han insertado en *Google Earth* y nos ha mostrado el mapa con la posición donde estaba situada la *PDA* en el momento de las pruebas.

El resultado de esta prueba ha sido satisfactorio, ya que la aplicación ha sido capaz de encontrar el puerto adecuado por el cual llega la señal *GPS*, procesar la información y obtener las coordenadas correctamente, como se puede observar en la figura 4.23. Además la comunicación con el servicio *ServicioServidorCercano* ha sido correcta ya que se ha conectado al campus adecuado.



Fig. 4.23 Formulario “Inicio”

4.2.3.3. Tiempo de descarga

En esta prueba se desea comprobar que se tarda menos en descargar un archivo del campus más próximo al estudiante.

Análisis teórico

Nuestro sistema trabaja sobre Internet, es decir, en una red *TCP/IP* de conmutación de paquetes. Este tipo de redes se caracteriza por dividir la información que se desea transmitir en varios paquetes de longitud máxima conocida.

En este tipo de redes hay tres tipos de retardos posibles: tiempo de propagación, tiempo de transmisión y tiempo de proceso.

- El tiempo de propagación es el tiempo que tarda el paquete transmitido en recorrer la distancia que le separa del siguiente nodo. Este tiempo se puede definir entre con la fórmula siguiente:

$$T_{prop} = \text{distancia} / \text{Velocidad de propagación del medio} \quad (4.2)$$

La velocidad de propagación del medio varía en función del material utilizado. Por ejemplo, en fibra óptica esta velocidad es diferente de si utilizamos un cable *UTP* (par trenzado).

- El tiempo de transmisión es el tiempo que se tarda en enviar la señal. En este caso el tiempo depende de la velocidad de transmisión. Se puede expresar mediante la siguiente fórmula:

$$T_{tx} = \text{datos} / V_{tx} \quad (4.3)$$

Para la transmisión de un mismo archivo, este tiempo es menor cuanto mayor sea la velocidad de transmisión utilizada. Si se utiliza un servidor con una velocidad de transmisión de 100 *Mbps*, el tiempo de transmisión será menor que si utilizamos uno con una velocidad de 10 *Mbps*.

- El tiempo de proceso es el tiempo que tarda un nodo en procesar un paquete. Este tiempo puede variar en función del estado del nodo. Si un nodo está saturado, ya que tiene que procesar muchos paquetes en un corto espacio de tiempo, el tiempo de proceso probablemente sea mayor. En el caso contrario, es decir, que el nodo tenga pocos paquetes que procesar, el tiempo de proceso probablemente sea, a efectos prácticos, nulo.

En la siguiente figura se muestra un ejemplo de los tiempos explicados anteriormente.

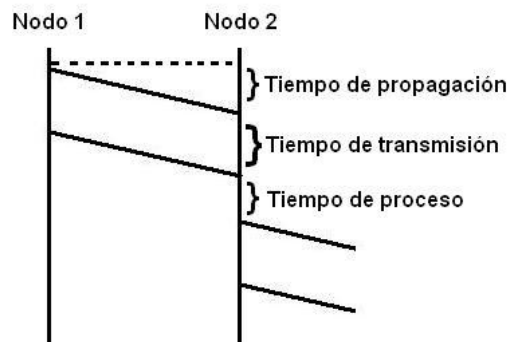


Fig. 4.24 Tiempos

Consideraciones del sistema

Este sistema dispone de varios servidores donde realizar una misma descarga. Puesto que se quiere ofrecer el mismo servicio a todos los clientes, la velocidad de transmisión, así como la velocidad de propagación del medio para estos servidores debería ser similar. Por este motivo, el tiempo de transmisión será similar independientemente del servidor utilizado, tanto si es próximo como si no, y el tiempo de propagación solamente variará en función de la distancia.

Tanto el tiempo de propagación como el tiempo de proceso pueden variar en función del servidor utilizado. Como ya se ha explicado a lo largo de este proyecto, el utilizar un servidor más próximo hace que las descargas se realicen en menor tiempo. Concretamente se pretende que estos tiempos sean menores.

Si la distancia entre cliente y servidor es pequeña, resulta obvio que el tiempo de propagación también debe de ser menor (a una misma velocidad de propagación).

Respecto al tiempo de proceso es más difícil predecirlo. Según el estado de la red este tiempo de proceso variará. Por ello, cada nodo tiene una probabilidad de estar saturado.

Con una descarga de un servidor más próximo se pretende que el número de nodos utilizados sea el menor posible. Con ello se consigue que la probabilidad de encontrar saturación durante la descarga también sea menor. Cuantos más

nodos se tengan que recorrer mayor será la probabilidad de encontrar saturación.

Resultados experimentales

Con tal de evaluar el funcionamiento de las descargas en nuestro sistema se utilizó un único servidor, concretamente el que tenemos instalado en el campus de Castelldefels.

Además fue necesario realizar las descargas desde dos campus de la UPC diferentes. Estas descargas las realizamos mediante la *PDA* desde el campus de Castelldefels y desde campus Nord, utilizando la red inalámbrica de cada uno de ellos. El sistema utilizado para estas pruebas se resume en la siguiente figura.

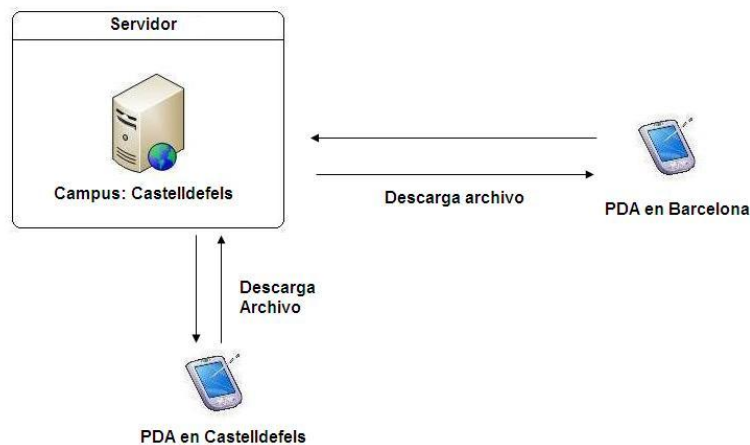


Figura 4.25 Esquema del sistema utilizado para las pruebas de descargas

Se realizaron diversas descargas de archivos de vídeo a través de nuestra aplicación desde los diferentes campus. Para cada archivo se realizó más de una descarga para obtener un tiempo en media del tiempo total de la descarga. El resultado final de estas pruebas lo mostramos en la siguiente tabla.

Nombre	Tamaño	Tiempo de descarga	
		Castelldefels	Barcelona
Igor.mpg	19,3 MB	1 min 3 seg	1 min 7 seg
Indiana Jones.mpg	19,7 MB	1 min 04 seg	1 min 10 seg
Shrek 2.mpg	9,20 MB	29 seg	33 seg
Star Wars.mpg	20,0 MB	1 min 04 seg	1 min 08 seg

Tabla 4.1 Tiempos de descarga

Se puede observar que para las descargas realizadas desde el campus de Barcelona al servidor de Castelldefels obtenemos un tiempo mayor. La diferencia en este caso oscila entre los 4 y 6 segundos.

Esta diferencia es debida a la mayor distancia entre cliente y servidor, y también a los posibles nodos saturados que haya podido encontrar el cliente al realizar la descarga.

Cabe destacar que todas estas pruebas se han realizado utilizando la red de la UPC sin salir a la red exterior. La red de la UPC dispone de un gran ancho de banda por lo que por este motivo los tiempos de todas las descargas son tan pequeños.

Durante el transcurso de estas pruebas nos surgió un pequeño problema. El mal dimensionamiento del *buffer* en la aplicación cliente limitaba el tiempo de descarga independientemente si esta se hacia desde el campus de Barcelona o Castelldefels. Una vez nos dimos cuenta del error, y dimensionamos bien el *buffer* de la aplicación pudimos realizar estas pruebas correctamente.

4.2.3.4. Robustez del sistema

Las últimas pruebas realizadas pretenden demostrar la robustez del sistema frente a posibles errores inesperados. Estos errores se pueden producir por diversas causas y en diferentes puntos del sistema. Aun así, la aplicación cliente es capaz de detectarlos y actuar en consecuencia. Cuando sea posible se solucionan estos problemas y cuando no lo sea se informa al usuario del error producido.

Utilización de servidores de reserva

Para esta prueba instalamos un segundo servidor en el campus de Castelldefels. Además de la instalación, añadimos un nuevo registro en la base de datos para informar que este servidor ya está activo.

En esta prueba se pretende que el servidor principal se caiga y que la aplicación cliente utilice automáticamente el servidor de reserva.

Iniciamos una descarga del servidor principal de Castelldefels y, antes que esta descarga finalice, hacemos caer el servidor. Para ello desconectamos su cable de red. En la aplicación cliente se muestra un mensaje de error informando que no se ha podido completar la descarga. Posteriormente se muestra un mensaje preguntando si desea hacer la descarga de otro servidor. Respondemos afirmativamente y podemos observar como la descarga vuelve a realizarse sin ningún problema. En la figura 4.26 se muestra un resumen de este proceso.

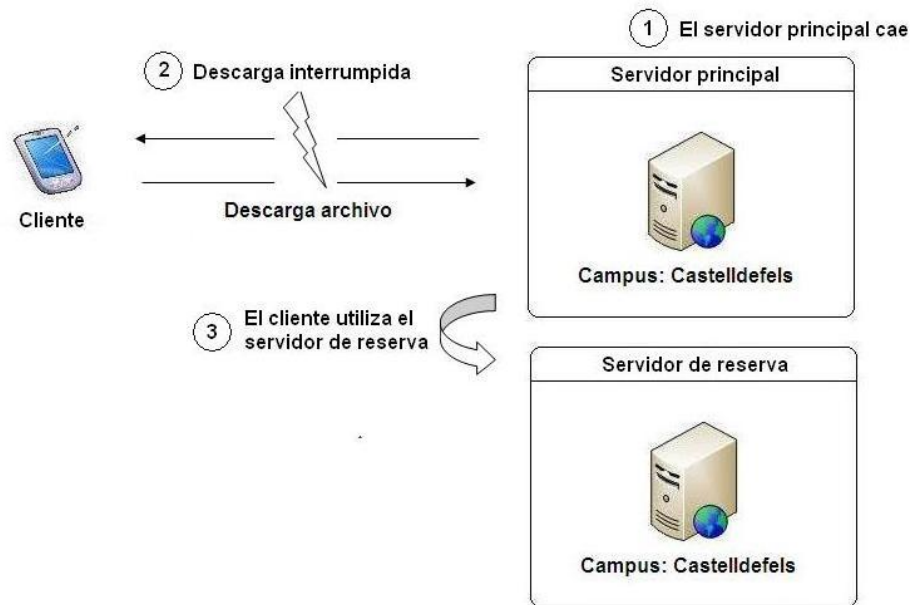


Figura 4.26 Robustez del sistema

Cuando el cliente ejecuta el servicio *Web ServicioServidorDirecto*, o bien el *ServicioServidorCercano*, la respuesta de ambos es una lista de servidores. Esta lista está almacenada en el dispositivo móvil del cliente. Si se produce un error al conectar a uno de los servidores el cliente podrá utilizar cualquiera de los otros contenidos en la lista.

Después de esta prueba podemos afirmar que el cambio de servidor utilizado se realiza con éxito.

Mensajes en la aplicación cliente

Otro aspecto importante dentro de la aplicación cliente ha sido los mensajes mostrados al usuario. Es importante poder informar al cliente de los problemas surgidos a lo largo de la utilización del programa.

En estas pruebas se han querido provocar diferentes incidencias que impidan el correcto funcionamiento del sistema. Con ello queremos comprobar si la aplicación informa al usuario en todos los casos.

Para realizar las pruebas hemos tenido que realizar acciones que sabemos que provocarán algún tipo de error. Algunas de estas acciones han sido: borrar manualmente los archivos de configuración de la aplicación, dejar al cliente sin conexión a Internet, suscribirse a un canal de *podcast* con un *RSS* mal formado, conectar a un campus donde no hay servidores, etc.

Después de realizar estas pruebas el resultado es satisfactorio ya que cada error es detallado al cliente.

Una completa lista y explicación de todos los errores de la aplicación cliente viene incluida en el anexo 3.

CAPÍTULO 5. CONCLUSIONES

Una vez implementado el sistema y realizadas las pruebas pertinentes, se comentan los resultados obtenidos, comparándolos con los objetivos iniciales de este proyecto.

En este capítulo también se comentan algunos ejemplos de usos comerciales y algunas posibles ampliaciones para este proyecto. Finalmente, se exponen las conclusiones personales después de haber realizado este TFC.

5.1. Objetivos conseguidos

El objetivo final de este proyecto era implementar una aplicación *podcast* para dispositivos móviles en *.NET* utilizando servicios Web y *GPS*.

Los objetivos marcados al inicio de este proyecto se han cumplido satisfactoriamente.

Se ha implementado una aplicación para dispositivos móviles capaz de conectarse con el campus más cercano al estudiante. Esta aplicación también es capaz de suscribirse a los *podcast* ofrecidos por la UPC. Por otro lado, se han implementado los servicios Web adecuados que la aplicación pueda obtener la información necesaria de la base de datos. También se ha conseguido obtener un sistema robusto y distribuido, consiguiendo así poder realizar las descargas de manera rápida y fiable.

Durante las pruebas realizadas se ha podido verificar el correcto funcionamiento del sistema, tanto por parte de los servicios Web, como por parte de la aplicación.

Aunque hay que añadir que algunas funcionalidades de la aplicación no han podido ser implementadas. No se han podido incrustar reproductores de archivos de vídeo ni de *PDF*, ya que ocasionaban muchos errores. De todas formas, estos archivos se pueden reproducir mediante otros programas instalados por el usuario. Además no ha sido posible cumplir con la idea inicial de realizar una aplicación para cualquier tipo de dispositivo móvil, ya que al trabajar con la tecnología *.NET*, estábamos limitando los dispositivos móviles a aquellos que dispongan de sistema operativo *Windows CE*.

5.2. Ámbito de uso

Este TFC se adentra en el mundo de la movilidad y podría aplicarse en diferentes ámbitos. Puesto que se ha diseñado una aplicación para detectar y conectarse al servidor más próximo, se podría adaptar el diseño de este TFC para otros sistemas. A continuación se explica algunas posibles adaptaciones de este proyecto.

En determinadas estaciones de tren podría estar ubicado un servidor que ofrezca mediante servicios Web información sobre el estado de los transportes, como por ejemplo si algún tren ha sufrido algún retraso. El usuario podrá conectarse a estaciones diferentes para informarse sobre el estado actual de los transportes.

Otra posibilidad es que en los viajes de trayecto largo de tren, por ejemplo, de Barcelona a Galicia, el pasajero pueda ver una película desde su dispositivo móvil, que se irá descargando de un servidor u otro, dependiendo de por dónde esté pasando el tren en ese momento.

Otra posible salida comercial sería la de diseñar un sistema para ofrecer a los turistas de Barcelona información acerca de la ciudad a través de un *podcast* de audio. Por ejemplo, cuando un turista pasee cerca de la Sagrada Familia, reciba en su dispositivo móvil un mensaje de aviso que muestre “¿Desea escuchar información acerca de la Sagrada Familia?”. En caso afirmativo, se descargará y reproducirá el archivo de audio.

5.3. Ampliaciones y mejoras

Una posible ampliación de este TFC, sería implementar un reproductor propio y añadirlo a la aplicación cliente. De esta forma, se podría reproducir cualquier archivo de los canales *podcast* desde la aplicación.

Otra posible mejora sería ampliar la aplicación de forma que se descargara los archivos de otros dispositivos de la zona, en vez de descargárselos de unos servidores. Para ello sería necesario implementar un módulo de comunicación P2P y agregarlo al diseño actual.

El funcionamiento de este módulo sería el siguiente. Un usuario se suscribiría a un canal de *podcast*. En vez de descargarse los archivos del *podcast* de un servidor, se los descargaría de otros clientes. La decisión de conectarse con un cliente u otro dependería de su posición.

Otra posible mejora sería dotar de seguridad el sistema. Para ello sería necesario cifrar el contenido de todas las comunicaciones realizadas entre el dispositivo cliente y los servicios Web.

5.4. Conclusiones personales

El haber realizado este proyecto nos ha parecido una experiencia muy satisfactoria a nivel personal, ya que nos ha permitido adquirir una serie de conocimientos que nos parecen muy interesantes y útiles.

Este proyecto abarca muchos campos diferentes, entre ellos, la tecnología *.NET*. Ha sido necesario el estudio de esta tecnología, la cual era desconocida para nosotros.

Este estudio nos ha permitido conocer el funcionamiento de los servicios Web, así como la programación para dispositivos móviles. Para ello ha sido necesario el aprendizaje del lenguaje de programación *Visual Basic .NET*, con el cual no habíamos trabajado anteriormente. El aprendizaje de un lenguaje de programación tan importante y extendido como éste nos ha parecido de gran utilidad.

Hemos tenido algunas dificultades a lo largo del desarrollo del proyecto, sobre todo en lo que respecta a la programación de la aplicación cliente.

Al ser un proyecto conjunto, nos ha permitido poner en práctica nuestras capacidades de trabajo en equipo, tal y como se nos ha enseñado a lo largo de esta carrera.

CAPÍTULO 6. BIBLIOGRAFÍA

- [1] Charte, F., *Programación con Visual Basic .NET*, Ed. Anaya Multimedia, 2002.
- [2] Parsons, A. y Randolph, N. , *Professional Visual Studio 2005*, Ed. Wiley Publishing, Indiana, 2006.
- [2] W3Schools Online Web Tutorials (2007, setiembre) *.NET Mobile* [En línea].
Página web, URL <<http://www.w3schools.com/dotnetmobile/default.asp>>
- [3] W3Schools Online Web Tutorials (2007, octubre) *Miscrosoft .NET* [En línea].
Página web, URL <<http://www.w3schools.com/ngws/default.asp>>
- [4] W3Schools Online Web Tutorials (2008, enero) *RSS* [En línea].
Página web, URL <<http://www.w3schools.com/rss/default.asp>>
- [5] MSDN (2007, diciembre) *MSDN Library* [En línea].
Página web, URL <<http://www.msdn.microsoft.com/en-us/library/default.aspx>>
- [6] El Guille (2007, diciembre) *Curso Visual Basic .NET* [En línea].
Página web, URL <<http://www.elguille.info/NET/cursoVB.NET/indice.htm>>



Escola Politècnica Superior
de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA

ANEXOS

TÍTULO DEL TFC: Acceso a aplicaciones desde dispositivos móviles mediante servicios Web en función del posicionamiento geográfico.

TITULACIÓN: Ingeniería Técnica de Telecomunicación, especialidad Telemática

AUTORES: David Andrés López Nuevo, Rubén Cuadrado Sánchez

DIRECTOR: Dolors Royo Vallés

FECHA: 24 de Julio de 2007/08

ANEXO 1. IMPLEMENTACIÓN DEL DISEÑO

En este apartado se muestra un diagrama de clases de las diferentes partes del proyecto. Estos diagramas muestran el nombre de cada clase, así como los atributos y los métodos que pertenecen a cada una de éstas. Estos diagramas los genera el *Visual Studio* automáticamente.

También se explicará una breve descripción de la funcionalidad de cada clase.

Diagrama de clases de servicios Web

A continuación se muestra el diagrama de clases de un servicio Web, concretamente del servicio “ServicioAdministrador”. Aunque en este proyecto se utiliza más de un servicio, en esta sección solamente se explicará uno de ellos ya que las clases utilizadas en todos ellos son similares.

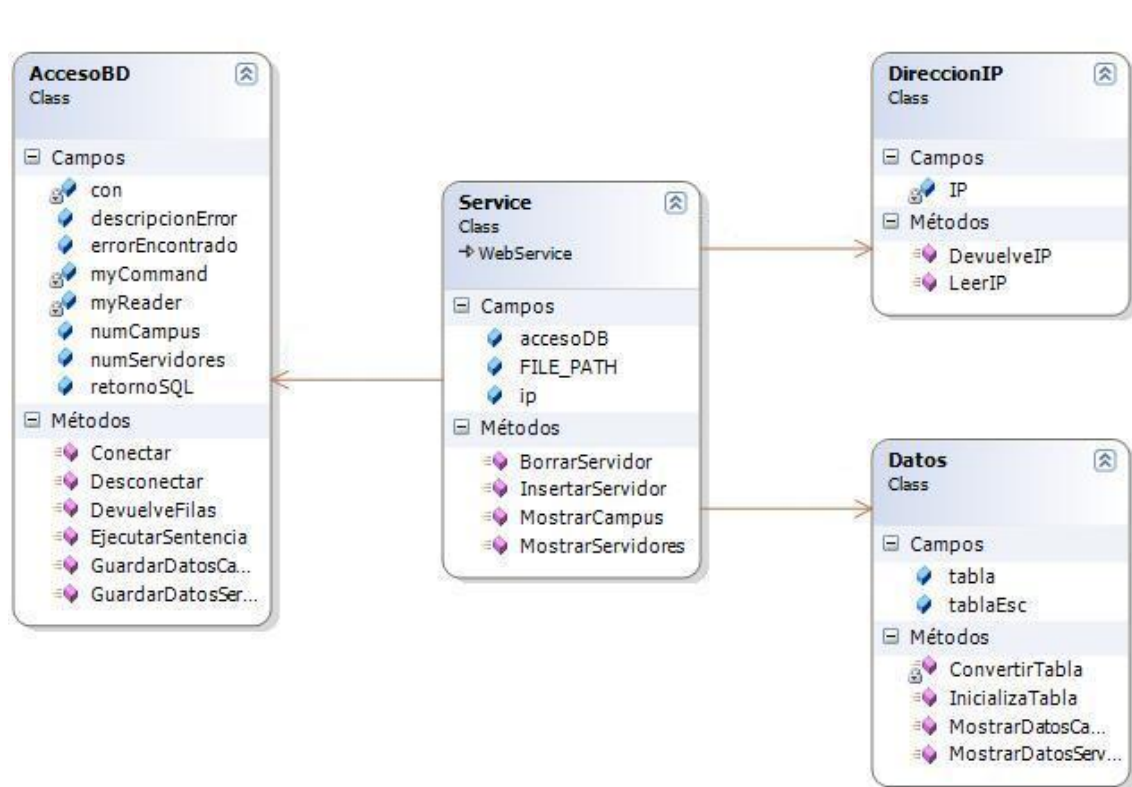


Fig. A1.1 Diagrama de clases del *ServicioAdministrador*

Estas clases son las que pertenecen a la parte del proyecto en la cual los administradores acceden a un servicio Web para hacer consultas en la BD.

- **Clase Service:** Esta clase representa un objeto de un servicio Web. Dispone de cuatro métodos, que son los diferentes servicios que ofrece. Estos métodos son: insertar y borrar un servidor en la BD, mostrar los

datos de los servidores y de los campus registrados. Cada método hace una consulta a la base de datos para insertar u obtener información.

- **Clase AccesoBDD:** Realiza una conexión con la base de datos y comprueba que dicha conexión se haya establecido correctamente. Además dispone de unos métodos que le permiten ejecutar las sentencias *SQL* en la base de datos para realizar acciones sobre ella. También dispone de unos métodos para guardar los datos que lee de la BD.
- **Clase DireccionIP:** Representa una dirección *IP* que se lee de un archivo de configuración *XML* y la devuelve en forma de Cadena de texto. Esta dirección nos indica donde se aloja la BD.
- **Clase Datos:** Dimensiona las matrices con el tamaño adecuado donde se guardarán los datos que se leen de la base de datos.

Clases de la aplicación cliente

En la figura siguiente se muestran las clases que pertenecen a la aplicación cliente.

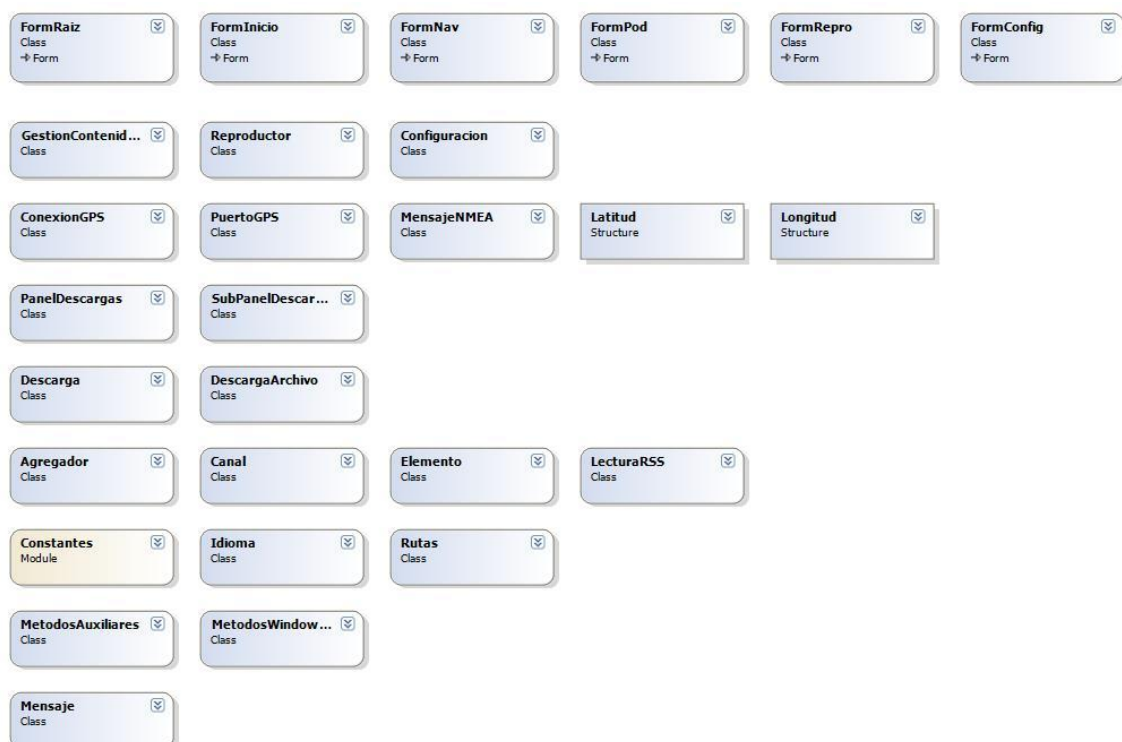


Fig. A1.2 Clases de la aplicación cliente

- **Clase FormRaiz:** Esta clase se encarga de crear un formulario con un menú principal, el cual también gestiona. Este formulario actúa como contenedor para los otros formularios.
- **Clase FormInicio:** Se encarga de crear el formulario de inicio, que muestra información al cliente acerca del estado de la aplicación. Si está activado el *GPS* muestra las coordenadas donde está ubicado actualmente.
- **Clase FormNav:** Actúa de contenedor de un navegador Web Internet Explorer donde se cargarán las páginas Web del sistema. Además se encarga de llamar a los servicios Web “ServicioServidorDirecto” y “Servicio ServidorCercano”.
- **Clase FormPod:** Este formulario se encarga crear una interfaz gráfica donde el usuario podrá realizar diversas acciones sobre los *podcast*.
- **Clase FormRepro:** Actúa de contenedor del reproductor que se usará para abrir los archivos descargados. Este reproductor variará en función del tipo de archivo seleccionado.
- **Clase FormConfig:** Este formulario se encarga de ofrecer al cliente la selección de una serie de opciones y guardarlas o cargarlas para sesiones posteriores.
- **Clase GestiónContenidos:** Esta clase se encarga de gestionar los *podcast*, suscribirse y darse de baja de un canal mediante la clase agregador; también gestiona los archivos descargados, los borra o los reproduce; y por último realizar descargas de elementos no descargados.
- **Clase Reproductor:** Esta clase se encarga de llamar al programa adecuado para abrir un archivo y lo incrusta en el formulario creado por FormRepro.
- **Clase Configuración:** Se encarga de guardar los valores de las diferentes opciones de “FormConfig”.
- **Clase ConexionGPS:** A través de esta clase se controlan los accesos al Objeto “PuertoGPS”.

- **Clase PuertoGPS:** Representa a un puerto del dispositivo. Se encarga de escanear, abrir y cerrar puertos; encontrar por cuál de ellos recibe la señal *GPS* e intenta interpretar los datos recibidos.
- **Clase MensajesNMEA:** Representa un objeto de un mensaje *NMEA*, donde almacena los datos de los mensajes recibidos.
- **Estructura Latitud:** Esta estructura guarda la latitud de la posición en grados, minutos y segundos. Además guarda la dirección.
- **Estructura Longitud:** Esta estructura guarda la longitud de la posición en grados, minutos y segundos. Además guarda la dirección.
- **Clase PanelDescargas:** Representa un control de tipo panel, que añade elementos dependiendo del número de descargas que hay activas actualmente.
- **Clase SubPanelDescargas:** Se encarga de proporcionar a *PanelDescargas* los controles que añadirá al Panel.
- **Clase Descarga:** Se encarga de realizar las descargas de los archivos XML y de instanciar la clase *DescargaArchivo* cuando sea necesario.
- **Clase DescargaArchivo:** Su función es la de descargarse un elemento de un canal y de actualizar el panel de descargas.
- **Clase Agregador:** Esta clase representa a un agregador de contenidos. Analiza los RSS mediante la clase *LectorRSS* y almacena la información en objetos *Canal*. También se encarga de gestionar los canales: suscribirse, darse de baja y actualizarlos.
- **Clase Canal:** Representa un canal de un *podcast*. Contiene el nombre, descripción y *URL*. Además cada canal contiene una serie de objetos de la clase *Elemento*.
- **Clase Elemento:** Representa un elemento de un canal de *podcast*. Contiene nombre, *URL* y descripción. Además contiene información sobre el estado del elemento, si está descargado o si se está descargando.
- **Clase LectorRSS:** Permite interpretar el contenido un archivo *RSS* a partir de un documento *XML* y guardarlo en un objeto *Canal*.

- **Módulo Constantes:** Módulo donde se almacena el valor de todas las constantes usadas por la aplicación.
- **Clase Idioma:** Se encarga de cambiar el idioma de todos los textos que aparecen en la interfaz, dependiendo de si el idioma especificado en “FormConfig” es catalán, castellano o inglés. Para ello hace uso del Módulo “Constantes”.
- **Clase Rutas:** Esta clase se encarga de almacenar y devolver las rutas donde están ubicados los directorios relativos a la aplicación. En caso de que el usuario borre algún directorio, lo vuelve a crear.
- **Clase MetodosAuxiliares:** Permite usar una serie de métodos que necesitan algunas clases pero que no son propios de éstas.
- **Clase MetodosWindowsApi:** Permite usar funciones de la librería “coredll.dll” que son propias del Sistema Operativo del dispositivo móvil.
- **Clase Mensaje:** Esta clase es la que se encarga de mostrar al usuario cajas de mensaje de información, error o advertencia con el texto adecuado.

Diagrama de secuencia

Un diagrama de secuencia muestra cómo se comportan y se relacionan las diferentes clases que pertenecen a una aplicación desde el momento en que ésta se inicia.

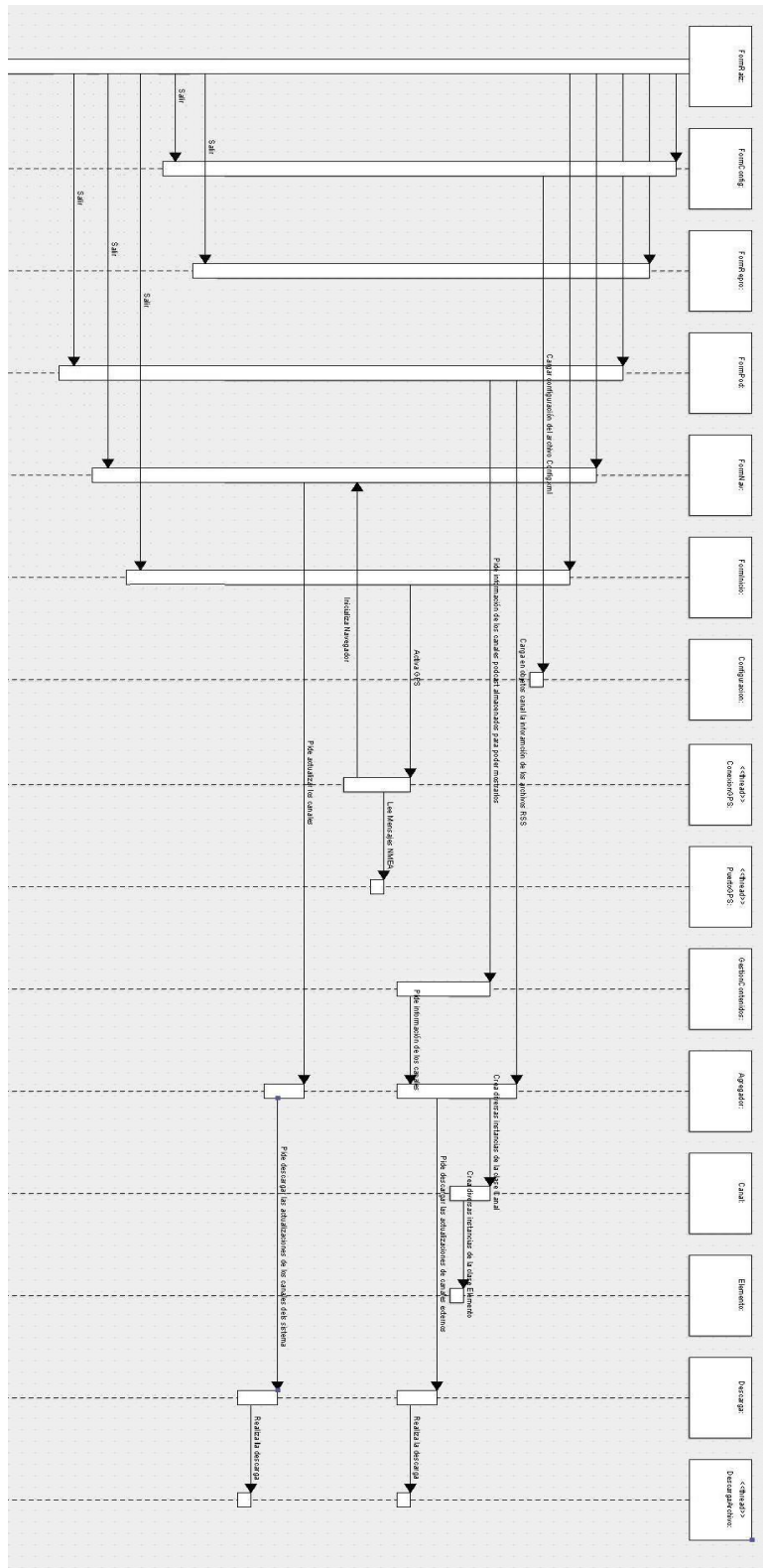


Fig. A1.3 Diagrama de secuencia

En la figura A1.3 se muestra un diagrama de secuencia simplificado de la aplicación cliente. En este diagrama solamente se ha tenido en cuenta el inicio y el final de la ejecución del programa.

Además de lo que se puede observar en este diagrama, el usuario puede activar eventos que realizarán acciones determinadas.

La clase “FormRaiz” es la clase principal de la aplicación. Desde esta clase, se instancian los demás formularios. Una vez se han creado los objetos de los formularios, “FormConfig” se encarga de cargar la información del archivo “config.xml”, estableciendo así el valor de sus controles. Además, guarda estos valores en una instancia de la clase “Configuracion”, a la cual podrán acceder las diferentes clases cuando necesiten consultar algún parámetro de la configuración.

A continuación, “FormPod” crea un objeto “Agregador”, que creará tantos objetos “Canal” como canales esté suscrito el usuario. También creará un objeto “Elemento” por cada uno de los *items* de cada canal. “Agregador” accede a la clase “Configuracion” y consulta si las descargas automáticas están activadas.

De ser así, crea un nuevo objeto de la clase “Descarga” y llama a uno de sus métodos para realizar la descarga del archivo RSS del canal a actualizar. Una vez obtiene dicho archivo, lo compara con el objeto “Canal” correspondiente. Si encuentra elementos nuevos, llamará al método de “Descarga” correspondiente. Este método se encargará de crear un *thread* por cada archivo que deba descargar. De esta forma se conseguirán descargar varios archivos al mismo tiempo, y seguir con la ejecución del hilo principal. Esto solamente ocurre para el caso de descargas externas, ya que aun no se habrá conectado a ningún campus.

Mientras se está ejecutando este *thread*, “FormPod” crea una instancia de “GestionContenidos”, que es la clase encargada de la gestión de los archivos y de los RSS. “GestionContenidos”, actualiza todos los controles de “FormPodcast” y realiza una llamada al *ServicioListaPodcast*, el cual le devuelve una lista con los títulos de los *podcast* del sistema.

Cuando se crea una nueva instancia de “FormInicio”, éste activará un *thread* (método de “ConexionGPS”) que iniciará el proceso de búsqueda de la señal GPS por los diferentes puertos del dispositivo. Para ello creará un objeto “PuertoGPS”. Cuando encuentre el puerto por el cual llegan la señal, “ConexionGPS” activará un *thread* que llamara al método para leer por el puerto de “PuertoGPS”, que será el encargado de leer e interpretar la información de los mensajes NMEA.

Una vez se procesa la señal, si hay datos útiles, “ConexionGPS” llama al método “Inicializar” de la clase “FormNavegador” y le pasa las coordenadas del cliente. Éste método es el encargado de llamar al servicio Web *ServicioServidorCercano*. También se encarga de actualizar los canales si la opción actualizaciones automáticas está activa.

Al cerrar la aplicación, será “FormRaiz” el encargado de llamar a los métodos apropiados de cada clase para cerrar los formularios y abortar los hilos.

ANEXO 2. PORTAL WEB

Uno de los servicios ofrecidos al estudiante es el acceso a un portal Web adaptado para dispositivos móviles (*.NET Mobile*). En función del dispositivo que acceda a estas páginas (*PDA*, teléfono móvil u ordenador), el servidor devuelve un código *HTML* diferente para cada uno de ellos. De este modo se consigue adaptar las páginas Web para que se visualicen correctamente en todos los dispositivos. Este portal Web es un pequeño ejemplo de las posibilidades que ofrece *.NET Mobile* y está formado por un conjunto de páginas Web.

La aplicación cliente dispone de un navegador Web, en el cual se cargará directamente la página Web principal en cuando se conecte con algún servidor.

Este portal Web consta de 4 secciones: “Noticias”, “Podcast”, “Servicios” y “Contacto”.

Se podrá acceder a cada una de estas secciones mediante diferentes enlaces a través de la página de inicio, figura 3.17.



Fig. 3.17 Página Web “Inicio” para Móviles

Noticias

En la sección “Noticias”, como su propio nombre indica, se pueden encontrar los titulares de las últimas noticias referentes a la universidad. Cuando se presiona sobre un titular, se cargará una página Web con el cuerpo de la noticia. Ver figura 3.18.



Fig. 3.18 Página Web “Noticias” para Móviles

Podcast

En la sección “Podcast” se pueden encontrar dos enlaces, tal y como se puede observar en la figura 3.19. El primero de ellos abre una página Web donde se listan todos los canales de *podcast* disponibles en los servidores. Estos datos son solamente informativos.

El segundo de los enlaces que se encuentran en esta sección, carga una página Web que permite descargar la aplicación cliente y desde esta página también se puede descargar el manual de usuario de dicha aplicación.



Fig. 3.19 P Página Web “Podcast” para Móviles

Servicios

En la sección “Servicios” se encuentra un enlace a una página Web donde se listan una serie de aplicaciones. Se incluye una calculadora donde el usuario podrá realizar operaciones simples como sumar, restar, dividir y multiplicar dos números. Una vez decida qué operación quiere realizar bastará con que pulse el botón “Veure resultat” para que se muestre el resultado en la página Web.



Fig. 3.20 Página Web “Servicios” para Móviles

Contacto

En la sección de “Contacto” el usuario podrá encontrar unas direcciones de correo de los administradores del sistema para cualquier sugerencia que quiera realizar.



Fig. 3.21 Página Web “Contacto” para Móviles

Todas las páginas anteriores tienen enlaces para volver a la página de inicio o volver a la página anterior. De esta forma se facilita la navegación por las páginas Web.

ANEXO 3. DESCRIPCIÓN DE LOS MENSAJES DE LA APLICACIÓN

Mensajes de Error

- Cuando se presiona el botón “Suscripción Externa” o el botón “Suscribirse” en la pestaña “Suscripción” del formulario “Podcast” y el usuario ya está suscrito al canal seleccionado.

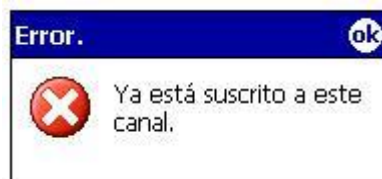


Fig. A3.1 Mensaje de Error

- Cuando el usuario se da de baja de un canal y borra todos los archivos asociados. Si algún archivo no se puede borrar.

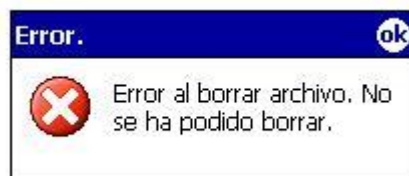


Fig. A3.2 Mensaje de Error

- Como en el caso anterior pero si no se encuentra el directorio donde deberían estar los contenidos.

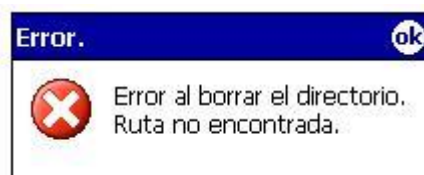


Fig. A3.3 Mensaje de Error

- Cuando el usuario actualiza un canal y éste no se puede actualizar debido a que no hay publicados nuevos contenidos. En lugar de “XXX” aparecerá el nombre del canal.

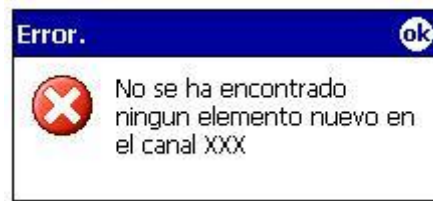


Fig. A3.4 Mensaje de Error

- Este mensaje se muestra cuando la clase “ConexiónGPS” no encuentra ningún puerto por el cual llega la señal *GPS*.

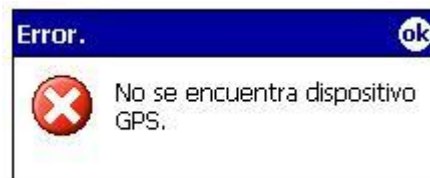


Fig. A3.5 Mensaje de Error

- Cuando se aborta el hilo de ejecución que procesa los mensajes *GPS*.



Fig. A3.6 Mensaje de Error

- Cuando se intenta descargar un archivo *XML*, es decir, un *RSS*, y no se encuentra dicho archivo en la *URL* utilizada para la descarga.

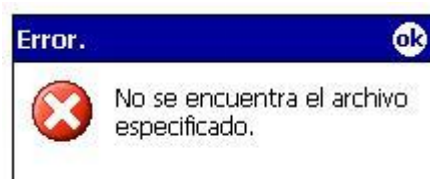


Fig. A3.7 Mensaje de Error

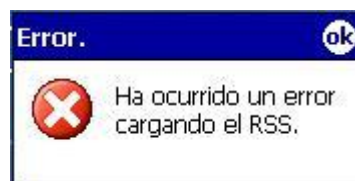
- Cuando se intenta descargar un *XML* y la *URL* utilizada no es válida.

**Fig. A3.8** Mensaje de Error

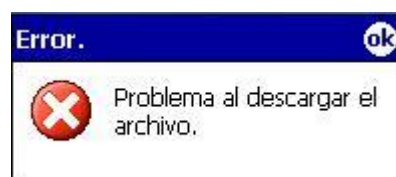
- Cuando se intenta descargar un archivo *XML*, y el dispositivo no tiene acceso a Internet.

**Fig. A3.9** Mensaje de Error

- Si ocurre algún error cuando se descarga un archivo *XML* y se intenta guardar en el dispositivo.

**Fig. A3.10** Mensaje de Error

- Si ocurre algún error durante la descarga de un archivo de un canal.

**Fig. A3.11** Mensaje de Error

- Este error puede aparecer en más de una ocasión.

Si en la pestaña “Archivos” del formulario “Podcast” se presiona sobre el botón “Borrar” y no se ha introducido ningún canal en el cuadro desplegable de la pestaña, aparecerá el error de la figura A3.12.

En la pestaña “Descargas” del formulario “Podcast”, si se pulsa el botón “Aceptar” o el botón “Descargar” sin haber seleccionado antes un canal de la lista desplegable del formulario.

En la pestaña “Suscripción” del formulario “Podcast”, si se presiona sobre el botón “Suscribirse” sin haber seleccionado un canal de la lista.



Fig. A3.12 Mensaje de Error

- En la pestaña “Archivos” del formulario “Podcast”, si se presiona sobre el botón “Aceptar” o sobre el botón “Borrar” y no hay ninguna *Checkbox* activada.

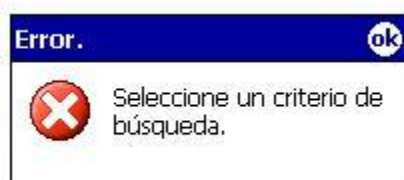


Fig. A3.13 Mensaje de Error

- En la pestaña “Descargas” del formulario “Podcast”, si se pulsa sobre el botón “Descargar” sin que se haya seleccionado un elemento de la lista.

También se muestra este error si se presiona sobre el botón “Borrar” sin haber seleccionado un elemento de la lista en la pestaña “Archivos” de este mismo formulario



Fig. A3.14 Mensaje de Error

- En el formulario "Podcast", en la pestaña "Archivos" y "Descargas", si se presiona sobre el botón de información y no hay ningún canal seleccionado.

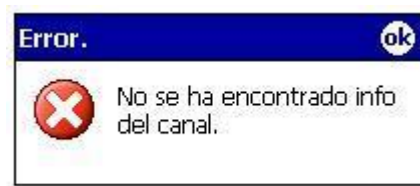


Fig. A3.15 Mensaje de Error

- En el formulario "Podcast", en la pestaña "Archivos" y "Descargas", si se presiona sobre el botón de información y no hay ningún elemento seleccionado.

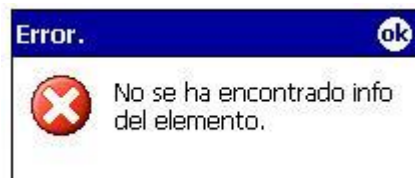


Fig. A3.16 Mensaje de Error

- En el formulario "Podcast", en la pestaña "Archivos" si se presiona sobre el botón "Aceptar" y el usuario ya tiene descargados todos los archivos del canal seleccionado.

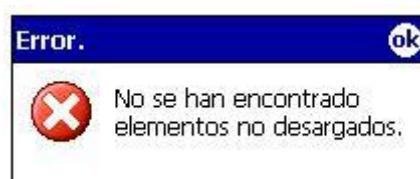


Fig. A3.17 Mensaje de Error

- En el formulario “Podcast”, en la pestaña “Archivos” si se presiona sobre el botón “Aceptar” y no se encuentra el canal seleccionado.

En la pestaña “Suscripción” del formulario “Podcast”, si se presiona sobre el botón “Suscribirse” y no se encuentra el canal seleccionado.

En el mismo formulario que los casos anteriores, pero en la pestaña “Descargas”, si se presiona sobre el botón “Aceptar” y no se encuentra el canal seleccionado.



Fig. A3.18 Mensaje de Error

- En el formulario “Podcast”, en la pestaña “Descargas” si se pulsa sobre el botón “Descargar” y el usuario tiene seleccionado un archivo de la lista que ya ha descargado anteriormente.



Fig. A3.19 Mensaje de Error

- En la pestaña “Descargas” del formulario “Podcast”, si el usuario intenta descargar un archivo de la lista que se está descargando.

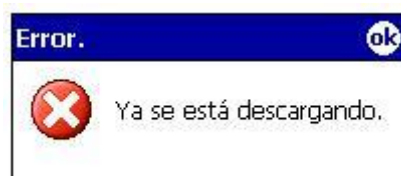


Fig. A3.20 Mensaje de Error

- En la pestaña “Descargas” del formulario “Podcast”, si el usuario intenta descargar un archivo de la lista que por algún motivo no ha encontrado.

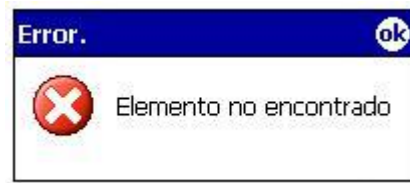


Fig. A3.21 Mensaje de Error

- En el formulario “Podcast”, en la pestaña “Archivos” si presiona el botón “Reproducir” sin seleccionar antes un archivo de los que tiene descargados.

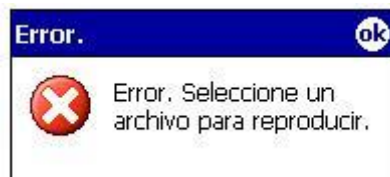


Fig. A3.22 Mensaje de Error

- Cada vez que la aplicación invoca a un servicio Web, se controla que el resultado de este proceso sea correcto. De no ser así, se mostrará el siguiente mensaje al usuario, explicando que la invocación al servicio ha fallado.



Fig. A3.23 Mensaje de Error

- Si la aplicación necesita acceder a Internet y el dispositivo ha perdido su conexión, se mostrará el mensaje de error de la figura A3.24.

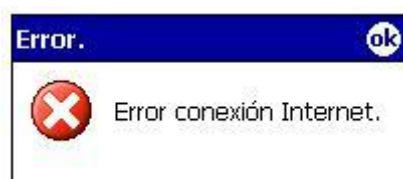
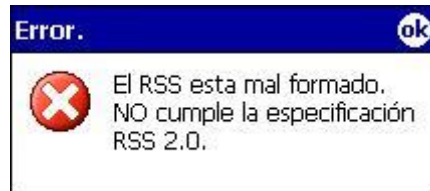
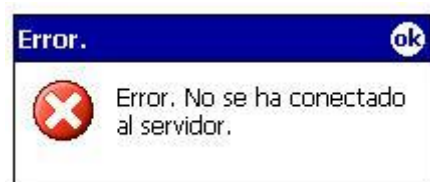


Fig. A3.24 Mensaje de Error

- La clase “LecturaRSS” se encarga de interpretar los archivos *RSS*. Si detecta que alguno no cumple la especificación, la aplicación lanzará el mensaje de error siguiente.

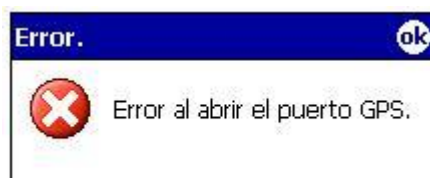
**Fig. A3.25** Mensaje de Error

- Si se intenta descargar un archivo sin haberse conectado previamente a un servidor del sistema.

**Fig. A3.26** Mensaje de Error

- Si ocurre un error mientras la aplicación intenta abrir un puerto del dispositivo para buscar la señal *GPS*.

Puede que intente abrir un puerto que ya esté abierto o que no exista.

**Fig. A3.27** Mensaje de Error

- Si el reproductor no puede abrir el archivo que el usuario ha seleccionado para reproducir.

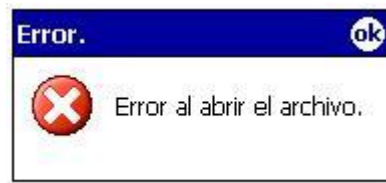


Fig. A3.28 Mensaje de Error

- Si la aplicación realiza alguna operación no permitida mientras intenta cerrar el programa de reproducción de los archivos.



Fig. A3.29 Mensaje de Error

- Cuando, desde el formulario "Configuración", se intentan guardar los datos de configuración para posteriores sesiones, si la aplicación está realizando alguna operación sobre el archivo "config.xml", el usuario deberá esperar un tiempo hasta que el archivo deje de estar en uso para poder guardar su configuración.

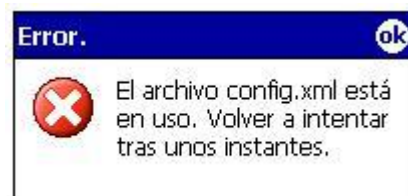


Fig. A3.30 Mensaje de Error

- Cuando la aplicación se intenta conectar a un campus y no puede establecer la conexión con ninguno de sus servidores.

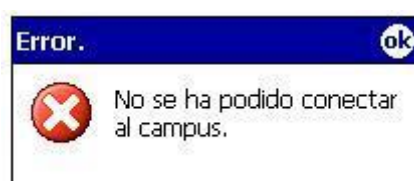


Fig. A3.31 Mensaje de Error

- Desde el formulario “Configuración” el usuario puede cambiar la orientación de la pantalla. Si surge algún problema mientras se realiza este proceso, aparecerá el siguiente mensaje de error.

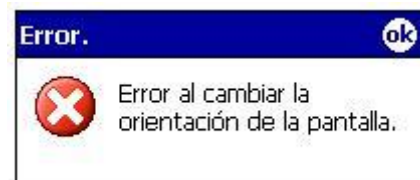


Fig. A3.32 Mensaje de Error

- Si el archivo de configuración está dañado y la aplicación no lo puede leer, o bien ha encontrado alguna modificación no permitida en el contenido del archivo.

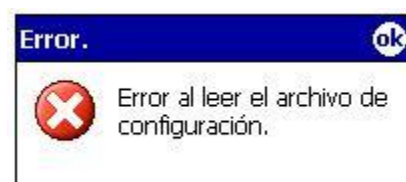


Fig. A3.33 Mensaje de Error

- Hasta que la aplicación no se conecte a ningún servidor, las descargas de archivos están restringidas. En cuando el dispositivo se conecte a un servidor de algún campus, se podrán descargar los elementos de los canales.

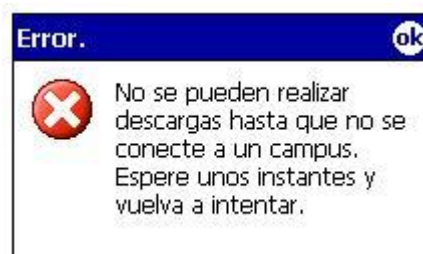


Fig. A3.34 Mensaje de Error

Mensajes de Información

- En la pestaña “Suscripción” del formulario “Podcast”, si se mantiene pulsado un canal de la lista, aparecerá un menú contextual con una opción. Si se pulsa sobre ella, aparecerá un mensaje de información que mostrará al usuario el tipo de archivos que contiene el canal. Por ejemplo, archivos de audio.



Fig. A3.35 Mensaje de Información

- Si se presiona sobre el botón con el símbolo de interrogación del formulario “Configuración”, aparecerá el siguiente mensaje, que describe la acción que realizará el botón “Guardar cambios”

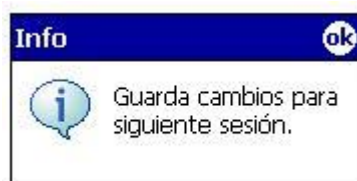


Fig. A3.36 Mensaje de Información

- Cuando el cliente se suscribe a un canal, sea éste externo o no, aparecerá el siguiente mensaje si el proceso de suscripción finaliza con éxito.

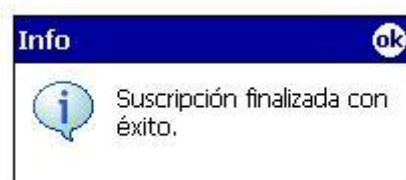


Fig. A3.37 Mensaje de Información

Mensajes de Advertencia

- Cuando un usuario se da de baja de un canal, la aplicación le preguntará si quiere borrar todos los archivos que tiene descargados de ese canal.

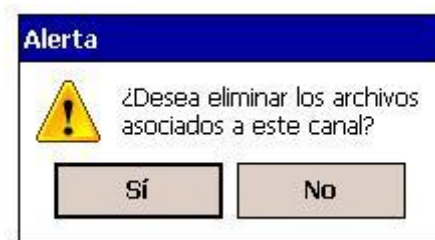


Fig. A3.38 Mensaje de Alerta

- Cuando el usuario actualiza manualmente un canal, aparecerá el siguiente aviso. Si las actualizaciones están en modo automático, no se mostrará el mensaje de alerta al usuario, sino que se descargarán los nuevos elementos directamente.

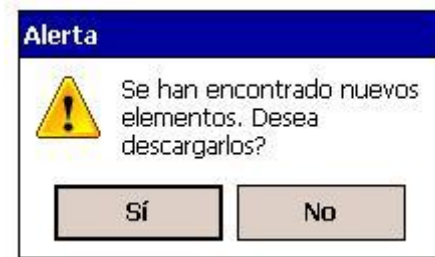


Fig. A3.39 Mensaje de Alerta

- Si el usuario no tiene activada la opción de usar puertos automáticos que se encuentra en el formulario "Configuración", y la aplicación detecta que llegan señales de *GPS* por más de un puerto, se le pedirá al usuario que seleccione un puerto por el cual leer los mensajes *NMEA*.

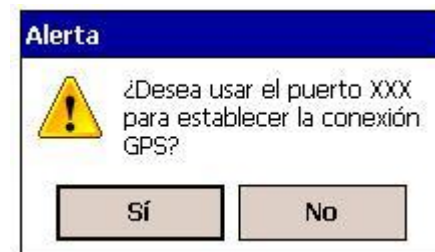


Fig. A3.40 Mensaje de Alerta

- Cuando el usuario borre un elemento mediante el botón "Borrar" de la pestaña "Archivos" del formulario "Podcast", se le mostrará el siguiente

mensaje de advertencia. De esta forma se evita que un usuario borre un archivo accidentalmente.

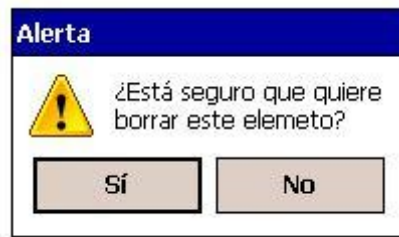


Fig. A3.41 Mensaje de Alerta

- Cuando el dispositivo está conectado con un servidor de un campus y el *GPS* está activado, si el usuario se desplaza el campus más cercano puede cambiar. Si sucede esto, se le preguntará al usuario si desea conectarse a algún servidor de este campus.



Fig. A3.42 Mensaje de Alerta