Sistema de posicionamiento e información de localización sobre Windows Mobile 7

Ingeniería Técnica en Informática de Gestión.

Autor:

Alberto Benetó Micó

Director:

Dr. Juan Carlos Cano Escribá

Julio de 2011.

Escola Tècnica Superior d'Enginyeria Informàtica etsinf

DISCA

1 Introducción

Nadie puede negar que, hoy en día, la informática está presente en casi todos los aspectos de nuestra vida diaria. El avance de este campo ha posibilitado cosas impensables, o que hace unos años nos parecerían imposibles.

Poca gente se atrevía siquiera a aventurar hace 10 años, que un dispositivo como podía ser un teléfono móvil pudiera albergar la capacidad de almacenamiento, de proceso y de conectividad de la que disponen hoy en día.

Pero actualmente, los dispositivos móviles han roto todas las expectativas y ocupan un lugar importante en el mundo, y especialmente, en el campo de la informática, al que van ligados sin lugar a dudas.

Hoy por hoy, un dispositivo móvil puede hacer prácticamente todo aquello que se nos antoje. Podemos comprar, hacer cualquier cálculo, fotografías, usarlo como GPS, como agenda, mantenernos conectados con todo tipo de redes sociales en tiempo real...y bueno, ¡también realizan y reciben llamadas, por supuesto!

Gracias al gran avance que han tenido, tienen, y seguirán teniendo estos dispositivos, hoy en día podemos desarrollar proyectos como este, que estoy seguro que dentro de unos años se quedarán obsoletos frente a la gran capacidad de proceso con que contarán los dispositivos del momento.

1.1 Motivación

Personalmente, vi la posibilidad de enfrentarme a un proyecto de esta envergadura como un reto. Es un proyecto muy ambicioso, que bien llegado puede sentar precedente en otras instalaciones similares donde el clásico mapa de papel acaba siendo pesado y a veces, difícil para saber dónde estamos realmente. Además, se apuesta por un S.O como es Windows Phone 7, que hasta donde conozco es un S.O que plantea muchas facilidades para el desarrollo de aplicaciones, y por ende, esto repercute en mejores y más elaboradas aplicaciones. También apostamos por una arquitectura Cliente-Servidor, referente hoy en día en muchos campos de la informática, y que combinado con el uso de bases de datos SQL nos dan mucho juego.

1.2 Objetivos

El objetivo principal de este proyecto es desarrollar un sistema para el Sistema Operativo Windows Phone 7 que constará de dos partes fundamentales. La primera parte constará de fichas y fotos de los animales del parque, para saber un poco más sobre ellos. La segunda parte constará de un sistema de Geo-Posicionamiento capaz de indicarnos en todo momento, en qué punto del parque estamos, que animales tenemos cerca (10 metros) y ser capaces de visualizar información de dichos animales con tan sólo pulsar su foto.

1.3 El sistema actual

Actualmente, el sistema utilizado en el parque Bioparc de Valencia, no difiere mucho de lo que se utiliza en diferentes parques, tanto de este estilo, como de atracciones, etc. Básicamente se trata de un mapa, en papel, en el que obtenemos una representación del parque, no una foto a satélite real, y en cada punto donde se encuentra un animal o punto de interés, se señaliza con un número. A los laterales del mapa obtenemos una lista con estos números y lo que hay en cada uno de ellos. Con este método, si queremos conocer información de algún punto, tenemos que dirigirnos a los paneles que existen en este punto. Con esto, evidentemente, es imposible usar algún elemento multimedia para hacer más amena y divertida la visita. El mapa en cuestión se encuentra aquí:

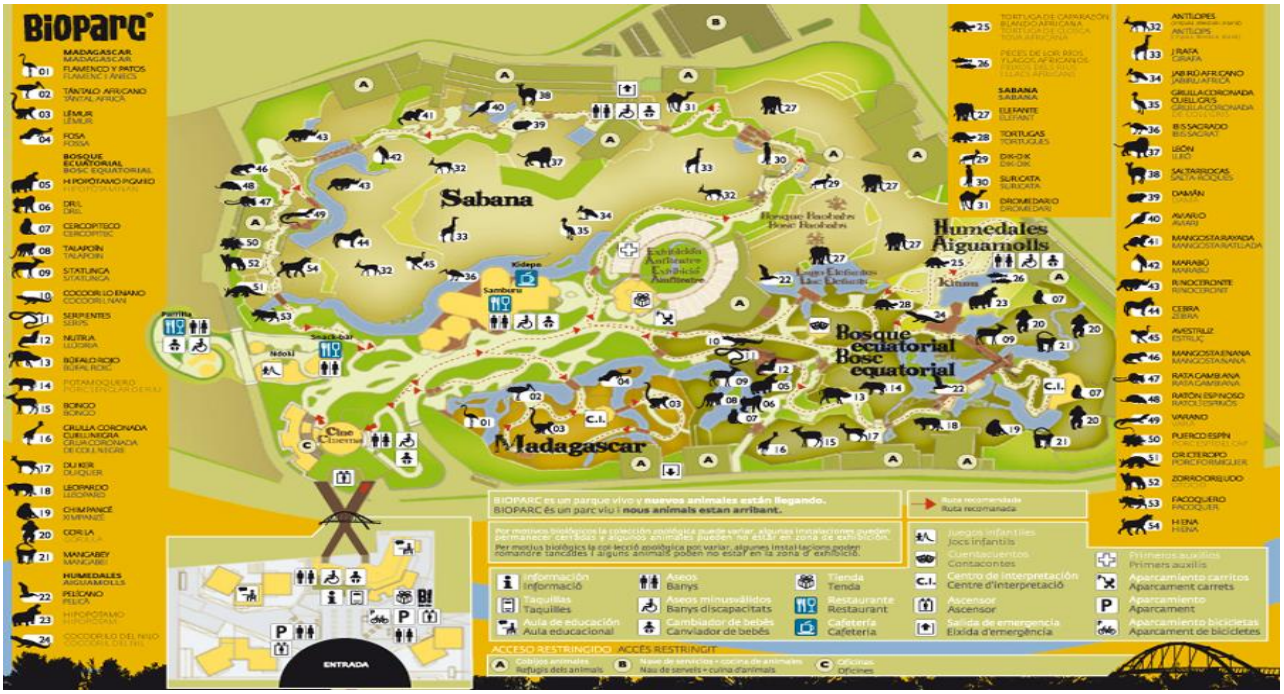

Figura 1: Mapa del parque Bioparc de Valencia

1.4. Windows Phone 7 como Sistema Operativo a usar en el cliente.

La decisión de usar un Sistema Operativo u otro depende de muchos factores que tenemos que tener en cuenta. No se puede negar que, hoy en día, en el mercado prima el coste económico, y que, por tanto, un Sistema Operativo más rentable o que cueste menos de mantener tiene todas las papeletas para ser el elegido. Pero no sólo este aspecto es importante, también hay otros aspectos a considerar como la facilidad de uso, que incluya librerías, el entorno de desarrollo que se va a utilizar, etc. Hay que reconocer que Windows Phone no es actualmente el Sistema Operativos para dispositivos móviles más usado. Pero aun así, se encuentra entre los primeros. Además, la constante evolución de estos dispositivos hacia dispositivos con la filosofía de "este siempre conectado, donde y cuando quiera", hace que sigan una evolución lineal en el tiempo con un crecimiento constante pero imparable

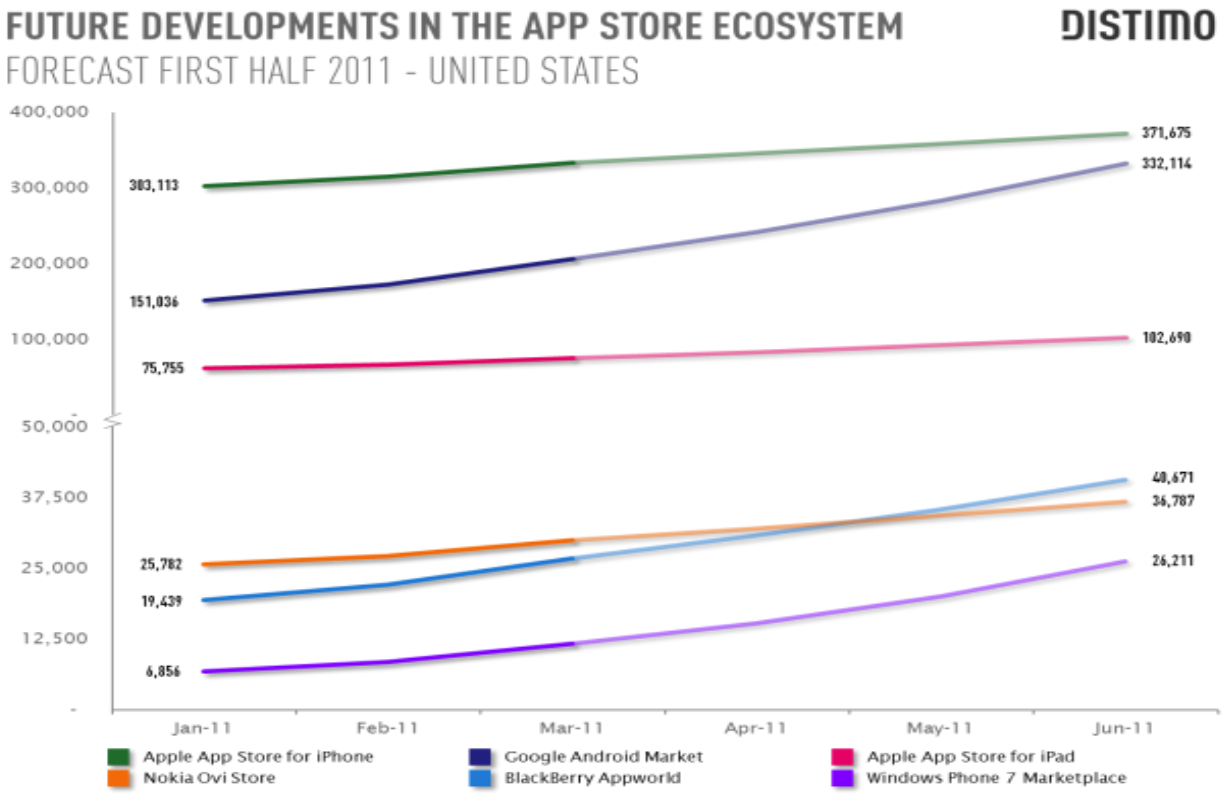

Figura 2: Situación actual y evolución que se esperaba de los S.O para dispositivos móviles

La situación real y actual de los distintos sistemas operativos que se utilizan en los Smartphone, es la ilustrada en la siguiente gráfica:

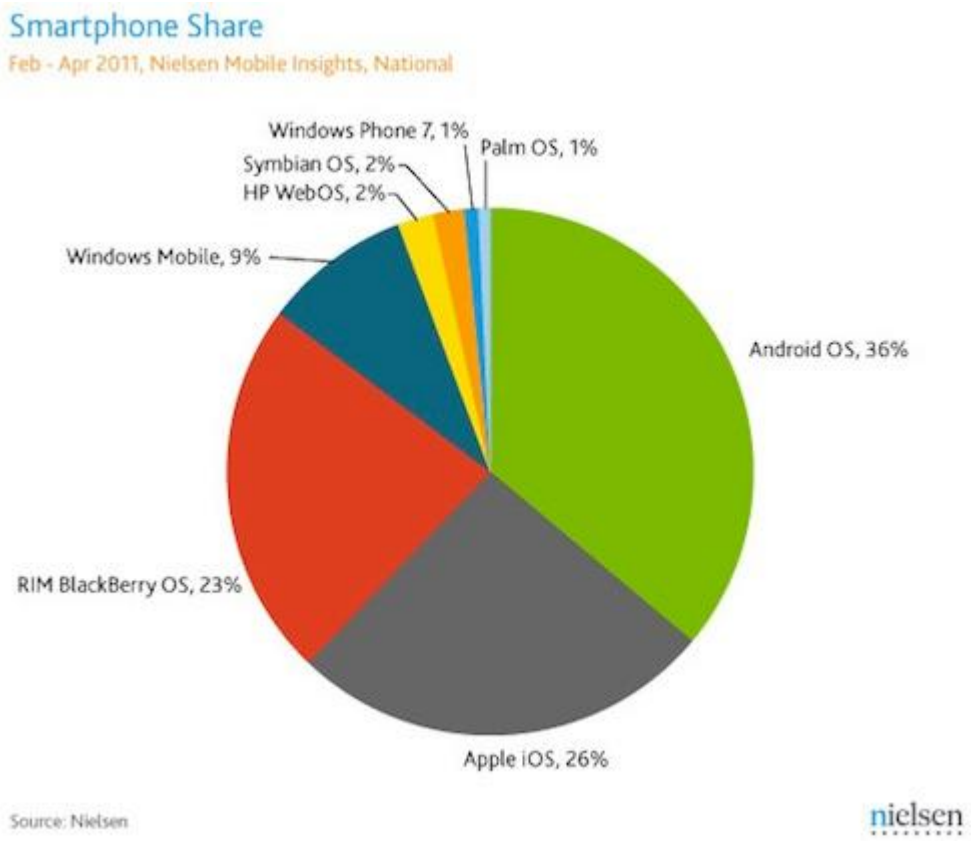

Figura 3: Porcentaje actual del uso de OS en Smartphone

En el caso de Windows Phone 7, hay que tener en cuenta que es un sistema prácticamente nuevo, ya que hasta hace bastante poco, el sistema operativo para dispositivos móviles de Microsoft era el llamado Windows Mobile, con un 9%. Sumado al 1% de los Windows Phone 7, hace un total de

10%, superando claramente a otros sistemas como Symbian OS, Palm Os, y HP WebOS, y situándolo en cuarto lugar en el mercado.

Con todo esto, podemos afirmar que Windows Phone 7 es un sistema que prácticamente acaba de nacer, y que a pesar de ello ha tenido una buena aceptación. Es de esperar que en los próximos años tenga un buen crecimiento como para llegar a estar a la altura de otros como Apple iOS o Android OS.

2 Planteamiento del Problema

El sistema a desarrollar se plantea como una especie de guía para el parque Bioparc de la ciudad de Valencia. Básicamente será un sistema para un dispositivo móvil que constará de dos partes. Una parte en la que podremos obtener información de todos los animales del parque. La información se obtendrá de una especie de fichas con la foto y descripción de las características de los animales, y también tendremos una galería con muchas más fotos de los animales.

Por otra parte, tendremos un mapa interactivo, en el cual el usuario podrá ver la posición actual que ocupa en el parque, y, además, podrá ver que animales tiene cerca en ese momento y tendrá la opción de visualizar su ficha con todos sus datos.

Como la aplicación va a ser utilizada por diferentes tipos de usuarios será necesaria la adaptación del dispositivo a las características determinadas de cada uno de ellos, consiguiendo de esta manera que el dispositivo muestre para cada tipo de usuario la información adaptada a su nivel. Los niveles en los que se ha englobado a los diferentes tipos de usuarios han sido niño, estudiante, adulto, avanzado y experto. Además se ha de tener en cuenta que los usuarios que usen estos dispositivos podrán tener orígenes diversos por lo que se ha incluido la información en varios idiomas. Estas modalidades vienen dadas de hace un tiempo, cuando el estudiante Addrien Deotto empezó el proyecto, y el cual yo he completado.

Este sistema ha de ser utilizada en unos dispositivos móviles con sistema operativo Windows Phone, y el dispositivo debe de tener conectividad WiFi y GPS. Por ejemplo, un dispositivo que cumpliría los requisitos sería el HTC HD7, que se muestra en la siguiente imagen:

Figura 4: Dispositivo móvil HTC HD7, con Windows Phone 7, WiFi y GPS

3 Arquitectura del Sistema

Uno de los aspectos que podríamos considerar clave a la hora de desarrollar un sistema es la definición de la arquitectura de este. La arquitectura no es, ni más ni menos, que como están estructurados los diferentes elementos que conforman el sistema y cómo interactúan entre sí para formar un todo que proporcione una funcionalidad al usuario final.

Aún más importante es la decisión de la arquitectura en los dispositivos móviles, ya que las características tan heterogéneas que tienen estos nos condicionan a la hora de decidir que aplicación queremos montar. En nuestro caso, hemos intentado adaptar la aplicación que hemos desarrollado al mayor número de dispositivos móviles que cumplan tres requisitos principales:

1. Que usen el sistema operativo **Windows Phone 7**
2. Que tengan posibilidad de conectarse a una red vía **WiFi**
3. Que tengan la posibilidad de conocer su posición vía **GPS**

No hemos restringido los dispositivos por memoria, ya que como veremos a continuación, tal y como está estructurado nuestro sistema, el dispositivo móvil no va a tener que almacenar prácticamente nada de información (o nada).

La arquitectura a seguir en nuestra aplicación es la arquitectura **Cliente – Servidor,** aunque siendo muy puristas, también se podría ver como una aplicación de **tres capas**. En este tipo de arquitectura tenemos un cliente (nuestro dispositivo móvil) que pide servicios a un servidor, en nuestro caso, básicamente, de obtención de datos. Este servidor contendrá tanto una pequeña base de datos implementada con **Windows SQL Server 2005**, como un script desarrollado con **ASP** que será el que obtenga los datos de la BD y los devuelva a la aplicación del usuario en formato **XML**. La principal ventaja de este tipo de implementación es que, como los datos los obtenemos en **XML**, es más portable y en caso de tener que cambiar de servidor o de base de datos, sólo tendríamos que cambiar unas órdenes mínimas para que la aplicación siguiera funcionando adecuadamente. El siguiente dibujo resume la arquitectura del sistema:

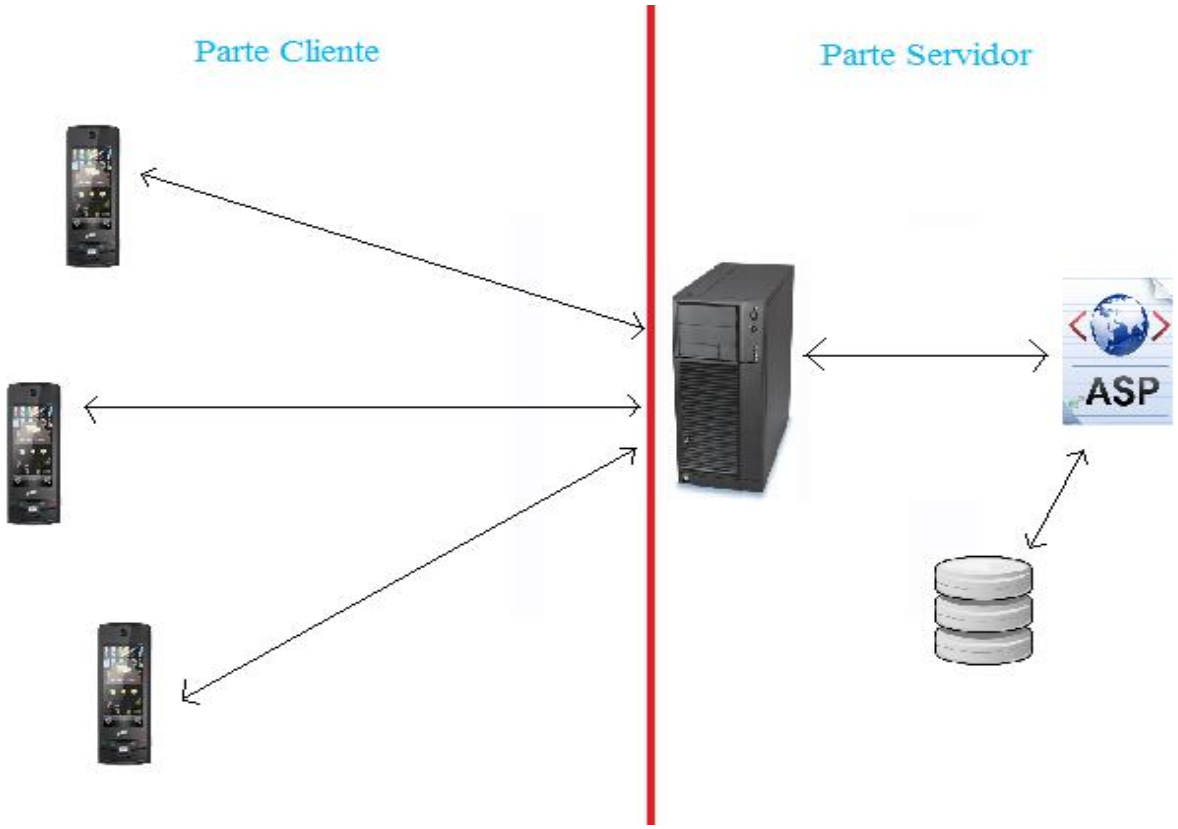

Figura 5: Arquitectura del sistema, Cliente – Servidor.

Hay que aclarar que tanto la base de datos como el script que la consulta residen en la misma máquina, y por tanto como ya hemos afirmado anteriormente, contamos con una arquitectura Cliente / Servidor, aunque si lo vemos como procesos independientes podríamos tener una arquitectura de 3 capas.

Entre otras cosas, una de las cosas que posibilitan la utilización de esta arquitectura y el diseño de la aplicación tal cual lo hemos enfocado es la red WiFi con la que cuenta el parque además de que, al estar al aire libre, la recepción y envío de datos de localización vía GPS es muy precisa.

4 Desarrollo del proyecto

En esta parte de la memoria, posiblemente la más extensa, nos vamos a centrar en que es realmente nuestra aplicación, como está montada y como funciona.

Si lo resumimos a grandes rasgos, la aplicación cliente corriendo en un dispositivo Windows Phone 7, realiza una serie de peticiones a un Servidor previamente configurado. Este servidor dispone de dos partes principales: una base de datos Microsoft SQL Server 2005 y un script ASP. El cliente, cuando realice alguna petición, la realizará sobré el script ASP, que será el encargado posteriormente de realizarle la petición a la base de datos. Una vez obtenidos los datos, nuestro script ASP se encargará de estructurarlos mediante un fichero XML, que será el que finalmente devuelva al usuario final, o sea, a nuestra aplicación cliente.

Pasaremos ahora a ver en detalle las dos partes, el cliente y el servidor.

4.1 Cliente

4.1.1 *Plataforma y lenguaje de desarrollo. Características.*

Para desarrollar aplicaciones sobre dispositivos Windows Phone 7 debemos de contar con una serie de herramientas que nos facilitarán mucho la vida a la hora de ponernos a programar. La principal de estas herramientas es el IDE, en este caso, usamos una versión en concreto del mundialmente conocido entorno de desarrollo Visual Studio .NET. En concreto esta versión se conoce como **“Visual Studio 2010 Express For Windows Phone”.** Este entorno de desarrollo ya

incluye en su instalación todos los kits de desarrollo y librerías necesarias para el desarrollo de una aplicación sobre Windows Phone 7, además de todas las herramientas de depuración y refactorización de código que ya incluyen otras versiones de Microsoft Visual Studio.

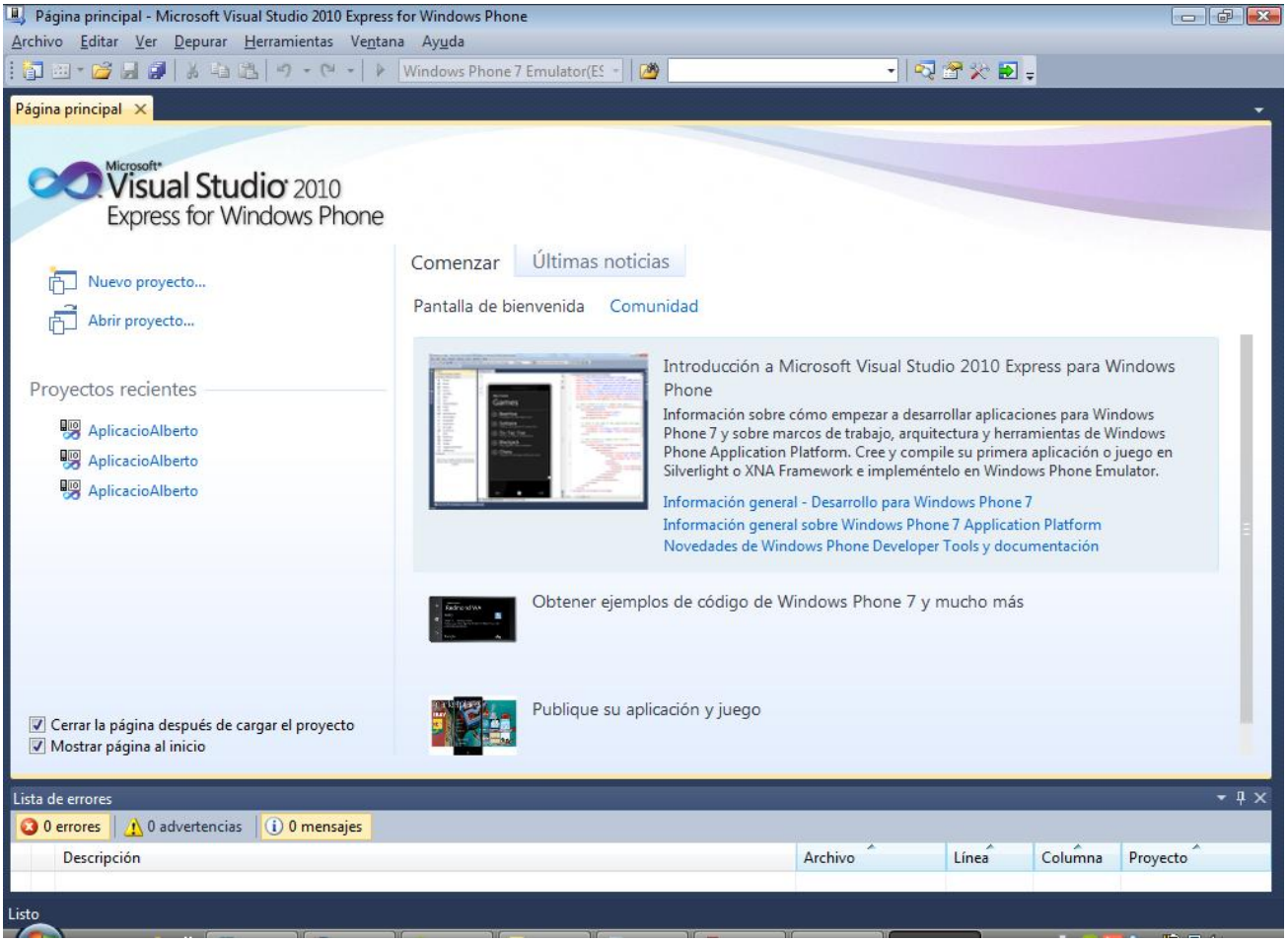

Figura 6: Pantalla principal de Microsoft Visual Studio Express 2010 for Windows Phone

Una novedad que si se incluye en esta versión de Microsoft Visual Studio y que no se incluye en otras versiones es el **"Windows Phone 7 Emulator"** , que no es ni más ni menos que un emulador de un dispositivo móvil estándar con el sistema operativo Windows Phone. Gracias a este emulador, podemos ver en directo, y antes de probarlo en un dispositivo móvil real, como quedaría nuestra aplicación y, sobretodo, como se integraría con los controles más comunes de un dispositivo Windows Phone común, como por ejemplo el botón de atrás, o la tecla de inicio. Además, este

emulador también nos permite cambiar la orientación de la pantalla, para así poder ver como quedaría nuestra aplicación con una orientación u otra.

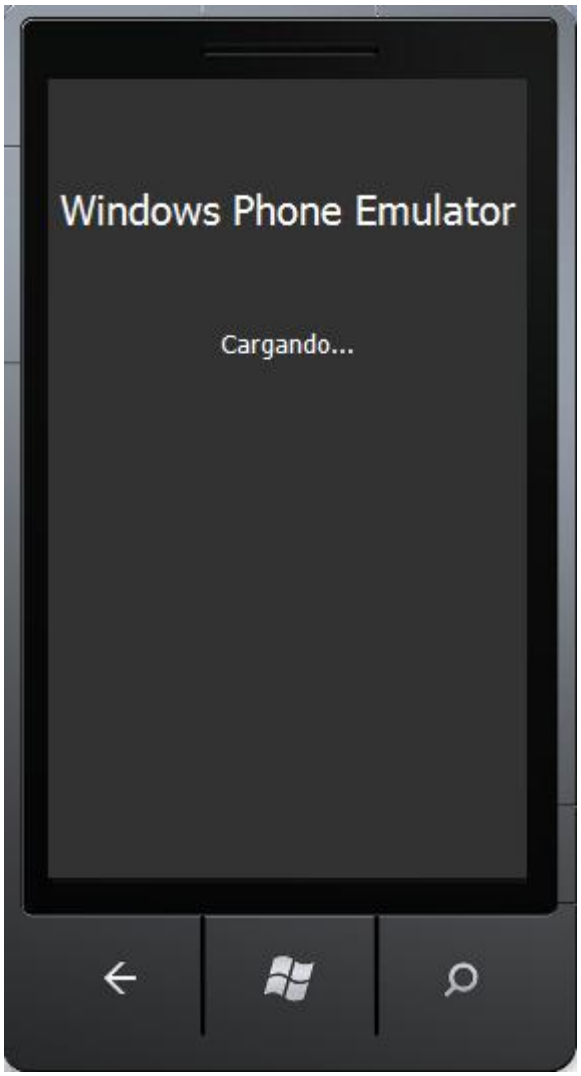

Figura 7: Windows Phone 7 Emulator.

Para desarrollar aplicaciones sobre Windows Phone 7 en la plataforma anteriormente mencionada, Microsoft opta por una clara división entre lo que es la capa de presentación de la capa de lógica. Así, cada clase utilizada en nuestra aplicación cuenta de dos partes. Para la capa de presentación, se utilizan archivos XAML. Un archivo XAML (acrónimo de ***eXtensible Application Markup Language)*** es un lenguaje declarativo basado en XML, optimizado para describir gráficamente interfaces de

usuarios visuales ricas desde el punto de vista gráfico. El aspecto que tiene un archivo en formato XAML es el siguiente:

```
<phone:PhoneApplicationPage
    x:Class="AplicacioAlberto.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    SupportedOrientations="PortraitOrLandscape"
    mc:Ignorable="d" FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}" d:DesignHeight="480"
d:DesignWidth="800" Orientation="Landscape">
       <phone:PhoneApplicationPage.Resources>
             <Color x:Key="Color1">#FFC63232</Color>
       </phone:PhoneApplicationPage.Resources>

    <Grid x:Name="Buttons" >
        <Grid.Background>
            <ImageBrush ImageSource="mainbackground.png" />
        </Grid.Background>

    <Grid.RowDefinitions>
        <RowDefinition Height="*"/>
        <RowDefinition Height="*"/>
        <RowDefinition Height="*"/>
    </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*" />
        </Grid.ColumnDefinitions>
        <Button Name="Spain_language" ClickMode="Press" Click="Spain_language_Click"
Height="160" VerticalAlignment="Bottom" Grid.Row="1">
            <Button.Background>
                <ImageBrush ImageSource="/AplicacioAlberto;component/valencia.png"
Stretch="Uniform" />
            </Button.Background>
        </Button>
        <Button Name="Valencian_language" ClickMode="Press" Click="Valencia_language_Click"
Grid.Row="1" Grid.Column="2">
            <Button.Background>
                <ImageBrush ImageSource="/AplicacioAlberto;component/english.png"
Stretch="Uniform" />
            </Button.Background>
        </Button>
    </Grid>

</phone:PhoneApplicationPage>
```

Mientras que, para la parte de lógica asociada a este fichero XAML se utilizan ficheros en lenguaje C#, uno de los más populares de Microsoft y en concreto de la plataforma Visual Studio .NET. El aspecto de un archivo en C# sería algo como lo siguiente:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;
…

namespace AplicacioAlberto
{
    public partial class MainPage : PhoneApplicationPage
    {
        public MainPage()
        {
            InitializeComponent();

            SupportedOrientations = SupportedPageOrientation.Landscape;

        }

        private void Spain_language_Click(object sender, RoutedEventArgs e)
        {
            Lenguaje.setlenguaje(1);
            gopage2();
        }

        private void Valencia_language_Click(object sender, RoutedEventArgs e)
        {
            Lenguaje.setlenguaje(2);
            gopage2();
        }

        private void English_language_Click(object sender, RoutedEventArgs e)
        {
            Lenguaje.setlenguaje(3);
            gopage2();
        }

        private void gopage2()
        {
            this.NavigationService.Navigate(new Uri("/SelecPerfil.xaml", UriKind.Relative));
        }
}}
```

Esto nos proporciona la máxima independencia entre Interfaz y lógica de la aplicación.

4.1.2 La aplicación Cliente

La aplicación cliente desarrollada sobre Windows Phone 7 está desarrollada para una optima ejecución con el dispositivo en horizontal, de forma apaisada. Empieza con una pantalla inicial, en la que tenemos que elegir el idioma con el que queremos seguir durante el resto de la ejecución del programa:

Figura 8: Pantalla inicial de la aplicación.

Una vez elegido el idioma que queramos, representado cada uno por un animal de los existentes en el parque, aparece la siguiente pantalla. Para esta demostración hemos elegido el idioma español:

Figura 9: Pantalla de selección de perfil de usuario.

En esta pantalla, el usuario elegirá de la lista mostrada, el perfil que más se adapte a su persona. Evidentemente la elección de un perfil u otro condicionará la ejecución de la aplicación, principalmente a la hora de mostrar información, que se adaptará al nivel del usuario. Nosotros elegimos la opción "Adulto", una de las opciones más estándar que hay entre todas las disponibles.

Una vez elegida esa opción, accedemos a la siguiente pantalla:

Figura 10: Pantalla de selección de la información a visualizar

Esta pantalla es prácticamente, la más importante de todas, ya que es aquí donde elegiremos el tipo de contenido a visualizar. Cada icono es ya bastante representativo de por sí, pero por si acaso, encima del mismo botón se le ha añadido un texto con lo que simboliza cada uno. El primer botón es el de fichas, y con el podemos acceder a las fichas de los diferentes animales que hay en el parque. Una ficha consta de foto y descripción del animal. El segundo botón, Imágenes, es una galeria con fotos de todos los animales del parque. Y por último, el tercer botón, Mapa, sirve para entrar al modo en el cual el sistema nos indica en que parte del parque estamos, situándonos sobre un mapa, y nos va indicando poco a poco los animales que tenemos cerca.

Accedemos ahora al primer botón de todos, al de fichas de animales. Desde este botón aparece una lista con todos los animales que tenemos en el parque. Hay que tener en cuenta que, por la cantidad de animales y las características del dispositivo, por lo menos en esta parte del programa, sería

adecuado utilizar algún tipo de lápiz para estos dispositivos, ya que con el dedo puede ser más costoso elegir el animal exacto.

Figura 11: Lista de todos los animales del parque.

Una vez seleccionado el animal de la lista, aparece su ficha. En esta encontramos los siguientes elementos:

- Foto en miniatura del animal a la parte superior derecha.
- Nombre del animal a la parte superior izquierda
- Características del animal en el cuerpo de la ficha.

Aquí tenemos un ejemplo de ficha, en este caso hemos seleccionado el león:

Figura 12: Ficha del León

Con el botón de "Atrás" que incorporan todos los dispositivos móviles basados en Windows Phone, volvemos hasta la pantalla de selección de la información a visualizar, y ahora vamos a visualizar la segunda opción, la galería de imágenes. En esta parte del programa, lo que tenemos es, básicamente, un visualizador de imágenes. Estas se obtendrán desde una base de datos que posteriormente explicaremos como ha estado implementada y como es la comunicación con esta, y se mostrarán por pantalla al usuario final. En esta pantalla tendremos dos botones, uno para pasar las fotos hacia adelante y uno para ir retrocediendo en la lista de fotos que hemos obtenido desde la base de datos. Aquí tenemos una captura de pantalla de este apartado en concreto de nuestra aplicación:

Figura 13: Galería de fotos

Pasamos ahora a la última opción de nuestra aplicación, el sistema de guiado por el parque. A esta opción accedemos volviendo a la parte de selección de la información a visualizar y pulsando sobre el botón del mapa. Una vez pulsado este botón, nos aparece una pantalla con un mapa físico del terreno que ocupa el parque Bioparc, obtenido en tiempo real desde el servicio Google Maps®. Aquí una captura de pantalla:

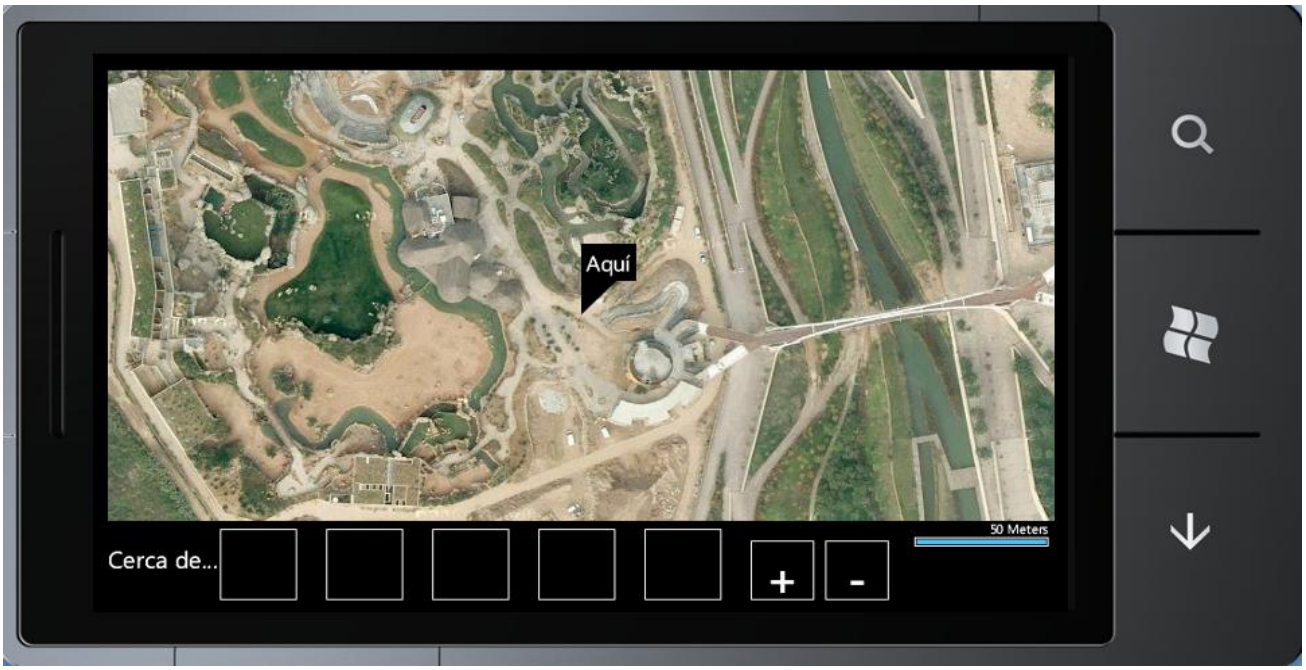

Figura 14: Mapa para visitas guiadas

En este mapa hemos de destacar varias cosas. Tenemos la posibilidad de hacer zoom, o reducir zoom con los botones + o – que aparecen en la parte inferior izquierda del mapa:

Figura 15: Botones para ampliar y reducir el zoom en el mapa.

También tenemos la opción de mover el mapa simplemente arrastrando el dedo por la pantalla podemos movernos por el mapa hacia donde queramos. Pero las dos partes importantes son las que tienen que ver con la Geo-Localización y el aviso de animales cercanos. Sobre el mapa aparece un elemento llamado Pushpin, el cual nos indica en todo momento la posición en que nos encontramos.

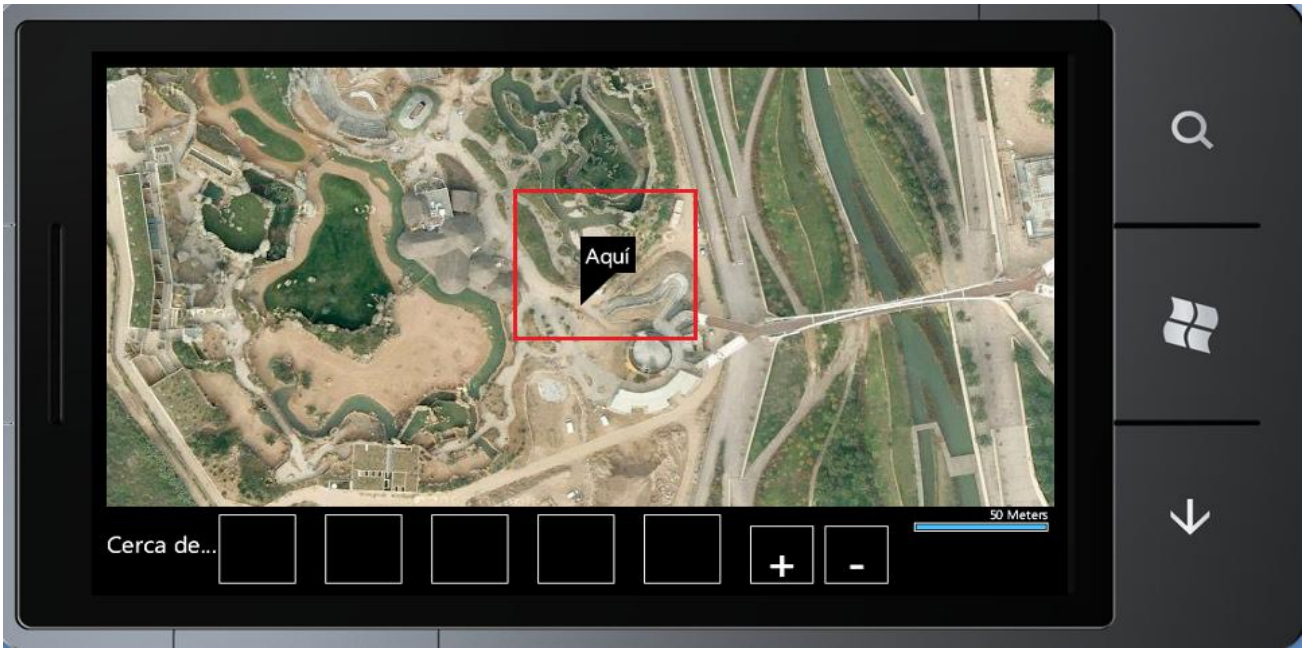

Figura 16: Pushpin que nos indica en todo momento donde nos encontramos.

¿Cómo hemos hecho esto posible? Windows Phone 7 incorpora un elemento llamado GeoCoordinateWatcher que sirve para usar las funciones del GPS que incorporan los dispositivos. Simplemente, sobrescribiremos el método que se dispara cuando se produce una variación de posición en este elemento, en concreto el método PositionChanged, para que actualice la posición del pushpin a la nueva posición en la que nos encontremos. Por tanto, estará constantemente actualizada nuestra posición en el mapa. En el ***Anexo 1*** se incluye el código de esta parte del programa. Además de esto, nuestro programa incorpora en la parte de debajo, una barra con una serie de botones. En principio pueden estar en blanco, pero estos botones sirven para que, una vez empezamos a movernos por el parque, aparezcan los animales que se encuentran **a menos de 10 metros de nosotros**. Además de la ventaja que nos produce no tener que buscarlos, ya que iran apareciendo

solos, tenemos la oportunidad de pulsar sobre su imagen y acceder a la información del animal. Adjuntamos unas capturas de pantalla para mostrar el proceso:

Figura 17: Aparece en animales cercanos el flamenco al acercarnos a él.

Figura 18: Aparece el Tántalo al acercarnos a él.

4.2 Servidor

Como ya hemos comentado, nuestra aplicación dispondrá de un ordenador que hará las veces de servidor, y ofrecerá a la aplicación los datos que consultemos de la base de datos en formato XML, lo que nos proporciona muchas ventajas a la hora de, por ejemplo, reutilizar este servidor.

Para el desarrollo de la aplicación, hemos utilizado un ordenador en local que hace las veces de servidor. ¿Plantea esto algún problema para el uso de nuestra aplicación? Ninguno. De hecho, junto con esta memoria se adjuntan unas instrucciones rápidas y unos archivos ya preparados para poder montar un servidor en cualquier parte. ¿Por qué no se ha montado en web? Básicamente, debido a su coste. Un servidor web que tenga soporte para Microsoft SQL Server 2005 y que soporte la ejecución de código ASP es difícil de encontrar de forma gratuita. Durante la realización del proyecto encontramos uno, pero la facilidad de uso era pésima, las cadenas de conexión fallaban, etc... Y se decidió hacer sobre una máquina personal que no estuviera conectada a Internet. Aún así, todo está enfocado a que, en cualquier momento, se pueda montar un servidor de forma facilísima (todos los archivos con backups y archivos de restauración están incluidos en el proyecto) y el programa seguiría funcionando sin ningún problema.

Como ya hemos dicho, la base de datos está montada sobre Microsoft SQL Server 2005, que posteriormente entraremos en detalle sobre la implementación y el diseño. Aparte de esto, se ha utilizado un pequeño programa, muy ligero, que hace que nuestro PC pueda trabajar en local tal como funcionaria un servidor, y así permitirnos ejecutar nuestro script ASP necesario para el funcionamiento de la aplicación. Este programa se llama Baby Web Server. Personalmente, fue un programa que me sorprendió muy gratamente, ya que a pesar de lo ligero que es, ofrece mucha funcionalidad y es muy sencillo de utilizar. Aquí podéis ver una captura de pantalla del programa en cuestión en funcionamiento:

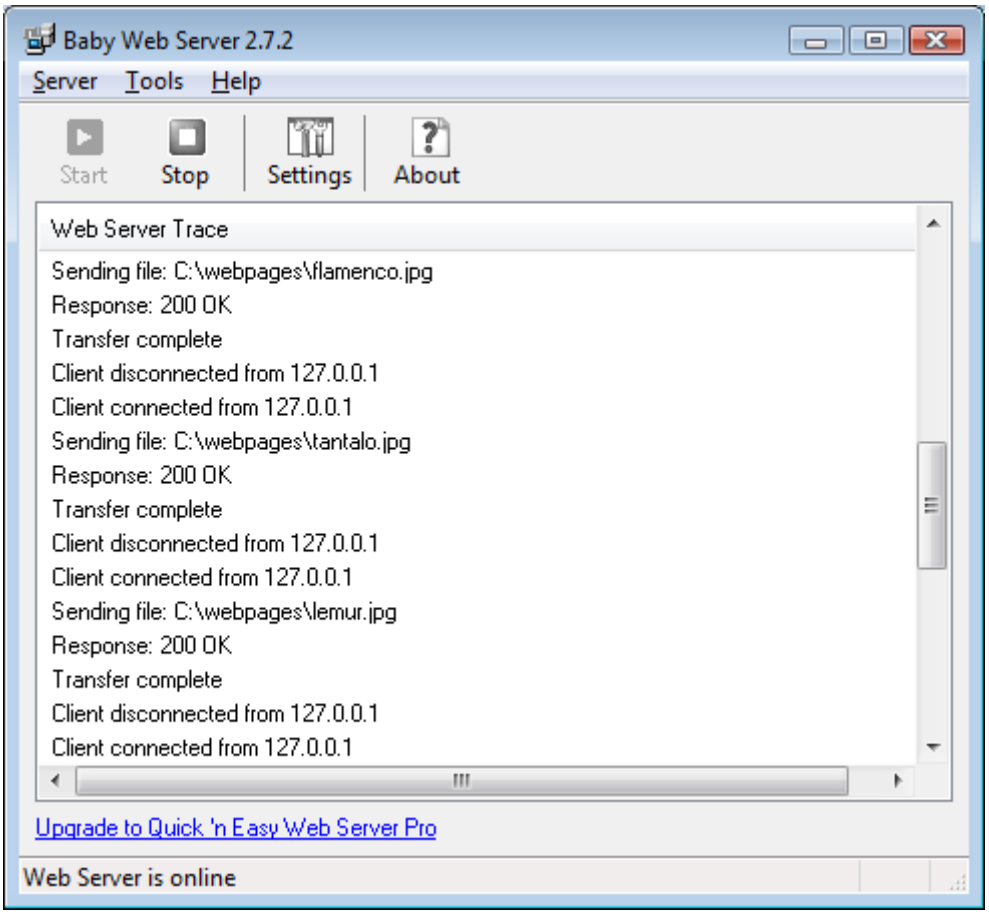

Figura 19: Baby Web Server funcionando.

Las características que nos ofrece el programa en su totalidad son las siguientes:

- Multihilo.
- Log del servidor en tiempo real.
- Configurar el directorio de páginas web (la misma para todas las conexiones).
- Configurar una página HTML predeterminada.
- Soporte para GET, POST, y HEAD
- Soporte nativo de ASP (colecciones Request, Response, Server, QueryString y Form, sesiones, etc.)
- Soporte de cookies.
- SSI (Server Side Includes)

- Estadísticas: Conexiones totales, las solicitudes con y sin éxito, entre otros
- Límite: 5 conexiones simultáneas.

Personalmente, desde aquí felicitar a sus desarrolladores y dar las gracias ya que su facilidad de uso ha facilitado mucho la labor de desarrollo de este proyecto.

4.2.1 Diseño e implementación de la Base de Datos.

Para el diseño de la base de datos que posteriormente hemos implementado en Microsoft SQL Server 2005 se ha desarrollado el siguiente diagrama en el que se muestra su estructura:

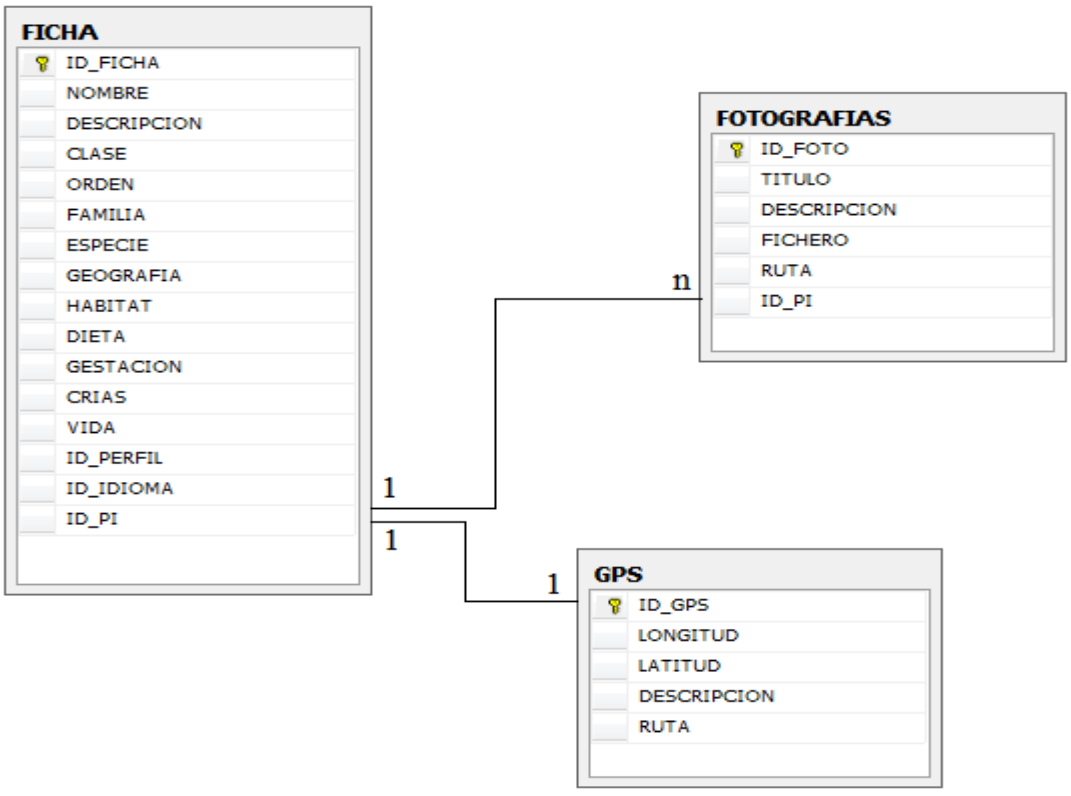

Figura 20: Base de datos utilizada en nuestro programa.

Este diagrama, aunque simple, recoge la funcionalidad necesaria para nuestra aplicación. Tenemos tres tablas, una que es la ficha del animal, otra con su posición GPS i otra con Fotografías de los animales. Las relaciones entre tablas son las siguientes. Partimos de la ficha del animal, que será única para cada uno de ellos. Esta ficha tendrá un identificador, y por tanto, cada foto de este animal contendrá un campo ID_PI con el identificador de la ficha del animal. A su vez, este identificador servirá para saber su posición GPS en el mapa que, como es lógico, sólo habrá una posición por animal, con el mismo identificador.

Además, la ficha contendrá dos identificadores mas, uno para el idioma y otro para el perfil que seleccione el usuario (Estudiante, Adulto, etc.), que serán los que se comprobarán al cargar los datos dependiendo del perfil e idioma que haya seleccionado el usuario.

Para implementar esta Base de Datos, hemos utilizado el programa Microsoft SQL Server 2005, que nos ofrece la suficiente funcionalidad para gestionar todas las tablas que necesitamos. Una vez arrancado SQL Server 2005, procederemos a “Crear una nueva Base de Datos” y le iremos añadiendo las diferentes tablas que hemos mencionado anteriormente. Aquí una captura de pantalla que muestra la base de datos ya creada.

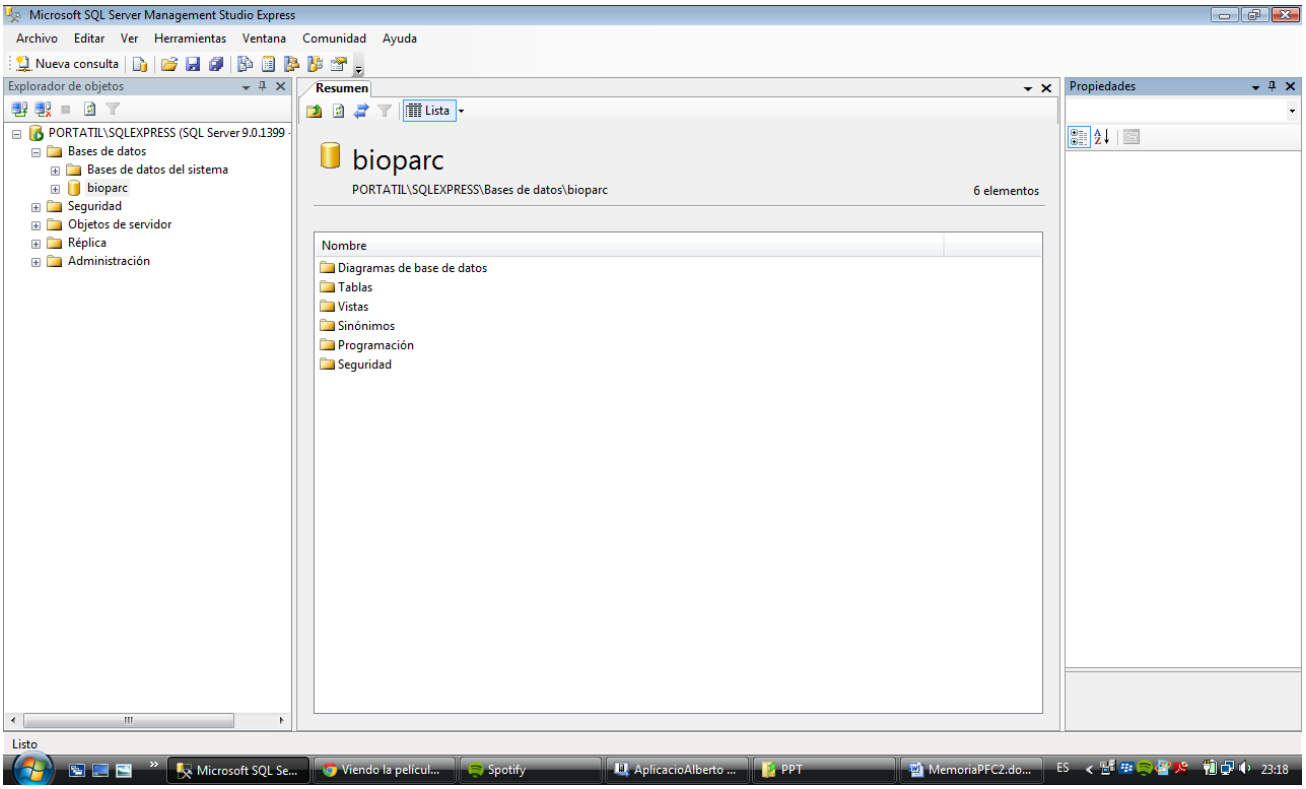

Figura 21: Base de datos creada con Microsoft SQL Server 2005

Una vez hecho esto, Microsoft SQL Server 2005 nos proporciona dos formas de salvaguardar nuestra base de datos:

1.- Creación de un script para generar la base de datos con sus diferentes tablas. Este script se incluye en el ***ANEXO 7*** . El problema de este script es que no genera los datos almacenados en la tabla y, por tanto, solo sirve para recuperar la estructura de toda la base de datos.

2.- "Separar" la base de datos de Microsoft SQL Server 2005. Esta opción consiste basicamente en que Microsoft SQL Server 2005 nos da la opción de obtener un fichero .mdf (Formato Acces) con la base de datos entera, incluida toda la información almacenada en ella. La ventaja de esta opción es que podemos transportar la base de datos de pc a pc, y en concreto en nuestro caso nos sería mas facil subir la base de datos en caso de que queramos establecerla junto con el script ASP en un servidor web accesible desde toda la red.

4.2.2 Desarrollo del Script ASP.

En este apartado hay que agradecer a las personas que trabajaron anteriormente en el mismo tipo de proyecto pero desarrollando para otro tipo de dispositivos como plataformas Android o IPhones. Su enfoque ha hecho posible la reutilización de este script que ellos proporcionaban con unos mínimos retoques, que es al fin y al cabo, la base para obtener los datos de la base de datos en formato XML. En el **ANEXO 6** se incluye el script ASP completo.

Básicamente, lo que hace este script es crear un objeto Connection y asignarle una cadena de conexión y establecer esta con nuestra base de datos.

```
Set Conn = Server.CreateObject("ADODB.Connection")

Conn.Open "Provider=sqloledb;SERVER=ALBERTO1\SQLEXPRESS;DATABASE=bioparc;UID=alberto;PWD=alberto;"
```

Posteriormente, una vez establecida la conexión, lo que hace es crear una consulta de forma dinámica. Nosotros le pasaremos unos datos al archivo ASP y el los procesará y creará la consulta en SQL y obtendrá los datos en formato XML. Aquí tenemos un ejemplo de una consulta sobre la tabla Fichas, que la realizamos directamente desde el navegador y obtenemos los datos. Posteriormente, en el apartado Comunicación Cliente-Servidor explicaremos la forma en que, una vez obtenidos estos datos, el programa los puede procesar.

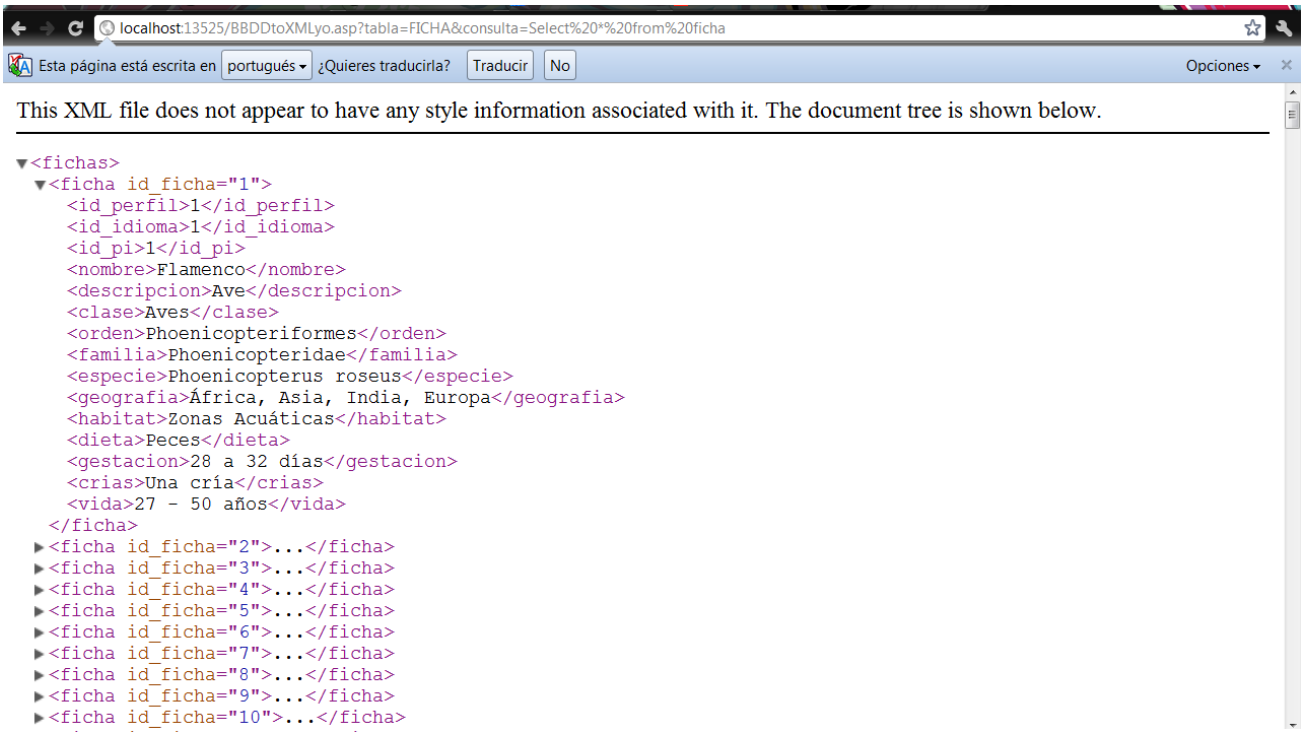

Figura 22: Consulta de la tabla Ficha mediante el script ASP

4.3 Comunicación Cliente-Servidor.

Durante toda la memoria nos hemos centrado en los elementos que tiene nuestro sistema. Básicamente se pueden resumir en dos. Una aplicación cliente, desarrollada sobre un dispositivo Windows Phone 7, que será la que utilice los datos de la parte servidora. Esta parte constará de dos elementos fundamentales: una base de datos desarrollada sobre **Microsoft SQL Server 2005**, y un script **ASP** que será el que nos proporciona los datos en formato XML. Ya hemos visto un poco la funcionalidad de nuestra aplicación cliente y sabemos cómo esta implementada la parte servidor. Pero, ¿cómo se comunican? Es decir, ¿cómo es capaz el cliente de obtener los datos desde el servidor, procesarlos, y mostrárnoslos por pantalla? En esta parte nos centraremos en explicar este proceso.

Cuando accedemos a alguna parte del programa en el que el cliente necesita obtener datos del servidor, lo primero que hace es obtener estos datos y posteriormente ya trabaja con ellos. Evidentemente no comentaremos todas las veces que obtiene datos, pero si las mas representativas. Presentaremos el caso de las fichas y el caso del mapa, para obtener los puntos GPS de los animales. Con estos dos ejemplos se puede extrapolar la explicación a cualquier parte del programa, ya que la forma de obtener los datos es la misma en todas las partes de este.

Vamos a comentar la parte del cliente en la que obtenemos las fichas de los animales. Como bien comentamos en el funcionamiento del programa, accedíamos al botón de fichas y ahí salían todos los nombres de todos los animales. Vamos a ver como el programa obtiene esta lista de nombres. El programa, desde el método "llegitXMLFiches()" crea una cadena para obtener los datos en formato XML de las fichas. En concreto esta cadena es "http://localhost:13525/BBDDtoXMLyo.asp?tabla=FICHA&consulta=Select%20*%20from%20ficha";

Una vez creada la cadena, creamos un WebClient, que será el que nos permita obtener datos de un servidor y trabajar con ellos. A este cliente, le asignamos un manejador para el evento "DownloadStringCompleted" que se dispara cuando está obteniendo datos del servidor. Este manejador le hemos llamado "DownloadComplete" y será el encargado de leer cada línea del XML y procesar los datos como toque.

Vamos a adentrarnos en este método "DownloadComplete". En el tendremos un elemento StringReader llamado stream:

```
StringReader stream = new StringReader(e.Result);
```

este será el encargado de obtener el resultado de la petición al Servidor. Además, tendremos también el elemento XmlReader:

```
XmlReader reader = XmlReader.Create(stream);
```

Este será el encargado de iterar por el documento obtenido e ir obteniendo los datos. Cada dato del documento tiene un "Name" que sería el nombre de la etiqueta XML y un "Value" que seria el valor que acompaña a esa etiqueta. Por tanto, lo que haremos simplemente será iterar por el documento

hasta que encontremos un “Name” que sea el que buscamos, y obtendremos su “Value”, para guardarlo en memoria. En concreto en este caso lo que hacemos es obtener todos los datos de la ficha, y guardarlos en un elemento ficha que posteriormente meteremos en una List<> de fichas.

Finalmente, una vez completado todo este proceso y con los datos ya en la lista, ejecutamos el método “mostrarEnLista()” que lo que hace simplemente es, recorrer la lista que tenemos de fichas, obtener el nombre de cada una, y mostrarlos por pantalla para que el usuario pueda elegir un animal de todos los disponibles. En el **ANEXO 2** se exponen los tres métodos y la descripción de XmlReader y sus principales métodos para recorrer un documento en formato XML.

La principal ventaja de usar XML como lenguaje para obtener datos es que es independiente de la plataforma. En nuestro caso es cuando más claro se ve. El script se utilizaba para trabajar con Android, pero como XML no depende de ningun sistema ni plataforma, se ha podido usar el mismo script para obtener los datos de igual forma. Normalmente, como es nuestro caso, la mayoria de los lenguajes proporcionan mecanismos para recorrer ficheros en formato XML y procesar sus datos, guardarlos, etc...

Bien, una vez obtenida esta lista, el usuario tendrá disponibles todos los nombres de los animales en pantalla para poder elegir cualquiera de ellos, tal como hemos visto anteriormente, En el momento que el usuario elija un animal, el programa pasará el identificador de la ficha y la otra página mostrará la información. En el **ANEXO 3** se muestra esta forma de pasar información entre una página del programa y otra.

Lo mismo sucede con la parte que trabaja con el posicionamiento GPS, pero no la trataremos en este punto, sino en el siguiente, pues es un tema más extenso que simplemente obtener información.

4.4 Sistema de Geo-Posicionamiento y avisos de animales

Nos centraremos ahora en la parte de Geo-Posicionamiento y aviso de los animales. Tenemos un mapa, basado en Google Maps, que nos ofrece una vista de satélite del parque. En el mapa tenemos un indicador, un elemento que nos proporciona la API de Windows Phone 7, llamado PushPin, que nos indica la posición en la que nos encontramos.

Cuando accedemos al mapa, el sistema obtiene nuestra posición mediante un elemento GeoCoordinateWatcher que accede a la información GPS del dispositivo móvil. Es muy importante destacar que si el GPS no está conectado, el dispositivo intentará detectar la posición vía wifi, y en el peor de los casos vía red móvil.

Una vez obtenida la posición, el PushPin se colocará dentro del mapa en la posición que hayamos obtenido. Este pushpin se actualizará, debido a que el elemento GeoCoordinateWatcher cada vez que la posición varia, lanza el evento PositionChanged, y dentro de este evento, entre otras cosas, se realiza esta función, la de actualizar la posición del PushPin. En el **ANEXO 4** se incluye la función de inicialización del PushPin, y en el **ANEXO 1** se incluye todo el evento PositionChanged.

Una vez hecho esto, el sistema intenta obtener de la base de datos una lista con los puntos GPS donde están todos los animales del parque. Estos puntos GPS tienen entre otros, 3 elementos principales: latitud, longitud e identificador. Gracias al identificador podemos saber de qué animal se trata. Y gracias a la latitud y la longitud podemos saber exactamente donde está situado el animal. ¿Cómo obtenemos estos puntos de la base de datos? Igual que obteníamos las Fichas, o las fotos…simplemente creamos una cadena con la consulta que la realizaremos sobre el script ASP, y este obtendrá los datos en formato XML que interpretaremos, y meteremos en una lista de puntos GPS. Así, tendremos una lista con todos los puntos del mapa en los que hay animales. En el **ANEXO 5** se muestra el trozo de código con el que obtenemos los puntos GPS.

Ya tenemos la lista con los puntos GPS y la forma de mantener actualizado el Pushpin. Y ahora, ¿Cómo hacemos que nos alerte de la proximidad de algún animal? El método que se encargará de

mantener todo esto actualizado es el método PositionChanged. Cada vez que este evento se dispare, el sistema realizará un recorrido de la lista de puntos GPS que hemos obtenido de la BD. Por cada punto GPS, el sistema comprobará la distancia de ese punto con nuestra posición actual, de forma que si la distancia que nos separa es menor de 10 metros, se mostrará la foto del animal en la parte de debajo de la pantalla.

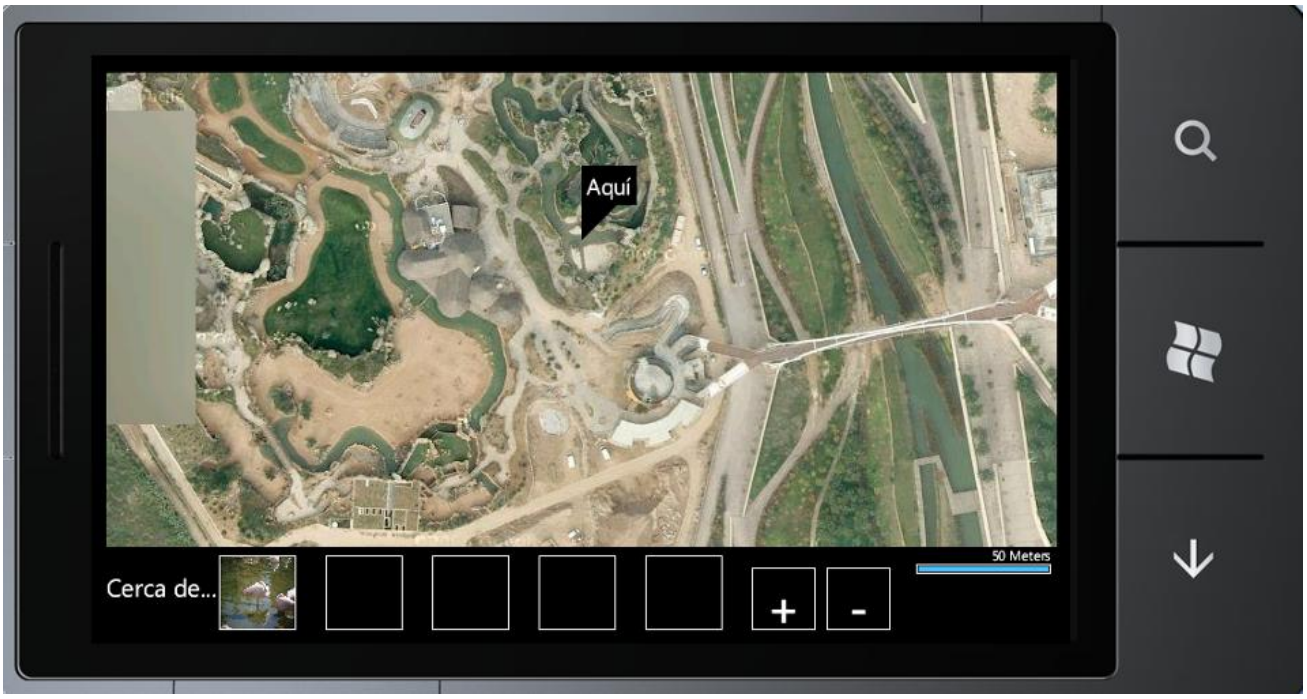

Figura 23: El mapa en funcionamiento, con el pushpin y la foto del animal cercano

Esta parte de la funcionalidad la encontramos como código en el**,** donde podemos encontrar todo el código que hace posible esto. Como ya hemos visto, básicamente, se trata de incluir el código en el evento PositionChanged, para que en cada movimiento, se actualice tanto la posición del PushPin como las imágenes que aparecen en la parte de debajo, es decir, los animales cercanos.

Si pulsamos en la foto de alguno de los animales cercanos, aparece la ficha de dicho animal con sus datos. Y si pulsamos luego el botón de atrás, volvemos otra vez al mapa, para poder seguir explorando y viendo características de más animales.

5 Conclusiones.

El desarrollo de software sobre dispositivos móviles es, actualmente, uno de los campos más importantes en los que se va a centrar la informática. Estos dispositivos, cada día mas potentes, nos ofrecen la ventaja de no depender de un punto fijo, sino que podemos movernos donde y cuando queramos, y estar constantemente trabajando con estos dispositivos. La evolución de la tecnología en el campo de las redes ha sido también otro factor importante.

Hoy en día disponemos, como hemos comentado al principio de multitud de sistemas y dispositivos móviles, internamente muy diferentes entre sí, externamente no tanto y, que al final, nos acaban ofreciendo una funcionalidad parecida. Además, las principales compañías que dominan el campo de los dispositivos móviles, Microsoft, Apple y Google, son también grandes compañías a nivel de ordenadores de sobremesa y demás dispositivos. Es de esperar que estas compañías dediquen cada vez más tiempo y esfuerzo al desarrollo y evolución de estos dispositivos, ya que claramente podemos observar que la movilidad que nos ofrecen y el "estar siempre conectado" van a ser factores muy importantes para la sociedad que viene en unos años, e incluso ya lo empieza a ser para la generación actual.

Es por esto, que sistemas rudimentarios, como el uso de un mapa para guiarnos en cierto lugar (como es el caso del proyecto que nos ocupa) van a ir siendo reemplazados, poco a poco, por dispositivos móviles con una funcionalidad que sobrepasará en lo imaginable a estos dispositivos más rudimentarios.

El proyecto que nos ocupa, un sistema de guiado para el Bioparc de Valencia, es un proyecto ambicioso que desarrollamos sobre Windows Phone 7, y que ya se ha desarrollado en otros sistemas operativos como Android o MacOS. Este proyecto pondrá al alcance de todos los visitantes del parque, una funcionalidad que no incluía el mapa clásico: la posibilidad de saber en todo momento nuestra posición exacta, la posibilidad de ver que animales tengo cerca, la posibilidad de conocer más sobre estos animales con un solo toque de dedo en la pantalla de nuestro dispositivo. Y todo esto

desarrollado sobre un sistema puntero que es, como ya hemos comentado, Windows Phone 7, que será uno de los más potentes en el mercado de aquí a unos años.

Personalmente, creo que la tecnología móvil va a ser la tecnología que reine dentro de unos años, y no solo en el campo de la informática, sino en todos los ámbitos de nuestra vida. La posibilidad de tener toda, o casi toda, la funcionalidad que nos brinda un ordenador de sobremesa, comprimida en el tamaño de un dispositivo que no excede el tamaño de la palma de nuestra mano, y que podemos llevar donde queramos y usar donde queramos. Eso si que es realmente el futuro. La tecnología precisa, en el momento y en el lugar adecuado.

ANEXO 1: Posicionamiento en el mapa. Inicialización del GPS y captura del movimiento.

```
public void inicializarGPS()
      {
          miGPS = new GeoCoordinateWatcher(GeoPositionAccuracy.High);
          miGPS.Start();

          //quan reba la posicio li posem el manejaor

     miGPS.PositionChanged += new System.EventHandler<GeoPositionChangedEventArgs
                              <GeoCoordinate>>(miGPS_PositionChanged);
      }

    void miGPS_PositionChanged(object sender, GeoPositionChangedEventArgs<GeoCoordinate> e)
      {
          googlemap.Children.Remove(pushpin);
          GeoCoordinate posicion = (GeoCoordinate)sender;

          pushpin.Location = posicion;
          pushpin.Style = (Style)(Application.Current.Resources["PushpinStyle"]);
          pushpin.Content = "Aquí";
          googlemap.Children.Add(pushpin);
      }
```

ANEXO 2: XML Reader. Descripción y métodos principales.

XmlReader proporciona acceso de solo lectura y con desplazamiento sólo hacia delante a un flujo de datos XML. La clase XmlReader se ajusta al lenguaje de marcado extensible 1.0 (XML) y a los espacios de nombres de las recomendaciones de XML del Consorcio W3C.

El nodo actual hace referencia al nodo en el que está situado el lector. Para hacer avanzar el lector, utilice cualquiera de los métodos de lectura; las propiedades reflejarán el valor del nodo actual.

XmlReader produce una excepción XmlException en los errores de análisis de XML. Tras producirse una excepción, el estado del lector es imprevisible. Por ejemplo, el tipo de nodo notificado puede ser distinto del tipo de nodo real del nodo actual. Utilice la propiedad ReadState para comprobar si el lector está en estado de error.

Métodos.

Create(String)	Crea una nueva instancia de XmlReader con el URI especificado.
MoveToFirstAttribute	Cuando se invalida en una clase derivada, se desplaza hasta el primer atributo.
MoveToNextAttribute	Cuando se invalida en una clase derivada, se desplaza hasta el siguiente atributo.
Read	Cuando se invalida en una clase derivada, lee el siguiente nodo de la secuencia.
ReadAttributeValue	Cuando se invalida en una clase derivada, analiza el valor de atributo en uno o varios nodos Text, EntityReference o EndEntity.

Atributos.

Name	Cuando se invalida en una clase derivada, obtiene el nombre completo del nodo actual.
Value	Cuando se invalida en una clase derivada, obtiene el valor de texto del nodo actual.

ANEXO 3: Forma de pasar información entre una página y otra.

Aquí se incluye el código que utilizamos para pasar la información del animal que seleccionamos a la página que muestra información de ese animal. En concreto, se pasa el identificador del animal.

```
public void muestraInfo(Ficha f)
{
         this.NavigationService.Navigate(new Uri("/InformacionAnimal.xaml?text="+f.id_ficha,
UriKind.Relative));

}
```

ANEXO 4: Inicialización del PushPin, indicador de posición en el mapa.

Aquí se incluye el código que utilizamos para inicializar la variable PushPin, que es el elemento que nos indica en cada momento en que posición del mapa nos encontramos.

```
public void crearPushpin()
        {

            pushpin.Location = googlemap.Center;
            pushpin.Style = (Style)(Application.Current.Resources["PushpinStyle"]);
            pushpin.Content = "Aquí";
            googlemap.Children.Add(pushpin);

        }
```

ANEXO 5: Obtención de la tabla con los datos GPS de los animales.

```
void llegirXMLPuntsGps()
      {
         puntosGPS = new List<PuntoGPS>();
         String urlString =
"http://localhost:13525/BBDDtoXMLyo.asp?tabla=GPS&consulta=Select%20*%20from%20gps";
         WebClient client = new WebClient();
         client.Encoding = System.Text.Encoding.GetEncoding("ISO-8859-1");


         client.DownloadStringCompleted += DownloadComplete;
         client.DownloadStringAsync(new Uri(urlString));

      }

      private void DownloadComplete(object sender, DownloadStringCompletedEventArgs e)//LECTOR XML TABLA
GPS
      {
         var client = sender as WebClient;
         if (client != null)
         {
            client.DownloadStringCompleted -= DownloadComplete;
         }
         if (e.Error != null)
         {
            throw e.Error;
         }
         if (e.Result != null)
         {
            //e.Result contains a String with the downloaded data.

            StringReader stream = new StringReader(e.Result);

            XmlReader reader = XmlReader.Create(stream);

            int id_gps = 0;
            double latitud;
            double longitud;
            string descripcion;
            string ruta;

            while (reader.Read())
            {
               if (reader.NodeType == XmlNodeType.Element)
               {
                  if (reader.Name == "gps")
                  {

                     if (true == reader.MoveToFirstAttribute())
                     {
                        id_gps = Convert.ToInt32(reader.Value);
                     }
```

```
                reader.Read();
                reader.Read();

                longitud = Convert.ToDouble(reader.Value);

                reader.Read();
                reader.Read();
                reader.Read();

                latitud = Convert.ToDouble(reader.Value);

                reader.Read();
                reader.Read();
                reader.Read();

                descripcion = reader.Value;

                reader.Read();
                reader.Read();
                reader.Read();

                ruta = reader.Value;

                PuntoGPS puntogps = new PuntoGPS(id_gps, latitud, longitud, descripcion,ruta);

                puntosGPS.Add(puntogps);

            }

        }
      }//fi while

      // mostrarEnLlista();
   }
}
```

ANEXO 6: Script ASP, para obtener datos de la base de datos.

```
<% @ LANGUAGE="VBSCRIPT" %>

<%
Dim xmlDoc
Dim tabla, consulta
tabla = request.Querystring("tabla")
consulta = request.Querystring("consulta")

Response.ContentType = "text/xml"

'Antes de nada hay que instanciar el objeto Connection
Set Conn = Server.CreateObject("ADODB.Connection")

'Una vez instanciado Connection lo podemos abrir y le asignamos la base de datos donde vamos a efectuar las operaciones
Conn.Open
"Provider=sqloledb;SERVER=sql207.zobyhost.com;DATABASE=zoby_7673719_bioparcbd;UID=zoby_7673719;PWD=04f2d70;"

'Ahora creamos la sentencia SQL que nos servira para hablar a la BD
'Funciona el select con los where (testeado)
sSQL = consulta
'Ejecutamos la orden
set RS=createobject("ADODB.Recordset")
RS.open sSQL,Conn
'Mostramos los registros%>

<%
comillas = Chr(34)


'Response.write("<?xml version='1.0' encoding='UTF-8' ?>")
'Acepta caracteres especiales, ñ y acentos
Response.write("<?xml version=" & comillas)
Response.write("1.0" & comillas)
Response.write(" encoding=" &comillas)
'Response.write("UTF-8" &comillas)
Response.write("ISO-8859-1" &comillas)
Response.write(" ?>")

if tabla="PUNTOS_DE_INTERES" Then
Response.write("<puntos_de_interes>")
Do While Not RS.Eof
'    comillas = Chr(34)

    Response.write("<punto_de_interes id_pi=" & comillas)
    Response.write RS("ID_PI")
    Response.write(comillas & ">")
    Response.write("<id_zona>")

    Response.write RS("ID_ZONA")
    Response.write("</id_zona>")
    Response.write("<id_foto>")

    Response.write RS("ID_FOTO")
    Response.write("</id_foto>")

    Response.write("<id_tipopi>")
```

```
    Response.write RS("ID_TIPOPI")
    Response.write("</id_tipopi>")

    Response.write("<descripcion>")

    Response.write RS("DESCRIPCION")
    Response.write("</descripcion>")
    Response.write("</punto_de_interes>")


RS.MoveNext

Loop
RS.close

Response.write("</puntos_de_interes>")
'Cerramos el sistema de conexion

ElseIf tabla="FOTOGRAFIAS" Then

Response.write("<fotos>")
Do While Not RS.Eof
   comillas = Chr(34)

    Response.write("<foto id_foto=" & comillas)
    Response.write RS("ID_FOTO")
    Response.write(comillas & ">")
    Response.write("<id_pi>")

    Response.write RS("ID_PI")
    Response.write("</id_pi>")

    Response.write("<titulo>")

    Response.write RS("TITULO")
    Response.write("</titulo>")

    Response.write("<descripcion>")

    Response.write RS("DESCRIPCION")
    Response.write("</descripcion>")

    Response.write("<fichero>")

    Response.write RS("FICHERO")
    Response.write("</fichero>")
    Response.write("<ruta>")

    Response.write RS("RUTA")
    Response.write("</ruta>")

    Response.write("</foto>")

RS.MoveNext

Loop
RS.close

Response.write("</fotos>")
```

```
'Cerramos el sistema de conexion

ElseIf tabla="SERVICIOS_PARQUE" Then

Response.write("<servicios_parque>")
Do While Not RS.Eof
   comillas = Chr(34)

    Response.write("<servicio_parque id_sp=" & comillas)
    Response.write RS("ID_SP")
    Response.write(comillas & ">")
    Response.write("<id_gps>")

    Response.write RS("ID_GPS")
    Response.write("</id_gps>")

    Response.write("<descripcion>")

    Response.write RS("DESCRIPCION")
    Response.write("</descripcion>")

    Response.write("</servicio_parque>")

RS.MoveNext

Loop
RS.close

Response.write("</servicios_parque>")
'Cerramos el sistema de conexion

ElseIf tabla="GPS" Then

Response.write("<gpses>")
Do While Not RS.Eof
   comillas = Chr(34)

    Response.write("<gps id_gps=" & comillas)
    Response.write RS("ID_GPS")
    Response.write(comillas & ">")
    Response.write("<longitud>")

    Response.write RS("LONGITUD")
    Response.write("</longitud>")
    Response.write("<latitud>")

    Response.write RS("LATITUD")
    Response.write("</latitud>")
    Response.write("<descripcion>")

    Response.write RS("DESCRIPCION")
    Response.write("</descripcion>")

    Response.write("</gps>")

RS.MoveNext

Loop
RS.close
```

```
Response.write("</gpses>")
'Cerramos el sistema de conexion


ElseIf tabla="TIPO_PUNTO_INTERES" Then

Response.write("<tipos_punto_interes>")
Do While Not RS.Eof
   comillas = Chr(34)

    Response.write("<tipo_punto_interes id_tipopi=" & comillas)
    Response.write RS("ID_TIPOPI")
    Response.write(comillas & ">")
    Response.write("<descripcion>")

    Response.write RS("DESCRIPCION")
    Response.write("</descripcion>")

    Response.write("</tipo_punto_interes>")

RS.MoveNext

Loop
RS.close

Response.write("</tipos_punto_interes>")
'Cerramos el sistema de conexion

ElseIf tabla="PUNTOS_INTERES_GPS" Then

Response.write("<puntos_interes_gps>")
Do While Not RS.Eof
   comillas = Chr(34)

    Response.write("<punto_interes_gps id_pgps=" & comillas)
    Response.write RS("ID_PGPS")
    Response.write(comillas & ">")
    Response.write("<id_pi>")

    Response.write RS("ID_PI")
    Response.write("</id_pi>")
    Response.write("<id_gps>")

    Response.write RS("ID_GPS")
    Response.write("</id_gps>")

    Response.write("</punto_interes_gps>")

RS.MoveNext

Loop
RS.close

Response.write("</puntos_interes_gps>")
'Cerramos el sistema de conexion

ElseIf tabla="ZONAS" Then

Response.write("<zonas>")
Do While Not RS.Eof
```

```
   comillas = Chr(34)

    Response.write("<zona id_zona=" & comillas)
    Response.write RS("ID_ZONA")
    Response.write(comillas & ">")
    Response.write("<descripcion>")

    Response.write RS("DESCRIPCION")
    Response.write("</descripcion>")
    Response.write("</zona>")

RS.MoveNext

Loop
RS.close

Response.write("</zonas>")
'Cerramos el sistema de conexion

ElseIf tabla="TIPO_AUDIO" Then

Response.write("<tipos_audio>")
Do While Not RS.Eof
   comillas = Chr(34)

    Response.write("<tipo_audio id_tipoa=" & comillas)
    Response.write RS("ID_TIPOA")
    Response.write(comillas & ">")
    Response.write("<descripcion>")

    Response.write RS("DESCRIPCION")
    Response.write("</descripcion>")
    Response.write("</tipo_audio>")

RS.MoveNext

Loop
RS.close

Response.write("</tipos_audio>")
'Cerramos el sistema de conexion



ElseIf tabla="CURIOSIDADES" Then

Response.write("<curiosidades>")
Do While Not RS.Eof
   comillas = Chr(34)

    Response.write("<curiosidad id_curiosidad=" & comillas)
    Response.write RS("ID_CURIOSIDAD")
    Response.write(comillas & ">")
    Response.write("<id_pi>")

    Response.write RS("ID_PI")
    Response.write("</id_pi>")
    Response.write("<id_idioma>")

    Response.write RS("ID_IDIOMA")
```

```
    Response.write("</id_idioma>")
    Response.write("<id_perfil>")

    Response.write RS("ID_PERFIL")
    Response.write("</id_perfil>")
    Response.write("<descripcion>")

    Response.write RS("DESCRIPCION")
    Response.write("</descripcion>")

    Response.write("</curiosidad>")

RS.MoveNext

Loop
RS.close

Response.write("</curiosidades>")
'Cerramos el sistema de conexion

ElseIf tabla="AUDIO" Then

Response.write("<audios>")
Do While Not RS.Eof
   comillas = Chr(34)

    Response.write("<audio id_audio=" & comillas)
    Response.write RS("ID_AUDIO")
    Response.write(comillas & ">")
    Response.write("<id_perfil>")

    Response.write RS("ID_PERFIL")
    Response.write("</id_perfil>")
    Response.write("<id_idioma>")

    Response.write RS("ID_IDIOMA")
    Response.write("</id_idioma>")
    Response.write("<id_tipoa>")

    Response.write RS("ID_TIPOA")
    Response.write("</id_tipoa>")
    Response.write("<id_pi>")

    Response.write RS("ID_PI")
    Response.write("</id_pi>")
    Response.write("<titulo>")

    Response.write RS("TITULO")
    Response.write("</titulo>")
    Response.write("<descripcion>")

    Response.write RS("DESCRIPCION")
    Response.write("</descripcion>")
    Response.write("<fichero>")

    Response.write RS("FICHERO")
    Response.write("</fichero>")
    Response.write("<ruta>")

    Response.write RS("RUTA")
```

```
    Response.write("</ruta>")

    Response.write("</audio>")

RS.MoveNext

Loop
RS.close

Response.write("</audios>")
'Cerramos el sistema de conexion


ElseIf tabla="TIPO_BIOLOGIA" Then

Response.write("<tipos_biologia>")
Do While Not RS.Eof
   comillas = Chr(34)

    Response.write("<tipo_biologia id_tipob=" & comillas)
    Response.write RS("ID_TIPOB")
    Response.write(comillas & ">")
    Response.write("<descripcion>")

    Response.write RS("DESCRIPCION")
    Response.write("</descripcion>")
    Response.write("</tipo_biologia>")

RS.MoveNext

Loop
RS.close

Response.write("</tipos_biologia>")
'Cerramos el sistema de conexion

ElseIf tabla="BIOLOGIA" Then

Response.write("<biologias>")
Do While Not RS.Eof
   comillas = Chr(34)

    Response.write("<biologia id_biologia=" & comillas)
    Response.write RS("ID_BIOLOGIA")
    Response.write(comillas & ">")
    Response.write("<id_pi>")

    Response.write RS("ID_PI")
    Response.write("</id_pi>")
    Response.write("<id_idioma>")

    Response.write RS("ID_IDIOMA")
    Response.write("</id_idioma>")
    Response.write("<id_perfil>")

    Response.write RS("ID_PERFIL")
    Response.write("</id_perfil>")
    Response.write("<id_tipob>")

    Response.write RS("ID_TIPOB")
```

```
    Response.write("</id_tipob>")
    Response.write("<descripcion>")

    Response.write RS("DESCRIPCION")
    Response.write("</descripcion>")

    Response.write("</biologia>")

RS.MoveNext

Loop
RS.close

Response.write("</biologias>")
'Cerramos el sistema de conexion

ElseIf tabla="IDIOMA" Then

Response.write("<idiomas>")
Do While Not RS.Eof
   comillas = Chr(34)

    Response.write("<idioma id_idioma=" & comillas)
    Response.write RS("ID_IDIOMA")
    Response.write(comillas & ">")
    Response.write("<descripcion>")

    Response.write RS("DESCRIPCION")
    Response.write("</descripcion>")

    Response.write("</idioma>")

RS.MoveNext

Loop
RS.close

Response.write("</idiomas>")
'Cerramos el sistema de conexion

ElseIf tabla="FICHA" Then

Response.write("<fichas>")
Do While Not RS.Eof
   comillas = Chr(34)

    Response.write("<ficha id_ficha=" & comillas)
    Response.write RS("ID_FICHA")
    Response.write(comillas & ">")
    Response.write("<id_perfil>")

    Response.write RS("ID_PERFIL")
    Response.write("</id_perfil>")

    Response.write("<id_idioma>")

    Response.write RS("ID_IDIOMA")
    Response.write("</id_idioma>")
    Response.write("<id_pi>")
```

```
    Response.write RS("ID_PI")
    Response.write("</id_pi>")
    Response.write("<nombre>")

    Response.write RS("NOMBRE")
     Response.write("</nombre>")
    Response.write("<descripcion>")

    Response.write RS("DESCRIPCION")
    Response.write("</descripcion>")
    Response.write("<clase>")

    Response.write RS("CLASE")
    Response.write("</clase>")
    Response.write("<orden>")

    Response.write RS("ORDEN")
    Response.write("</orden>")
    Response.write("<familia>")

    Response.write RS("FAMILIA")
    Response.write("</familia>")
    Response.write("<especie>")

    Response.write RS("ESPECIE")
    Response.write("</especie>")
    Response.write("<geografia>")

    Response.write RS("GEOGRAFIA")
    Response.write("</geografia>")
    Response.write("<habitat>")

    Response.write RS("HABITAT")
    Response.write("</habitat>")
    Response.write("<dieta>")

    Response.write RS("DIETA")
    Response.write("</dieta>")
    Response.write("<gestacion>")

    Response.write RS("GESTACION")
    Response.write("</gestacion>")
    Response.write("<crias>")

    Response.write RS("CRIAS")
    Response.write("</crias>")
    Response.write("<vida>")

    Response.write RS("VIDA")
    Response.write("</vida>")

    Response.write("</ficha>")

RS.MoveNext

Loop
RS.close

Response.write("</fichas>")
'Cerramos el sistema de conexion
```

```
ElseIf tabla="PERFIL" Then

Response.write("<perfiles>")
Do While Not RS.Eof
   comillas = Chr(34)

    Response.write("<perfil id_perfil=" & comillas)
    Response.write RS("ID_PERFIL")
    Response.write(comillas & ">")
    Response.write("<descripcion>")

    Response.write RS("DESCRIPCION")
    Response.write("</descripcion>")

    Response.write("</perfil>")

RS.MoveNext

Loop
RS.close

Response.write("</perfiles>")
'Cerramos el sistema de conexion

ElseIf tabla="VIDEO" Then

Response.write("<videos>")
Do While Not RS.Eof
   comillas = Chr(34)

    Response.write("<video id_video=" & comillas)
    Response.write RS("ID_VIDEO")
    Response.write(comillas & ">")
    Response.write("<id_perfil>")

    Response.write RS("ID_PERFIL")
    Response.write("</id_perfil>")

    Response.write("<id_idioma>")

    Response.write RS("ID_IDIOMA")
    Response.write("</id_idioma>")
    Response.write("<id_tipov>")

    Response.write RS("ID_TIPOV")
    Response.write("</id_tipov>")
    Response.write("<id_pi>")

    Response.write RS("ID_PI")
     Response.write("</id_pi>")
    Response.write("<titulo>")

    Response.write RS("TITULO")
    Response.write("</titulo>")
    Response.write("<descripcion>")

    Response.write RS("DESCRIPCION")
    Response.write("</descripcion>")
    Response.write("<fichero>")
```

```
    Response.write RS("FICHERO")
    Response.write("</fichero>")
    Response.write("<ruta>")

    Response.write RS("RUTA")
    Response.write("</ruta>")
    Response.write("</video>")

RS.MoveNext

Loop
RS.close

Response.write("</videos>")
'Cerramos el sistema de conexion

ElseIf tabla="TIPO_VIDEO" Then
Response.write("<tipos_video>")
Do While Not RS.Eof
   comillas = Chr(34)

    Response.write("<tipo_video id_tipov=" & comillas)
    Response.write RS("ID_TIPOV")
    Response.write(comillas & ">")
    Response.write("<descripcion>")

    Response.write RS("DESCRIPCION")
    Response.write("</descripcion>")

    Response.write("</tipo_video>")

RS.MoveNext

Loop
RS.close

Response.write("</tipos_video>")
'Cerramos el sistema de conexion
End If
%>
```

ANEXO 7: Script SQL, para crear la base de datos.

```
CREATE TABLE IF NOT EXISTS `FICHA` (
  `ID_FICHA` int(11) NOT NULL,
  `NOMBRE` varchar(50) collate latin1_spanish_ci default NULL,
  `DESCRIPCION` text collate latin1_spanish_ci,
  `CLASE` varchar(50) collate latin1_spanish_ci default NULL,
  `ORDEN` varchar(50) collate latin1_spanish_ci default NULL,
  `FAMILIA` varchar(50) collate latin1_spanish_ci default NULL,
  `ESPECIE` varchar(50) collate latin1_spanish_ci default NULL,
  `GEOGRAFIA` varchar(50) collate latin1_spanish_ci default NULL,
  `HABITAT` varchar(50) collate latin1_spanish_ci default NULL,
  `DIETA` varchar(50) collate latin1_spanish_ci default NULL,
  `GESTACION` varchar(50) collate latin1_spanish_ci default NULL,
  `CRIAS` varchar(50) collate latin1_spanish_ci default NULL,
  `VIDA` varchar(50) collate latin1_spanish_ci default NULL,
  `ID_PERFIL` int(11) default NULL,
  `ID_IDIOMA` int(11) default NULL,
  `ID_PI` int(11) default NULL,
  PRIMARY KEY  (`ID_FICHA`)
)


CREATE TABLE IF NOT EXISTS `FOTOGRAFIAS` (
  `ID_FOTO` int(11) NOT NULL,
  `TITULO` varchar(50) collate latin1_spanish_ci default NULL,
  `DESCRIPCION` varchar(50) collate latin1_spanish_ci default NULL,
  `FICHERO` varchar(50) collate latin1_spanish_ci default NULL,
  `RUTA` varchar(150) collate latin1_spanish_ci default NULL,
  `ID_PI` int(11) default NULL,
  PRIMARY KEY  (`ID_FOTO`)
)

CREATE TABLE IF NOT EXISTS `GPS` (
  `ID_GPS` int(11) NOT NULL,
  `LONGITUD` float default NULL,
  `LATITUD` float default NULL,
  `DESCRIPCION` varchar(128) collate latin1_spanish_ci default NULL,
RUTA` varchar(128) collate latin1_spanish_ci default NULL,
 PRIMARY KEY  (`ID_GPS`)
)
```

Bibliografía.

- Fling, Brian. *Mobile design and development* . Beijing: O'Reilly, 2009.
- Petzold, Charles. *Programming Windows phone 7* . Microsoft silverlight ed. Redmond, Washington: Microsoft Press, 2010.
- Watson, Karli, and Robert Ensor. *Beginning Windows Phone 7 Application Development Building Windows® Phone Applications Using Silverlight® and XNA®.*. EEUU: Microsoft, 2011.
- Gunderloy, Mike, Joseph L. Jorden, and David W. Tschanz. *La biblia de SQL Server 2005* . Madrid: Anaya Multimedia, 2007.
- Kalani, Amit, and Priti Kalani. *Developing XML Web services and server components with Visual C# .NET and the .NET framework* . Indianapolis, Ind.: Que Certification, 2004.
- Microsoft MSDN Library. *Windows Phone Development.* [http://msdn.microsoft.com/en-us/library/ff402535%28v=VS.92%29.aspx]
- Durán, Luis. *Bases de datos con visual basic* . Barcelona: Marcombo Ediciones Técnicas ;, 2007.
- Liberty, Jesse, Dan Hurwitz, and Dan Maharry. *Programación con ASP.NET 3.5* . Madrid: Anaya Multimedia, 2009.
- Frederick, Gail, and Rajesh Lal. *Smartphone Web Development Building Javascript, CSS, HTML and Ajax-Based Applications for iPhone, Android, Palm Pre, Blackberry, Windows Mobile and Nokia S60.*. New ed. New York: Apress L.P.
- Microsoft MSDN Library. *Programación web con ASP.NET* . [http://msdn.microsoft.com/es-es/asp.net/aa336522]