UNIVERSITAT POLITÈCNICA DE VALÈNCIA

CAMPUS D'ALCOI

Realización de una APP en Windows Phone para seguimiento online de una actividad deportiva.

MEMORIA PRESENTADA POR:

José Vicente Fuster Mañez.

Tutor:

Rubén Pérez Llorens.

Grado de Ingeniería informática

Convocatoria de defensa: Septiembre 2017

Resumen

El proyecto consiste en la realización de una app para el seguimiento de una actividad deportiva online, informando continuamente de la ubicación geográfica.

Al iniciar la aplicación hay que entrar cómo usuario registrado y en el caso de no estar registrado hay que darse de alta. Una vez registrado el usuario accede a la aplicación para iniciar su actividad, realizándose un seguimiento de su ubicación informando de la latitud y longitud de su posición por gps. Estos datos son guardados en una base datos del servidor remoto de Microsoft Azure y si no hay conexión online se guardan en el almacenamiento aislado del teléfono para despues enviarse cuando haya conexión al servidor remoto.

En el desarrollo de la aplicación como no se puede interactuar directamente con un SQL Server desde una aplicación de Windows Phone he creado una capa de servicio alrededor de la base de datos que expone todas las operaciones CRUD que se necesitan.Este servicio se llama desde la aplicación y para ello la tecnología que he utilizado ha sido el servicio WCF.

Para realizar la aplicación he utilizado el lenguaje de programación C#, XML, LINQ to SQL.

Las herramientas utilizadas han sido Visual studio community 2015, Microsoft Azure, Sql server Management Studio y Visio 2016.

Los resultados obtenidos han sido bastante favorables.

Como conclusión después de realizar este trabajo hoy en dia la localización por Gps proporciona una eficacia muy alta y con unas prestaciones y rendimientos bastante óptimos.

Palabras clave:

Windows Phone, Geolocation, localización geográfica, GPS, servidor remoto, XAML, LINQ to SQL, SDK, WCF, Servicio web, Mapa, latitud, longitud, referencia de servicio, interfaz de usuario, autenticación, implementación, emulador, API

Abstract

The project is in the realization of an application for the monitoring of an online sport activity, continuously informing the geographical location.

When starting the application to be registered and in case of not having a user you have to register. Once registered the user access to the application to start their activity, track their location by reporting latitude and longitude of their position per gps. This data is stored in a database on a remote Microsoft Azure server and if there is no online connection it is saved to the isolated storage of the phone to send information when it has been connected to the remote server.

In the development of the application as you can not directly interact with a SQL server from a Windows Phone application created a service layer around the database that exposes all the CRUD operations that is needed. This service is called from the application and for the technology that used has been the service WCF.

To make the application used the programming language c #, Xaml, LINQ to SQL.

The tools used have been Visual studio community 2015, Microsoft Azure, SQL server and visio.

The results obtained have been quite favorable.

As a conclusion of this work on the day of the location by Gps provides very high efficiency and with quite optimum performance and results.

Keywords:

Windows Phone, Geolocation, geographical location, GPS, remote server, XAML, LINQ to SQL, SDK, WCF, web service, Map, latitude, longitude, Service Reference, UI, Authentication, Implementation, Emulator, API

Licencia Creative Common

Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional (CC BY-NC-SA 4.0)

Esto es un resumen inteligible para humanos (y no un sustituto) de la licencia. Advertencia.

Usted es libre de:

Compartir — copiar y redistribuir el material en cualquier medio o formato

Adaptar — remezclar, transformar y crear a partir del material

El licenciador no puede revocar estas libertades mientras cumpla con los términos de la licencia.

Bajo las condiciones siguientes:

Reconocimiento — Debe reconocer adecuadamente la autoría, proporcionar un enlace a la licencia e indicar si se han realizado cambios<. Puede hacerlo de cualquier manera razonable, pero no de una manera que sugiera que tiene el apoyo del licenciador o lo recibe por el uso que hace.

NoComercial — No puede utilizar el material para una finalidad comercial.

CompartirIgual — Si remezcla, transforma o crea a partir del material, deberá difundir sus contribuciones bajo la misma licencia que el original.

No hay restricciones adicionales — No puede aplicar términos legales o medidas tecnológicas que legalmente restrinjan realizar aquello que la licencia permite.

Tabla de contenido

Estructura del documento.

Este documento ha sido redactado, siguiendo una estructura en la cual se diferencian tres partes. La primera es la memoria como tal del proyecto, en la que se muestra, sin gran nivel de detalle, el trabajo hecho. Después, se muestran los anexos, donde se detalla más el código. Por último, una tercera parte con la bibliografía, un listado de acrónimos y un índice de figuras.

Seguidamente, se muestra el esquema de la estructura:

- Parte I: Memoria:
  - Introducción
  - Descripción del trabajo realizado
  - Conclusiones
  - Posibles mejoras
- Parte II: Anexos:
  - Código de la aplicación.
- Parte III:
  - Bibliografía.
  - Acrónimos.
  - Índice figuras.

Parte I. Memoria

1. - Introducción.

En este apartado se explica y se describe en qué consiste el trabajo y la motivación por la cual me ha llevado a realizarlo y decidirme por él.

Después de un pequeño estudio del mercado, me he dado cuenta que en Windows Phone no existen muchas aplicaciones de este tipo, teniendo una valoración por parte de los usuarios no muy alta, por lo tanto ello ha sido uno de los motivos en decantarme en realizar este proyecto sobre esta plataforma, además otra motivación principal ha sido el que en un primer momento me parecía un proyecto interesante por poder tener muchísimas utilidades en la vida real, ya sea para la localización de personas a través de su teléfono móvil, como el seguimiento de actividades deportivas y gracias a este trabajo he podido adquirir un mayor conocimiento tanto del lenguaje C#, como XML.

El presente Trabajo Fin de Grado, tiene como objetivo la realización de una aplicación para Windows Phone 8, para el seguimiento de una actividad deportiva al aire libre todo ello realizado con las API de ubicación de Windows para determinar la localización geográfica actual del teléfono.

Como la aplicación sólo necesita la ubicación del usuario en el momento actual, he utilizado el espacio de nombres **Windows.Devices.Geolocation** que proporciona las clases para tener acceso a la ubicación geográfica del equipo, y la posición geográfica del usuario.

La actualización de la posicion del usuario en el pasa se hace cada cierto tiempo y no por el movimiento o cambio de posición del dispositivo ya que es mucho mejor para la vida de la batería del teléfono.

El emulador de Windows Phone que viene con el SDK incluye un simulador de sensor de ubicación, que ha permitido simular las ubicaciones geográficas del dispositivo.

2. - Antecedentes, Marco Teórico.

Como se ha dicho en la introducción existen hoy en día ciertas aplicaciones muy parecidas a la realizada en este trabajo, entre las cuales cabe destacar las siguientes:

- **Runtastic.**
  Esta aplicación se puede utilizar offline y con el simple aprovechamiento del chip GPS integrado en el terminal.

- **CoPilot GPS.**
  Con las opciones básicas: mapas sin conexión; navegación en coche, bicicleta o a pie; rutas alternativas y búsqueda de sitios.

- **GPS Maps Navigation.**
  Lleva un registro de las ubicaciones en tiempo real donde quiera que se vaya usando el localizador GPS avanzado.
- **GPS Tracker 2.0**
  La aplicación permite realizar un seguimiento de la ubicación del dispositivo Windows Phone

En todas se hace uso del espacio de nombres **Windows.Devices.Geolocation** para obtener la ubicación geográfica del dispositivo del usuario, es por ello que este concepto es un elemento fundamental que he tratado de cuidar al máximo detalle a la hora de realizar la aplicación, viendo todas las posibilidades que ofrece, ya que es utilizado sobretodo en el mundo móvil y en el desarrollo de aplicaciones móviles.

Es por esta importancia en auge que cada vez más smartphones, y no sólo los de gama alta, incluyen un GPS para poder determinar nuestra localización sobre el mapa. Precisamente es el sector del desarrollo de aplicaciones móviles el que está sabiendo ver la amplia variedad de posibilidades que ofrece la geolocalización. De hecho, hay muchas maneras en las que nos puede ayudar esta funcionalidad desde nuestro dispositivo móvil.

Por ejemplo, utilizando los datos de localización de nuestro dispositivo, encontrar comercios cercanos, cafeterías, cines, ferreterías, etc., o algo tan simple como establecer la franja horaria en la que nos encontramos simplemente conociendo la localización por GPS, información que obtiene de los diversos satélites que orbitan la Tierra. En mi caso lo he aprovechado para realizar la aplicación del seguimiento de una actividad deportiva al aire libre.

En ese sentido, es preciso aclarar algunos conceptos. Una aplicación puede utilizar el espacio de nombres de Geolocalización para solicitar acceso a la ubicación del usuario, obtener la ubicación de una vez, realizando de forma continua un seguimiento de la ubicación a través de los eventos de actualización de ubicación, o recibir alertas cuando el dispositivo ha entrado o salido de lugares de interés.

Esta aplicación también muestra cómo:

- Manejar los cambios en los permisos de ubicación.
- Registrarse para el evento ***StatusChanged*** del Geolocalizador y utilizar los ***StatusChangedEventArgs*** para determinar el estado de la posición actual.
- Obtener datos de satélite: Si está disponible, utiliza la clase ***GeocoordinateSatelliteData*** para obtener información adicional sobre la calidad de los datos de localización basados en satélites.
- Solicitar el acceso a la ubicación del usuario: Solicitud de acceso a la ubicación del usuario usando el método ***RequestAccessAsync***.

Importante: Hay que llamar a ***RequestAccessAsync*** antes de acceder a la ubicación del usuario. En ese momento, la aplicación debe estar en el primer plano y ***RequestAccessAsync*** debe llamar desde el hilo de interfaz de usuario. Hasta que el usuario otorga su permiso a la aplicación para su ubicación, la aplicación no puede acceder a los datos de localización.

Ayudar a los parámetros de ubicación del cambio de usuario: Enlace a la configuración de privacidad de ubicación de su aplicación en caso de que el usuario revoca el acceso al lugar, mientras que su aplicación está en primer plano. Llamar al método ***LaunchUriAsync*** con el URI:

"MS-settings: //privacidad/ubicación".

3. - Objetivos e hipótesis.

Por tanto una vez he revisado todo lo anterior ya estoy en condiciones de poder plantearme los objetivos de mi proyecto, los cuales especifican los propósitos que quiero conseguir en función de los conocimientos previos que tengo y de los que iré adquiriendo a lo largo del presente trabajo.

Entre los objetivos que he intentado conseguir cabe destacar los siguientes:

- A nivel personal :
  - Aprender un lenguaje de programación potente como es C#.
  - Tambien aprender XML y LINQ to SQL.
- El objetivo principal de este Trabajo de Fin de Grado es desarrollar una aplicación para dispositivo móvil en Windows Phone para la localización geográfica del usuario, donde los usuarios podrán registrarse, consultar y guardar datos relacionados con la posición, con el fin de seguir su ubicación mientras realizan una actividad.
- Otro de los principales objetivos del proyecto es la forma de llegar al usuario, por lo que una interfaz amigable y la facilidad de uso se hacen indispensables en este proyecto.
- La robustez es otro de los objetivos ya que través de la cual se mide, en su mayoría, la calidad del producto.
- Implementar una app para móvil que no consuma muchos recursos.
- El último de los objetivos a destacar es el reto que supone llevar a cabo un proyecto software desde cero, planificando todas sus etapas, desde el estudio y análisis, hasta la codificación y las pruebas necesarias
- Definir el público a quien va dirigida.
- Si la aplicación móvil que quiero lanzar está cubriendo una necesidad. La mejor forma de saberlo es preguntar a personas que sean público objetivo de la app móvil sobre si existe alguna aplicación que cubre esa necesidad y si estarían dispuestos a pagar por aplicaciones que cubran esa necesidad.

- Perspectivas de futuro de subir la app a windows store para que pueda descargarse.

4. - Software y hardware utilizado.

Para poder llevar a cabo la aplicación he necesitado el siguiente software:

- La herramienta de desarrollo **Visual Studio Community 2017.**

Figura1 - Visual Studio Community.

Visual Studio Community 2017 es un IDE completo y gratuito con características de productividad de codificación, herramientas de desarrollo móvil multiplataforma para Windows, iOS y Android, herramientas para el desarrollo web y en la nube y acceso a cientos de extensiones. Esta edición de Visual Studio es gratuita para desarrolladores individuales, desarrollo de código abierto, investigación académica, aprendizaje y equipos profesionales pequeños.

- El software **Visio profesional 2010.**

Figura2 – Visio Professinal 2016

Microsoft Visio sirve para diseñar diagramas de flujo y de procesos, mapas conceptuales, líneas de tiempo y organigramas con gran facilidad. Incluye también la opción de crear diagramas UML y a partir de bases de datos.
Los cronogramas de Gantt están contemplados dentro de los gráficos que es posible dibujar con esta herramienta y su materialización resulta muy sencilla. La presentación es visual, atractiva y moderna. Para crear un diagrama de Gantt con Microsoft Visio, tan sólo hay que especificar el nombre de las tareas, introducir las fechas de inicio y fin, y su duración

en unidades temporales, que pueden ser de horas, días, semanas y meses. Permite cambiar el orden de las filas para visualizar mejor las actividades y agregar hitos. Incluso deja transformar los hitos en tareas y, a la inversa, de forma sencilla. Las dependencias entre ellas se pueden crear para mostrar en el gráfico cuáles son las que preceden y/o suceden a otras, a la vez que para observar cómo los cambios en unas afectan a las otras. Puede agrupar varias tareas subordinadas en una tarea de resumen y agregar descripciones, recursos necesarios y porcentajes de finalización de cada actividad programada.

- **Microsoft Azure.**

Figura2.1 –Microsoft Azure

Los servicios en la nube de Microsoft Azure aprovechando la condición de ser estudiante, obtengo un código gratuito, para desarrollar en la nube sin costo alguno con Azure App Services, Hubs de notificación, base de datos SQL, creación de servidores y más.

- El software **SQL Server Management Studio** (SSMS).
  Es un entorno integrado para administrar cualquier infraestructura de SQL, desde SQL Server a SQL Database.SQL Server Management Studio proporciona herramientas para configurar, supervisar y administrar instancias de SQL Server desde cualquier lugar que se implementen.
  Lo he utilizado para gestionar la base de datos del servidor remoto Azure.

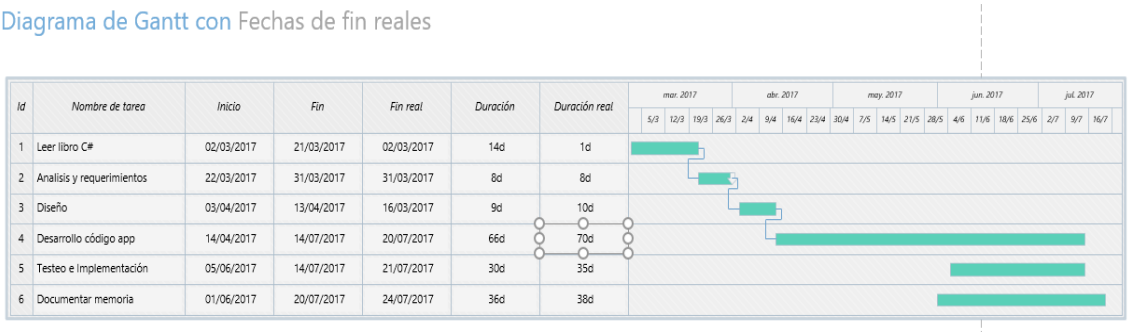
En cuanto al hardware utilizado, he podido desarrollar toda la aplicación con mi portátil, ya que reúne las todas las características para poder ejecutar el Visual Studio Community 2015 y el emulador de Windows Phone, ya que ambos requieren los siguientes requisitos de hardware, cosa que mi portátil reúne y supera con creces:

- Procesador de 1,6 GHz o más rápido
- 1 GB de RAM (1,5 GB si se ejecuta en una máquina virtual)
- 4 GB de espacio disponible en el disco duro
- Unidad de disco duro de 5400 rpm
- Tarjeta de vídeo compatible con DirectX 9 con una resolución de pantalla de 1024 x 768 o superior
- Otros requisitos

- Para la Tienda Windows y el desarrollo de aplicaciones universales de Windows
- El desarrollo de Windows 8.1 y Windows Phone 8.1 requiere Windows 8.1 Update o una versión posterior.
- El desarrollo de Windows Phone 8.0 requiere Windows 8.1 (x64) o una versión posterior.
- Para los emuladores de Windows, Windows 8.1 (x64) Professional Edition o versiones posteriores y un procesador que admita el Cliente Hyper-V y la traducción de direcciones de segundo nivel (SLAT).

5. – Diagrama de Gantt con tiempo estimado y real.

En el siguiente diagrama se puede ver el tiempo que he dedicado ha llevar a cabo la aplicación.



Id	Nombre de tarea	Inicio	Fin	Fin real	Duración	Duración real
1	Leer libro C#	02/03/2017	21/03/2017	02/03/2017	14d	1d
2	Analisis y requerimientos	22/03/2017	31/03/2017	31/03/2017	8d	8d
3	Diseño	03/04/2017	13/04/2017	16/03/2017	9d	10d
4	Desarrollo código app	14/04/2017	14/07/2017	20/07/2017	66d	70d
5	Testeo e Implementación	05/06/2017	14/07/2017	21/07/2017	30d	35d
6	Documentar memoria	01/06/2017	20/07/2017	24/07/2017	36d	38d

Figura 3 – Diagrama de Gantt.

En total en realizar el proyecto he tardado unos 85 días, dedicando diariamente cómo media unas 3 horas diarias, contabilizandolo serían unas 255 horas.

La fase de análisis, Diseño las he hecho de manera secuencial, es decir que hasta que no terminaba una no he empezado la otra, mientras que la de testeo e implementación he ido alternandolas y combinando con la de desarrollo, ya que hacia pruebas y si detectaba errores corregia el código y volvia a testear e implementar. Al mismo tiempo iba haciendo la memoria y documentando.

6. - Metodología.

Metodología de desarrollo de software en ingeniería de software es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información.

Para realizar la aplicación he combinado el modelo en cascada con el prototipado, ya que las dos primeras fases, analisis y diseño las he hecho de foma secuencial, es decir que hasta que no he terminado una no he empezado la otra, sin embargo déspues he utilizado el modelo prototipado ya que he ido desarrollando código, probando y retroalimentado con el cliente (yo mismo).

Gráfico que muestra la metodologia empleada en el proyecto:

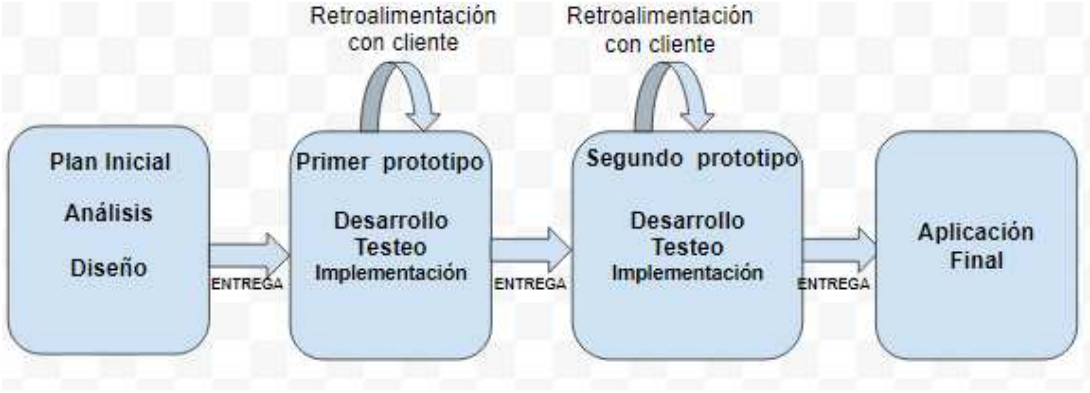


Figura3.1 – Gráfico modelo prototipado

6.1. - Análisis y Planificación.

Una vez conocidos y estudiados los objetivos a conseguir me he planteado una serie de cuestiones las cuales me han ayudado a seguir los pasos para desarrollar la app:

1. En primer lugar describir la funcionalidad de la aplicación:
   Cuántas pantallas diferentes habrá en la app móvil, que se podrá hacer en cada una de ellas, y sobre todo, qué tiene que hacer la propia app o el usuario dentro de la app móvil.
2. ¿Existe alguna app de referencia que pueda servir de ejemplo? Una aplicación similar aunque sea de otro sector o de la misma competencia me puede ayudar mucho a cómo desarrollar la aplicación móvil.
3. ¿Los usuarios de la aplicación necesitarán registrarse?
   En la aplicación he hecho que los usuarios para poder utilizarla se tengan que registrar.
4. ¿Necesitará la utilización de alguna característica del móvil? Cámara, geolocalización, compartir, multimedia, notificaciones push, agenda etc.
   La app no utiliza muchos recursos solamente la geolocalización y el almacenamiento.
   Hay que sacarle el máximo partido al dispositivo, teniendo en cuenta que las apps que necesitan mucha conexión de datos o que gastan mucha batería del dispositivo gustan poco y acaban borrándose.
5. ¿Para qué plataformas quiero la app?
   En este caso solo para Windows Phone.
6. ¿Necesitaré un servicio remoto con una base de datos que almacene datos?
   Para ello como soy estudiante he utilizado la plataforma de Microsoft Azure que proporciona servicios de servidores y bases de datos Sql para almacenar datos en la nube.

6.2. - Diseño.

El diseño consiste tanto en la confección del aspecto y usabilidad como en la correcta aplicación de las Guidelines de Windows Phone.

En primer lugar para diseñar las pantallas de la aplicación he dibujado en papel un prototipo de las mismas:

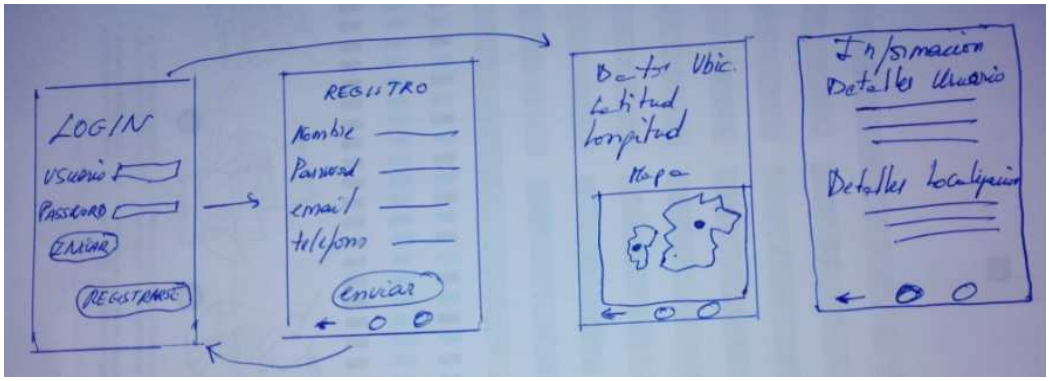


Figura4.1- Diseño pantallas en papel.

Esta ha sido la etapa más divertida de todo el proceso de desarrollo de la aplicación, la interfaz de una aplicación la capa que hay entre el usuario y la lógica de la aplicación, el lugar donde nacen las interacciones. En su mayor medida está compuesta por botones, gráficos, iconos y fondos, que tienen una apariencia.

Por tanto con el fin de seguir una linea correcta para llevar a cabo el diseño de la aplicación he intentado seguir las recomendaciones y consejos de WPDPS para Windows Phone.

1. Elementos de alineación y márgenes.
    a. Todas las páginas deben respetar el margen de 12 píxeles o 24px a la izquierda.
    b. Contenido, títulos, encabezados y los logotipos de cabecera deben ser alineados a la izquierda a 12 píxeles o margen de 24px.
    c. Si se requiere alineación a la derecha, el margen borde derecho será también 12px o 24px.

Para comprobarlo he habilitado la rejilla semitransparente de 25 × 25 cuadrados rojos de píxeles en la aplicación cuando se ejecuta en modo de depuración. Estas plazas están contenidas dentro de un acolchado de 24 píxeles alrededor de la página y se ven compensados por 12 píxeles entre uno y otro - el combo mágico para el diseño de Windows Phone. La cuadrícula permite identificar rápida y fácilmente cualquier problema de alineación en la página.

2. Pares de campo y el valor o bien presentarse en dos columnas de la izquierda alineados, como son los campos de Latitud y Longitud con sus valores.
3. Elemento de separación:
   El espacio entre los elementos debe ser coherente tanto horizontal como verticalmente. Se recomienda que los elementos estén espaciados un múltiplo o subdivisión de 12 píxeles separados con el fin de seguir la cuadrícula de diseño.
4. Colocación de los Botones.
   a. Siempre que ha sido posible, he intentado colocar los botones en la barra de aplicaciones.
5. Los cuadros de diálogo:
   a. He utilizado el cuadro de diálogo de Windows de teléfono estándar, con los botones individuales alineados a la izquierda y el centro de varios botones alineados.
   b. He evitado la creación de cuadros de diálogo personalizados, pero si esto es inevitable hay que asegurar el comportamiento de los imitadores de control personalizado que del cuadro de diálogo nativo.
6. El botón de retroceso no se ha alterado bajo ninguna circunstancia.

En cuanto a la estructura de la aplicación y su funcionamiento, en el siguiente diagrama de flujo se puede ver que después de iniciar la aplicación para poder logearse hay que estar registrado, si el usuario esta registrado puede consultar los datos de su cuenta, los datos de su localización geográfica o puede iniciar su localizacion en el mapa. Por el contrario si el usuario no está registrado si quiere entrar en la aplicación tiene que darse de alta. Estos datos se guardan en un servidor remoto de Azure en una base de datos SQL.

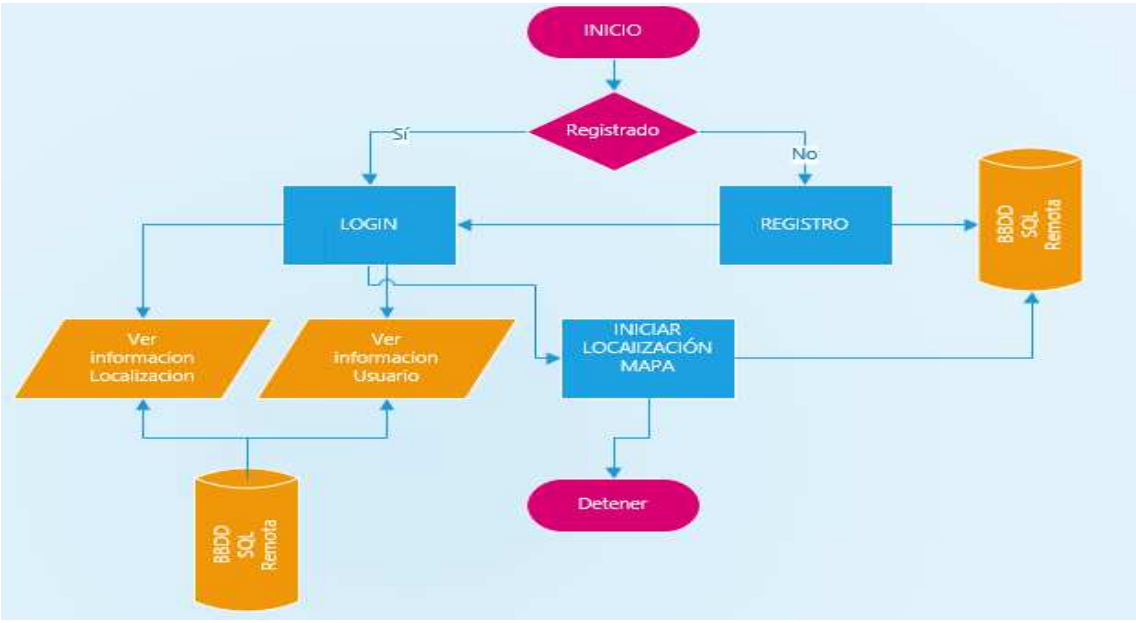


Figura4.2 – Diagrama de flujo aplicación.

Pantallas de la aplicación:

Figura5 – Pantallas de la aplicación.

6.3. - Desarrollo.

Para desarrollar la aplicación he tenido en cuenta que como los datos de los usuarios y de sus localizaciones geográficas se ubican tanto en almacenamiento aislado local del teléfono cuando no hay conexión online para conectar con el servidor, como en una base de datos SQL en Azure en la nube en un servidor remoto y como no se puede acceder directamente desde Windows Phone a la base de datos Sql entonces he tenido que crear un servicio web con tecnología WCF para poder realizar las operaciones CRUD con la base de datos y así realizar consultas de inserción, recuperación de datos, actualización, borrado etc.

El servicio web WCF consta de un archivo IService1.cs[1] en el cual defino las clases de Usuario y Ubicacion con sus atributos.

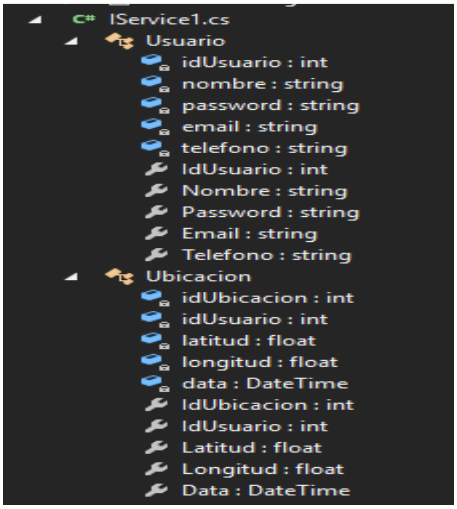


Figura6 - Interfaz

[1] Para ver codigo IService1.cs ver Anexo 8.1

Dentro de este archivo también defino una interfaz con las operaciones de servicio que serán llamadas desde la aplicación, para poder interactuar con la base datos y obtener o insertar los datos.

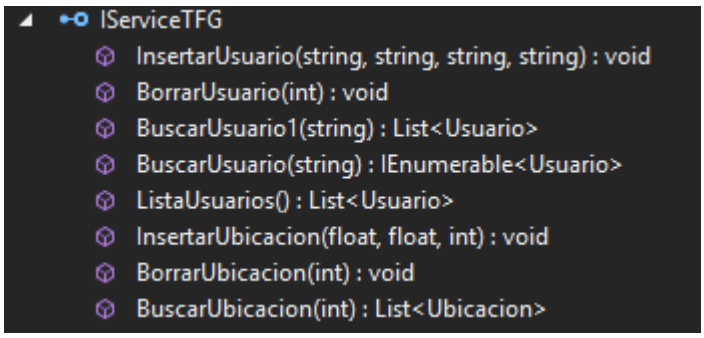


Figura7 – Métodos interfaz.

Estas operaciones se implementan en el archivo Service1.svc.cs [2]

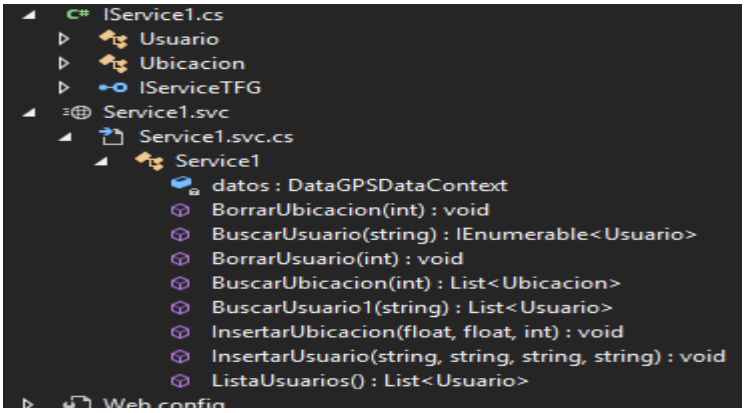


Figura8 – Implementacion metodos.

6.3.1. - Creación de la base de datos en el portal de Azure.

Para poder almacenar los datos en una base de datos SQL Server en un servidor remoto desde Windows Phone he utilizado la plataforma Azure que proporciona todos los recursos y herramientas necesarias:

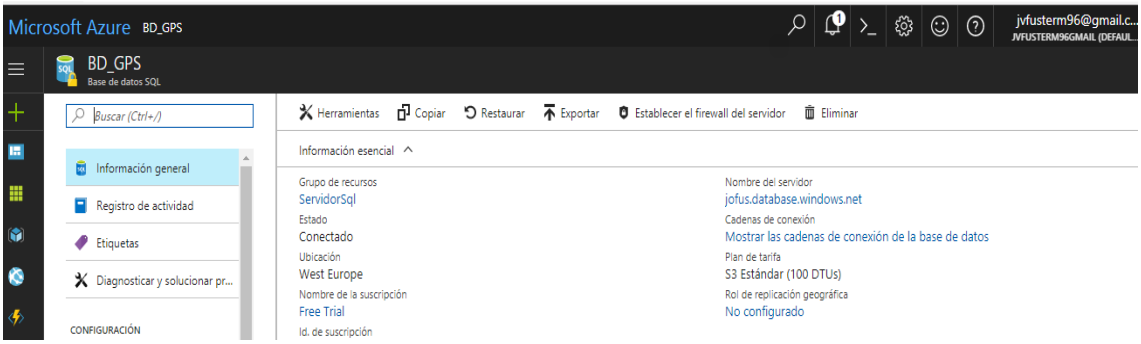


Figura 9 – Informacion Azure.

[2] Para ver código Service1.svc.cs ver Anexo 8.2

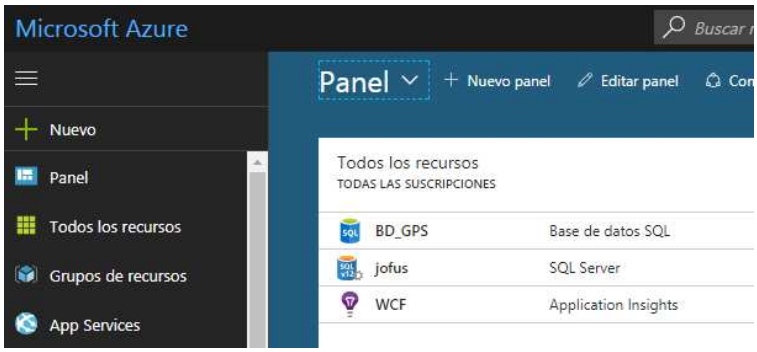
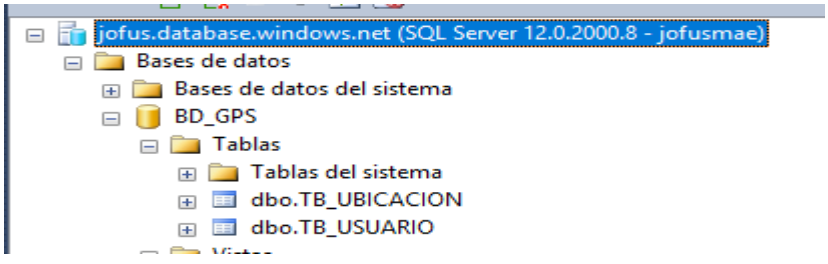


Figura 10 – Recursos Azure.

Y además con SQL Server Management Studio que proporciona herramientas para configurar, supervisar y administrar instancias de SQL Server desde cualquier lugar que las implemente, he creado las tablas y relaciones de la base de datos que tengo en el servidor remoto de Azure.

Figura 11 – Inicio sesión Management Studio.

He creado una base de datos con dos tablas: TB_USUARIO y TB_UBICACION.



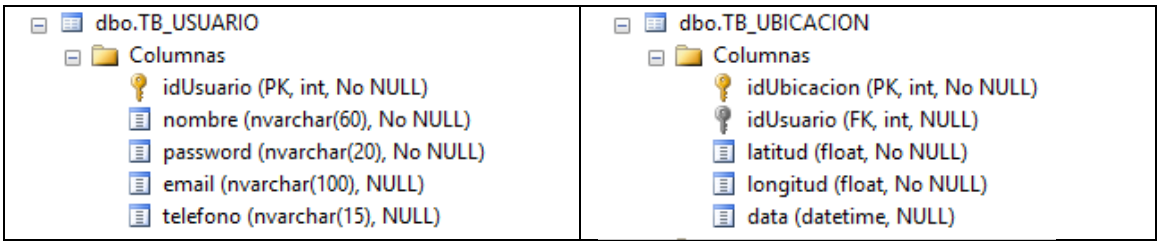


Figura 12 – Tablas de Aplicación.

6.3.2. - WCF service para manejar los datos de la base de datos SQL Server.

A continuación en visual studio para poder acceder desde Windows Phone a estos datos he creado un servicio web con WCF.

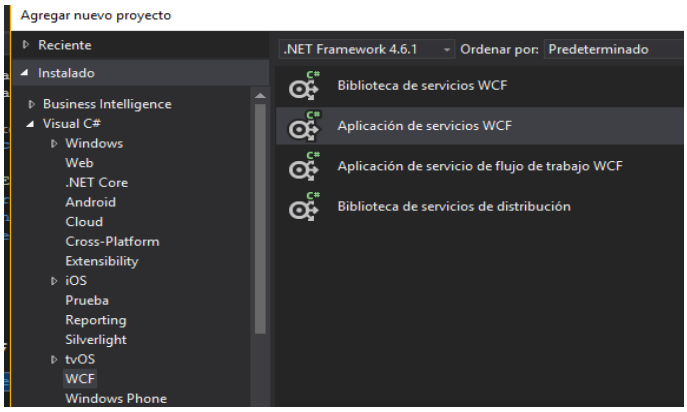


Figura 13 – Agregar WCF.

Agrego un nuevo elemento de clases de LINQ to SQL.

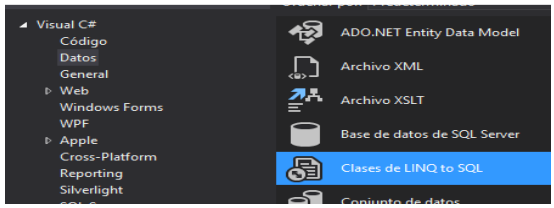


Figura 14 - Agregar LINQ toSQL.

Y en el explorador de servidores de visual studio creo una nueva conexión con SQL Server:

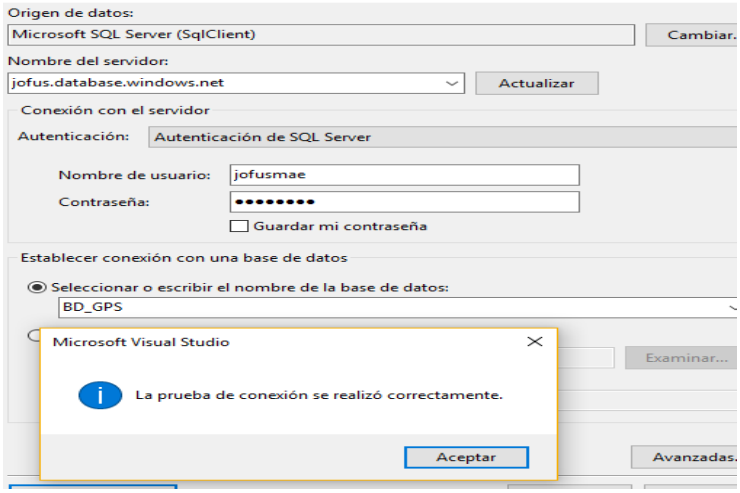


Figura 15 – Agregar conexión Base de datos.

Y tras aceptar:

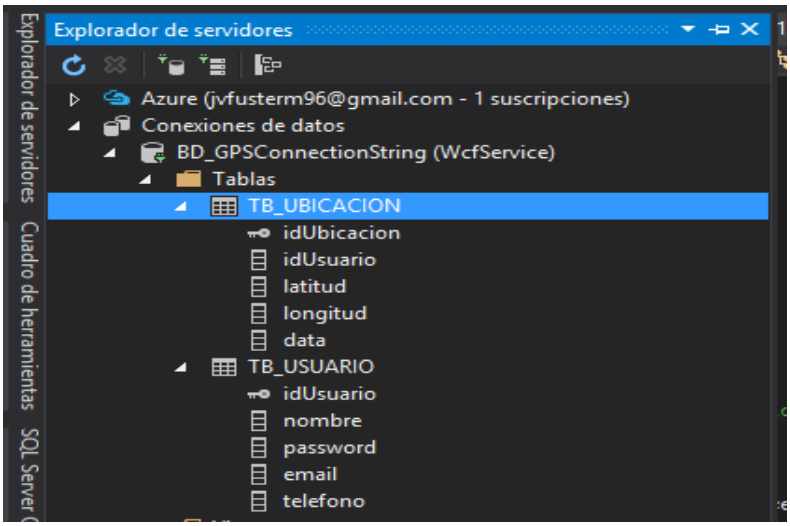


Figura 16 – Explorador de servidores.

Ahora la base de datos SQL está conectada a Visual Studio y se agrega al proyecto, quedando visible en el Explorador de servidores. Expando DB, luego las tablas y arrastro sus tablas a la pantalla central:

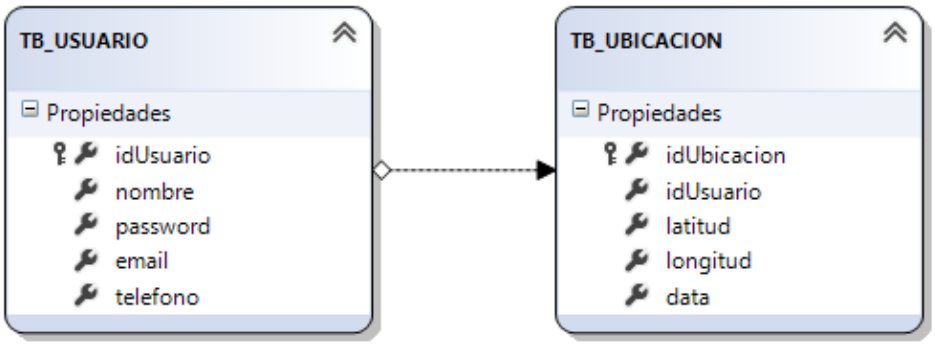


Figura 17 – Relación Entidades.

Creándose el modelo de datos y su contexto como se puede ver en la imagen siguiente:

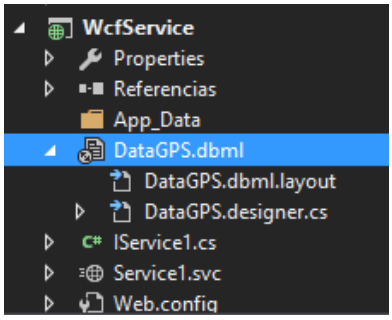


Figura 18 – Modelo datos.

A continuación abro el archivo IService1.cs y añado las dos clases: Usuario y Ubicación con sus atributos y las operaciones de servicio, que luego serán implementadas en el archivo Service.cs

- Ver código Interface ver Anexo: 8.1. – Código del archivo IService1.cs.

- Ver código implementación Interface Ver Anexo: 8.2. – Código del archivo Service1.svc.cs.

Para comprobar que el servicio web funciona correctamente hago click con el botón derecho en el archivo Service1.svc que está en el explorador de soluciones y después le digo ver en explorador (por ejemplo Google).

Figura 19 – Url Servicio web.

Esta Url Http://localhost:60423/Service1.svc?wsdl será la que depués utilizaré para agregar la referencia de servicio al proyecto de aplicación que contendrá las páginas para poder utilizar las operaciones de servicio o métodos implementados en WCF y acceder a la base de datos remota y obtener los datos o realizar otras consultas.

6.3.3. - Programando la aplicación cliente.

Una vez que tengo el servicio web instalado e implementado con todos los métodos y añadido el contexto de la base de datos entonces creo un nuevo proyecto que será la aplicación de Windows Phone que contendrá todas las páginas de la app, como el login, registro, detalles de la ubicación y usuario, localización geográfica en el mapa del usuario.

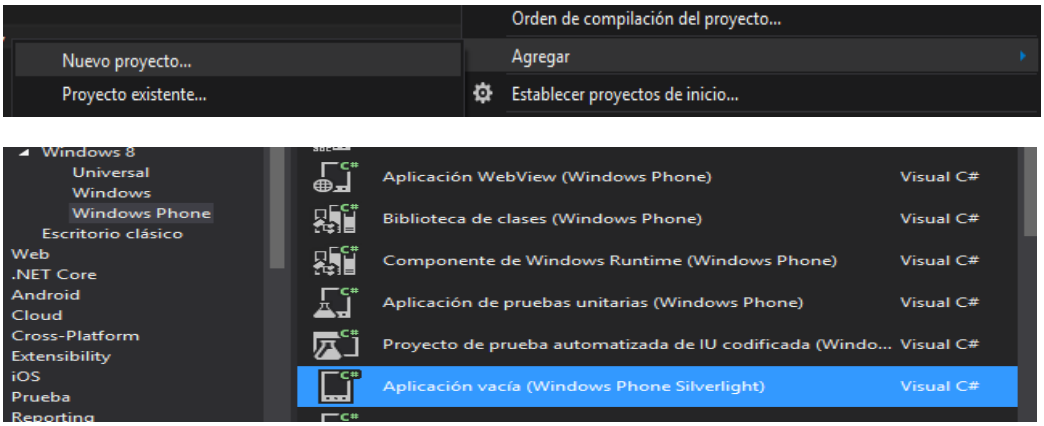


Figura 20 – Agregar nuevo elemento Windows Phone.

Para que las operaciones de servicio o métodos del servicio web puedan ser utilizadas por la aplicación tengo que agregar una referencia de servicio como se ve en la imagen:

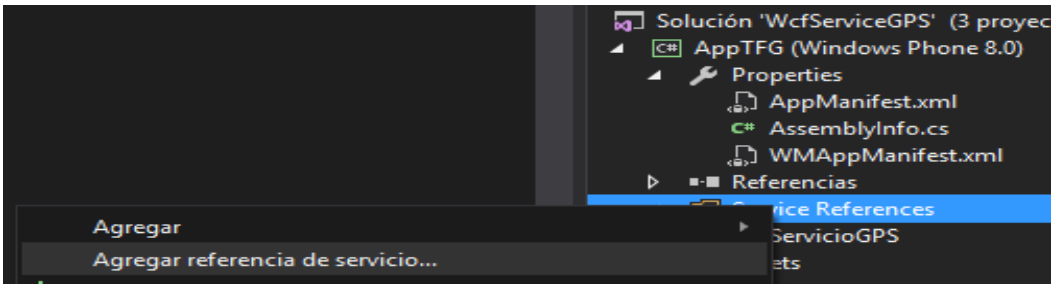


Figura 21 – Agregar referencia de servicio I.

Se abre una nueva pantalla en la cual introduzco la dirección del servicio web como se ve en la imagen siguiente:

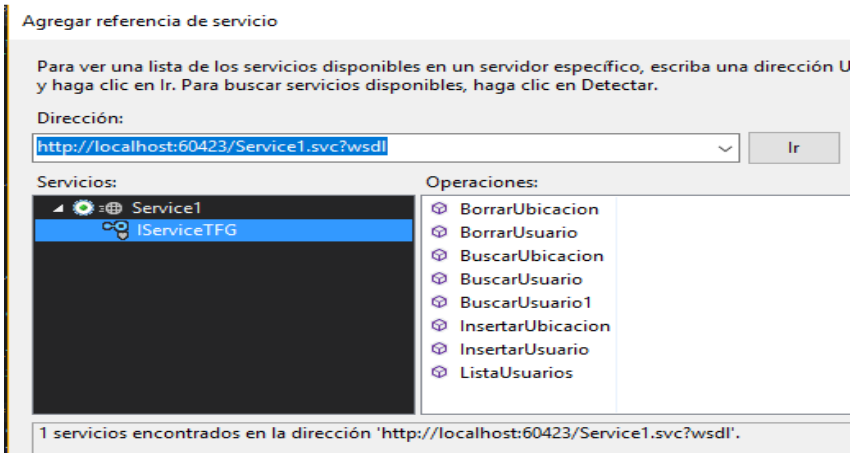


Figura 22 – Agregar referencia de servicio II.

Y tras aceptar se agrega la referencia de servicio a la aplicación gracias a la cual, desde las páginas de la aplicación se podrá acceder a los métodos implementados en el proyecto WCF y así manejar los datos que hay en la base de datos de Sql en Azure.

Una vez que esta añadido el proyecto a la solución que he creado con el web service para añadir las nuevas páginas dentro del proyecto, hago click con botón derecho sobre el proyecto de la aplicación y selecciono agregar nuevo elemento, eligiendo una página para Windows Phone:

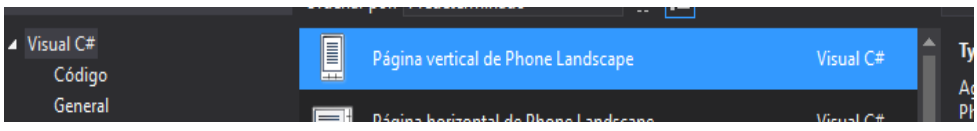


Figura 23 – Agregar página Windows Phone.

Quedando el proyecto con esta estructura:

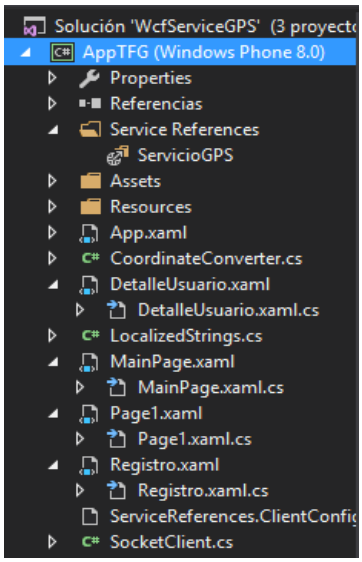


Figura 24 – Proyecto en explorador soluciones.

Para ver el código de los archivos:

- MainPage ver Anexo 8.3. – Código del archivo MainPage.xaml.cs.
- Registro ver Anexo 8.4. – Código del archivo Registro.xaml.cs.
- DetalleUsuario ver Anexo 8.5. – Código del archivo DetalleUsuario.xaml.cs.
- Clase CoordinateConverter.cs ver Anexo 8.6. – Código del archivo CoordinateConverter.cs.

6.3.2. – Cómo obtener la ubicación del usuario.

Para buscar la ubicación del usuario y responder a los cambios de ubicación en la configuración de privacidad de la aplicación Configuración se administra el acceso a la ubicación del usuario. También se comprueba si la aplicación tiene permisos para acceder a la ubicación del usuario.

6.3.2.1. - Habilitar la funcionalidad de ubicación.

En el Explorador de soluciones, dentro del proyecto en la carpeta propiedades en el archivo WMAppManifest.xml hago doble clic.

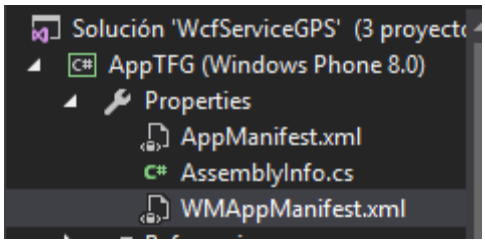


Figura 25 – Archivo WMAppManifest.xml

En la pestaña Capacidades, se agrega la funcionalidad Location del dispositivo al archivo *WMAppManifest.xml* del paquete.

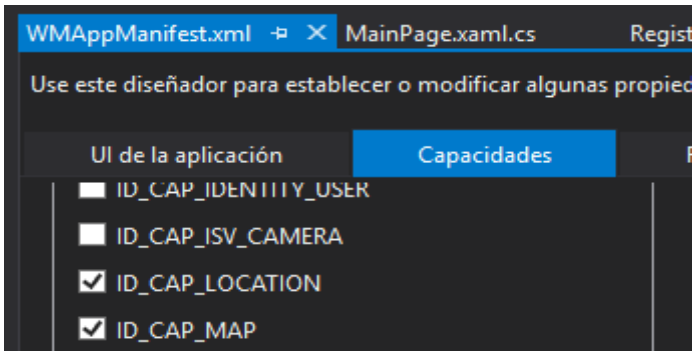


Figura 26 – Capacidades.

```
<Capabilities>
  <!-- DeviceCapability elements must follow Capability elements (if present) -->
  <DeviceCapability Name="location"/>
</Capabilities>
```

6.3.2.2. - Obtener la ubicación actual.

Para detectar la ubicación geográfica del usuario utilizo las API del espacio de nombres ***Windows.Devices.Geolocation.***

- Paso 1: Solicitar acceso para la ubicación del usuario

Hay que solicitar permiso para obtener acceso a la ubicación del usuario mediante el método ***RequestAccessAsync*** antes de intentar acceder a esta, también hay que llamar al método ***RequestAccessAsync*** desde el subproceso de la interfaz de usuario, y la aplicación debe estar en primer plano. Hasta que el usuario no conceda permiso a la aplicación, esta no podrá acceder a la información de ubicación del usuario.

```
using Windows.Devices.Geolocation;
...
var accessStatus = await Geolocator.RequestAccessAsync();
```

El método ***RequestAccessAsync*** pide permiso al usuario para acceder a su ubicación. Solo se pide confirmación al usuario una vez (por aplicación). Cuando el usuario concede o deniega el permiso por primera vez, este método ya no vuelve a solicitarlo. Para ayudar al usuario a cambiar los permisos de ubicación una vez se han solicitado, se recomienda proporcionar un vínculo a la configuración de ubicación, tal como se muestra más adelante.

- Paso 2: Obtener la ubicación del usuario y registrar los cambios en los permisos de ubicación.

El método ***GetGeopositionAsync*** realiza una lectura única de la ubicación actual. En la app se usa una instrucción switch con ***accessStatus*** (obtenido del ejemplo anterior), para que actúe solamente cuando se permita el acceso a la ubicación del usuario. Si se permite el acceso a la ubicación del usuario, el

código crea un objeto **Geolocator**, registra los cambios en los permisos de ubicación y solicita la ubicación del usuario.

```
switch (accessStatus)
{
  case GeolocationAccessStatus.Allowed:
     _rootPage.NotifyUser("Esperando actualizar...", NotifyType.StatusMessage);

     // Si DesiredAccuracy o DesiredAccuracyInMeters no están configurados (o el valor es 0),
DesiredAccuracy.Default se utiliza.
     Geolocator geolocator = new Geolocator { DesiredAccuracyInMeters =
_desireAccuracyInMetersValue };

     // Subscribe to the StatusChanged event to get updates of location status changes.
     _geolocator.StatusChanged += OnStatusChanged;

     // Carry out the operation.
     Geoposition pos = await geolocator.GetGeopositionAsync();

     UpdateLocationData(pos);
     _rootPage.NotifyUser("Location actualizado.", NotifyType.StatusMessage);
     break;

  case GeolocationAccessStatus.Denied:
     _rootPage.NotifyUser("Acceso a Location es denegado.", NotifyType.ErrorMessage);
     LocationDisabledMessage.Visibility = Visibility.Visible;
     UpdateLocationData(null);
     break;

  case GeolocationAccessStatus.Unspecified:
     _rootPage.NotifyUser("Error no especificado.", NotifyType.ErrorMessage);
     UpdateLocationData(null);
     break;
}
```

- Paso 3: Controlar cambios en los permisos de ubicación

El objeto ***Geolocator*** desencadena el evento ***StatusChanged*** para indicar que se ha modificado la configuración de ubicación del usuario. Ese evento pasa el estado correspondiente mediante la propiedad Status del argumento (de tipo ***PositionStatus***). Hay que tener en cuenta que no se llama a este método desde el subproceso de interfaz de usuario, y que el objeto ***Dispatcher*** invoca los cambios de la interfaz de usuario.

6.3.2.3. - Responder a actualizaciones de ubicación.

Esta sección describe cómo utilizar el evento ***PositionChanged*** para recibir actualizaciones de la ubicación del usuario durante un período de tiempo. Dado que el usuario puede revocar el acceso a la ubicación en cualquier momento, es importante llamar a ***RequestAccessAsync*** y utilizar el evento ***StatusChanged*** según se indica en la sección anterior.

Se supone que ya se ha habilitado la funcionalidad de ubicación y que se ha llamado a ***RequestAccessAsync*** desde el subproceso de la interfaz de usuario de la aplicación en primer plano.

- Paso 1: Definir el intervalo de informes y registrarse para obtener actualizaciones de ubicación.

En este ejemplo, se usa una instrucción switch con ***accessStatus*** (del ejemplo anterior) para que actúe solamente cuando se permita el acceso a la ubicación del usuario. Si se permite el acceso a la ubicación del usuario, el código crea un objeto ***Geolocator***, especifica el tipo de seguimiento y registra las actualizaciones de ubicación.

El objeto ***Geolocator*** puede desencadenar el evento ***PositionChanged*** en función de un cambio de posición (seguimiento basado en la distancia) o un cambio en el tiempo (seguimiento periódico).

Para realizar el seguimiento periódico, se establece la propiedad ***ReportInterval***.

Si no se establece ninguna propiedad, se devuelve una posición cada segundo (equivalente a ***ReportInterval*** = 1000). En este caso, se usa un intervalo de informes de 2 segundos (***ReportInterval*** = 2000).

```
using Windows.Devices.Geolocation;
...
var accessStatus = await Geolocator.RequestAccessAsync();
switch (accessStatus)
{
  case GeolocationAccessStatus.Allowed:
     // Crea Geolocator y defina el seguimiento basado en peródicos (intervalo de 2
segundos).
     _geolocator = new Geolocator { ReportInterval = 2000 };
     // Subscribe el evento PositionChanged actualizando location.
     _geolocator.PositionChanged += OnPositionChanged;

     // Subscribe el evento StatusChanged, actualiza los cambios de estado de location.
     _geolocator.StatusChanged += OnStatusChanged;
     _rootPage.NotifyUser("Esperando actualizar...", NotifyType.StatusMessage);
     LocationDisabledMessage.Visibility = Visibility.Collapsed;
     StartTrackingButton.IsEnabled = false;
     StopTrackingButton.IsEnabled = true;
     break;
  case GeolocationAccessStatus.Denied:
     _rootPage.NotifyUser("Acceso a location es denegado.", NotifyType.ErrorMessage);
        LocationDisabledMessage.Visibility = Visibility.Visible;
     break;
  case GeolocationAccessStatus.Unspecified:
     _rootPage.NotifyUser("error no especificado!", NotifyType.ErrorMessage);
     LocationDisabledMessage.Visibility = Visibility.Collapsed;
     break;
}
```

- Paso 2: Controlar actualizaciones de la ubicación

El objeto ***Geolocator*** desencadena el evento ***PositionChanged*** para indicar que ha pasado el tiempo, que hayamos configurado.

6.3.3. - Solicitar una clave de autenticación de mapas.

Para poder usar ***MapControl*** y los servicios de mapa en el espacio de nombres ***Windows.Services.Maps***. La aplicación debe autenticarse, para ello hay que especificar una clave de autenticación de mapas. A continuación describo cómo solicitar una clave de autenticación de mapas desde el Centro para desarrolladores de Mapas de Bing y agregarla a la aplicación.

Obtener una clave

Para poder crear y administrar claves de autenticación de mapas para las aplicaciones universales de Windows hay que ir al Centro para desarrolladores de Mapas de Bing. (https://www.bingmapsportal.com).

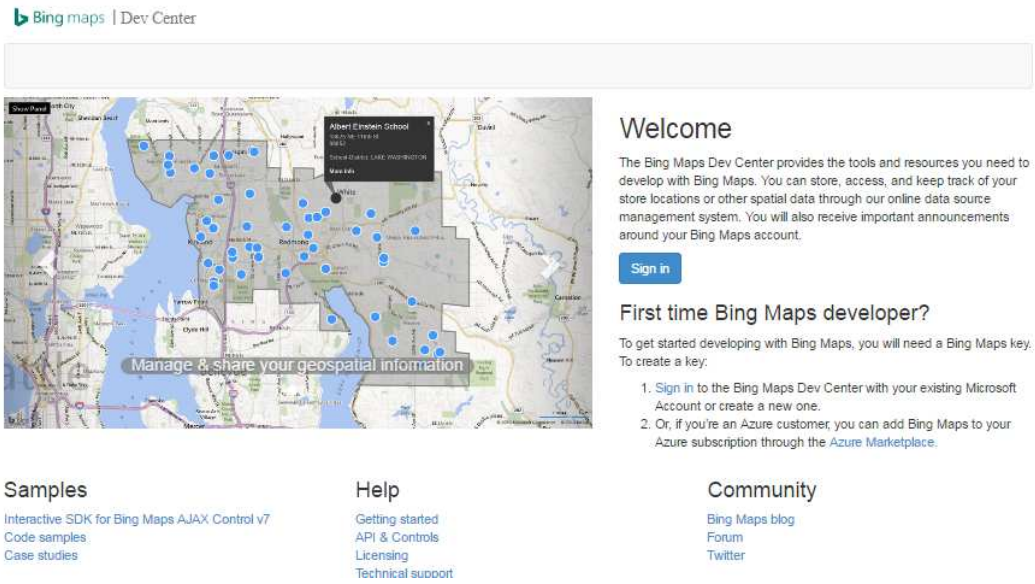


Figura 27 – Mapa bing.

- Si no se está registrado hay que registrarse e iniciar sesión con una cuenta de Microsoft. Elegimos la cuenta para asociarla con la cuenta de Mapas de Bing. Si queremos usar la cuenta de Microsoft, clicamos en sí, de lo contrario, hacemos clic en iniciar sesión con otra cuenta.
- Si no tenemos una cuenta de Mapas de Bing, creamos una nueva. Rellenamos los campos Nombre de cuenta, Nombre del contacto, Nombre de la compañía, Dirección de correo electrónico y Número de teléfono. Después de aceptar los términos de uso, hacemos clic en Crear.
- En el menú Mi cuenta, hacemos clic en Crear o ver claves.
- Hacemos clic en el vínculo para crear una clave nueva.
- Rellenamos el formulario Crear clave y, después en Crear.

- Nombre de la aplicación: el nombre de la aplicación.
- Dirección URL de la aplicación (opcional): dirección URL de la aplicación.
- Tipo de clave: seleccionamos Básica o Empresa.
- Tipo de aplicación: seleccionamos Aplicación de Windows para usarla en tu aplicación de Windows.
- A continuación se muestra un ejemplo del aspecto del formulario.

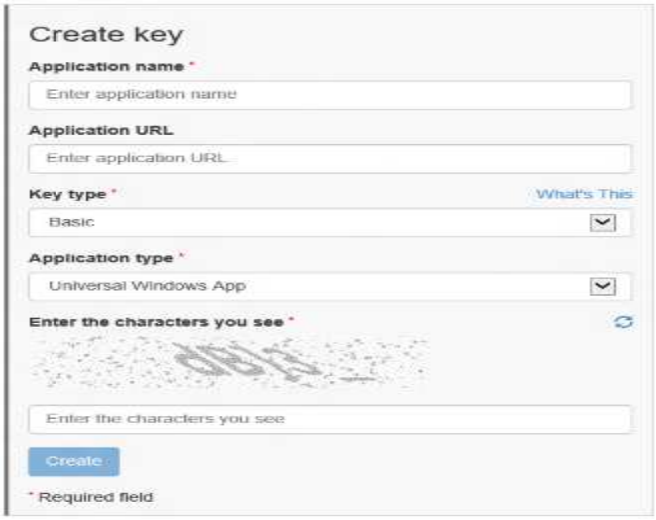


Figura 28 – Registro mapa bing.

- Después de hacer clic en Crear, la nueva clave aparece debajo del formulario Crear clave. La copiamos en un lugar seguro o la agregamos inmediatamente a la aplicación, como se describe en el siguiente paso.
- **Agregar la clave a la aplicación**
- La clave de autenticación de mapa es necesaria para usar MapControl y los servicios de mapa (Windows.Services.Maps) en la aplicación universal de Windows. La agregamos al control de mapa y asignamos los objetos de servicio, según corresponda.
- **Agregar la clave a un control de mapa**
- Para autenticar servicios en el espacio de nombres MapControl, establecemos la propiedad MapServiceToken en el valor de la clave de autenticación. Podemos establecer esta propiedad en el código o en el marcado XAML, según las preferencias. Para obtener más información sobre el uso de MapControl, consulta Mostrar mapas con vistas 2D, 3D y Streetside.
- Este ejemplo establece ***MapServiceToken*** en el valor de la clave de autenticación del código.

```
MapControl1.MapServiceToken = "abcdef-abcdefghijklmno";
```

- Este ejemplo establece ***MapServiceToken*** en el valor de la clave de autenticación del marcado XAML.

```
<Maps:MapControl x:Name="MapControl1"  MapServiceToken="abcdef-abcdefghijklmno"/>
```

- **Agregar la clave a servicios de mapa.**
- Para usar servicios en el espacio de nombres ***Windows.Services.Maps***, establecemos la propiedad ***ServiceToken*** en el valor de clave de autenticación. Este ejemplo establece ***ServiceToken*** en el valor de la clave de autenticación del código.

```
MapService.ServiceToken = "abcdef-abcdefghijklmno";
```

Para obtener un mayor rendimiento en la aplicación cuando se accede a la ubicación del usuario he seguido las siguientes recomendaciones:

- Utilizar el objeto ubicación sólo cuando la aplicación requiere datos de localización.
- Llamar al método ***RequestAccessAsync*** antes de acceder a la ubicación del usuario. En ese momento, la aplicación debe estar en el primer plano y ***RequestAccessAsync*** debe llamar desde el hilo de interfaz de usuario. Hasta que el usuario otorga su permiso a la aplicación para su ubicación, la aplicación no puede acceder a los datos de localización.
- El primer uso del objeto ***Geolocalizador*** debe hacerse en el hilo principal de la interfaz de usuario de la aplicación en activo, para desencadenar la petición de consentimiento para el usuario. El primer uso de la ***Geolocalizador*** puede ser la primera llamada a ***getGeopositionAsync*** o el primer registro de un controlador para el PositionChanged evento.
- Borrar datos de localización en caché y liberar el ***Geolocalizador*** cuando el usuario desactiva el acceso a la información de ubicación.

- Cuando se realiza una solicitud de ubicación de una sola vez, se debe establecer los siguientes valores.

- Especificar la precisión solicitada por la aplicación mediante el establecimiento de la ***DesiredAccuracy*** o los ***DesiredAccuracyInMeters***.

- Establecer el parámetro del tiempo máximo de ***GetGeopositionAsync*** para especificar cuánto tiempo hace que un lugar puede haber sido obtenida a ser útil para su aplicación. Si la aplicación puede utilizar una posición que está a unos pocos segundos o minutos de edad, la aplicación puede recibir una posición casi de inmediato y contribuir al ahorro de energía del dispositivo.

- Establecer el parámetro de tiempo de espera de ***GetGeopositionAsync***. Este es el tiempo que la aplicación puede esperar a una posición o un error que ser devuelto.

- Las aplicaciones que sí requieren datos en tiempo real se debe establecer ReportInterval a 0, para indicar que no se especifica el intervalo mínimo. El

intervalo de informe por defecto es 1 segundo o tan frecuente como el hardware puede soportar.

El usuario puede desactivar la funcionalidad de localización mediante la configuración de privacidad de ubicación en la aplicación Ajustes.

Si se trata de una aplicación en la que los datos de localización son esenciales, por ejemplo, una app-mapeo, hay que asegurarse de hacer lo siguiente:

Hay que tener en cuenta que el servicio de localización devolverá datos a medida que esté disponible. Puede que devuelva una ubicación con un radio mayor de error y luego actualizar la ubicación con la información más precisa en cuanto esté disponible.

Con ***Geocoordinate.accuracy*** indico la ubicación actual del usuario en el mapa con claridad. Hay tres bandas principales para una precisión-radio de error de aproximadamente 10 metros, un radio de error de aproximadamente 100 metros, y un radio de error de más de 1 kilómetro. Mediante el uso de la información de precisión, se puede asegurar que la aplicación muestra la ubicación precisa en el contexto de los datos disponibles.

Para una precisión aproximadamente igual a 10 metros (resolución) del GPS, la ubicación puede ser denotada por un punto o un alfiler en el mapa, en la aplicación inserto una elipse circular pequeña de color azul. Con esta precisión, las coordenadas de latitud y longitud se pueden mostrar también.

6.4. - Testeo.

Para probar la aplicación Windows Phone 8 incluye un simulador de sensor de ubicación. En este apartado se describe cómo probar la aplicación que utiliza datos de localización.

6.4.1. - Viendo el simulador de sensor de localización

Para ver el simulador de sensor de localización en Visual Studio, ejecuto la aplicación en el emulador, en la parte inferior de la barra de herramientas del emulador, hago clic en el botón con dos flechas para abrir la ventana de herramientas adicionales.

Figura 28 – Registro mapa de bing.

La barra de herramientas del simulador de sensor de localización contiene controles para encontrar un lugar, añadir puntos a la ubicación asignada, y grabar y reproducir sus datos de localización.

Hago clic en la ficha ubicación para ver el simulador de sensor de ubicación.

La siguiente imagen muestra la barra de herramientas simulador de sensor de ubicación.

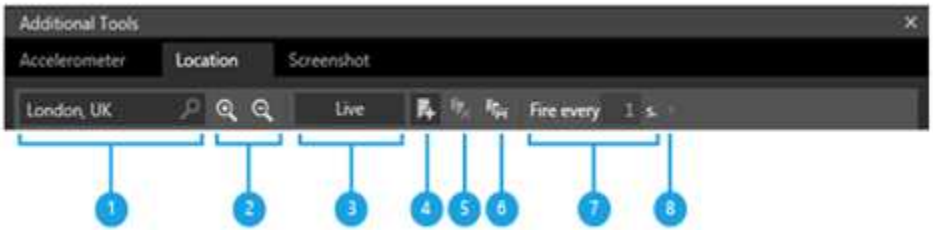


Figura 29 – Barra herramientas simulador.

1	Buscar
2	Enfocar
3	Alternar las interacciones en directo con un mapa o fuera
4	Modo de alfiler activar o desactivar
5	Borrar todos los puntos
6	puntos Guardar mapa
7	Intervalo en segundos
8	Juega todos los puntos

6.4.2. - Prueba de aplicaciones con el simulador de sensor de localización.

Se puede probar el sensor de localización con los siguientes tipos de datos de localización.

- Datos en tiempo real que se introducen en el simulador cuando la aplicación se ejecuta.
- Los datos grabados que ha introducido en el simulador antes de ejecutar la aplicación.
- Los datos grabados que se guardan en un archivo de una sesión anterior.

El simulador de sensor de localización no puede cargar más de 99 ubicaciones.

Para probar una aplicación con la entrada en directo de los datos de ubicación

1. En Visual Studio, ejecutamos la aplicación.
2. Abrimos la ventana Herramientas adicionales y haga clic en la ficha Ubicación.
3. Nos aseguramos que el modo de entrada en vivo está en moviendo el botón Live.
   Si el botón está en gris en vivo, en directo modo de entrada está apagado.
4. En el cuadro Buscar, escribimos la ubicación en la que deseamos mostrar en el mapa.

5. Hacemos clic en el botón Buscar o presionamos Intro.
6. Los controles de zoom para acercar o alejar de la ubicación asignada.

La siguiente imagen muestra el simulador de sensor de localización con una ubicación especificada.

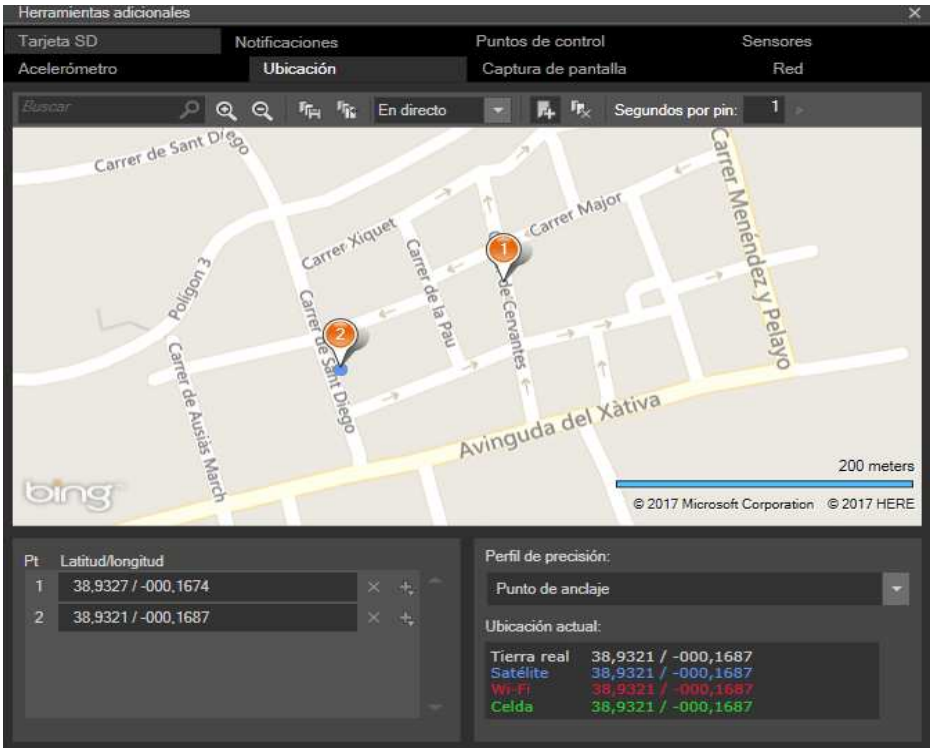


Figura 30 – Simulador sensor localización.

6.4.2. - Utilización de la clase GeoCoordinateWatcher con el simulador de sensor de localización.

Como la aplicación utiliza la clase ***GeoCoordinateWatcher***, hay que especificar un valor de ***GeoPositionAccuracy*** alto en el constructor o en la propiedad ***DesiredAccuracy*** de la clase antes de poder probar la aplicación con el simulador de sensor de ubicación. Si se deja la exactitud en su valor por defecto de ***GeoPositionAccuracy***. Por defecto, el ***PositionChanged*** no reconoce los cambios de posición que se producen en el simulador de sensor de ubicación.

Sin embargo, las posiciones obtenidas con un ajuste de ***GeoPositionAccuracy*** por defecto son satisfactorios para muchas aplicaciones. La obtención de una posición de alta precisión es más lenta y consume más energía. Si la aplicación no requiere una posición de alta precisión, se puede restablecer el valor de precisión al por defecto después de que se termine de probar la aplicación en el emulador.

6.4.3. – Ejecución de prueba de analisis de codigo de la aplicación.

El entorno de desarrollo Visual Studio permite poder analizar el código para poder ver si hay errores en la programación, para ello en la ventana de explorador de soluciones hago click con el botón derecho sobre la solución y a

continuación en ejecutar análisis de código de la solución o simplemente pulsando las teclas Alt+F11:

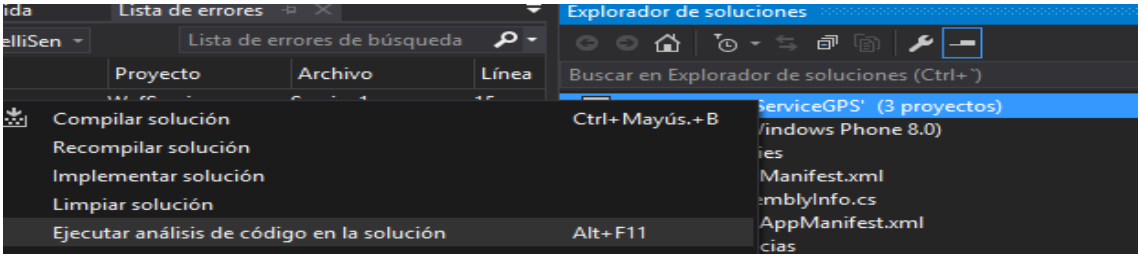


Figura 31 – Ejecutar análisis de código.

Después de pulsar aparecen una serie de advertencias que no son errores pero que ayudan a mejorar el código y por tanto el rendimiento de la aplicación.

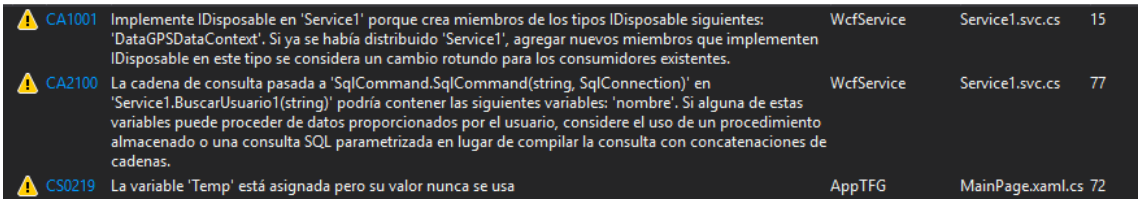


Figura 32 – Advertencias análisis de código.

Esto es una gran ayuda ya que informa con gran detalle donde está el error y a que es debido y aconseja cómo mejorarlo, gracias a ello he podido corregir algunos errores que ocurrían.

6.4.4. – Resultados de las pruebas realizadas de la aplicación.

En primer lugar he hecho una prueba para ver el registro de usuario y el inicio de sesión desde la aplicación, así cómo el funcionamiento de las pantallas de información sobre los usuarios y sus ubicaciones:

DESCRIPCIÓN	RESULTADO
Registrarse sin indicar algún campo	No permite el registro
Introducir caracteres no permitidos en nombre, email	Muestra una advertencia y no permite el registro
Registrarse con un correo electrónico que no tiene formato email	No permite el registro
Intento de registro de un usuario ya existente	No permite el registro
Dar de baja el usuario	Se da de baja el usuario y se elimina de la BD
Inicio de sesión con usuario no registrado	No permite el inicio de sesión en la aplicación
Acceder al detalle de un usuario	Permite ver el detalle
Visualización de posiciones geográficas en el mapa	Permite ver la posiciones en el mapa
Inserción de datos en servidor remoto Azure	Satisfactoria la inserción de datos
Consulta de datos en servidor remoto	Satisfactoria la recuperación de datos

Para comprobar el rendimiento de la aplicación he utilizado el administrador para ver el uso de cpu, memoria etc.

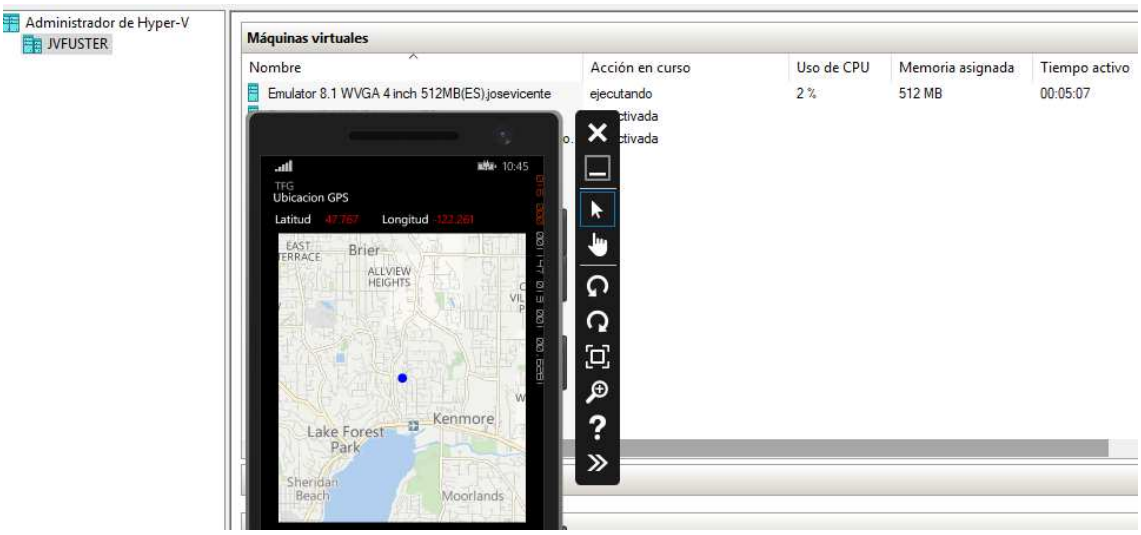


Figura 33 – Administrador Hyper-V.

Como se puede ver en la imagen el uso que hace de los recursos de hardware es muy bajo y óptimo.

6.5. - Implementación.

Antes de la implementación me he asegurado que en el archivo *WMAppManifest.xml* que está en la carpeta propiedades de la aplicación, están habilitadas las capacidades necesarias para que la aplicación puede acceder a los recursos necesarios para obtener la localización geográfica en el simulador así como el acceso a internet.

La implementación de la aplicación no la he podido llevar a cabo en ningun dispositivo de teléfono de windows phone 8 para probarla ya que no dispongo de ninguno, lo que sí que me gustaría como perspectiva de futuro es crear una cuenta de desarrollador y subir la aplicación a la tienda de windows phone para que pueda descargarse y poder ser utilizada en dispositivos reales.

Pero como el SDK de Windows Phone 8 en Visual Studio 2017 contiene las suficientes bibliotecas y herramientas con un emulador de Windows Phone 8 que permite poder localizar la ubicaciones geográficas entonces tras compilar la solución se puede realizar con muchas garantías la implementación de la aplicación en el simulador dando unos resultados bastante fiables que se asemejan bastante a la realidad.

7. - Conclusiones.

Como conclusión de este proyecto lo positivo que saco es que después de llevar a cabo de principio a fin todas las fases del mismo intentando resolver bastantes problemas como el poder conectarme desde Windows Phone a la base de datos, lo cual me costó bastante por errores que ocurrían por excepciones que al principio no controlaba, pero que después he ido aprendiendo a cómo manejarlas he adquirido un poco de experiencia en el desarrollo de una aplicación con C# y el lenguaje de marcas XML.

En un primer momento como no podía establecer conexión con un servidor remoto, simulé uno en local con una aplicación de escritorio al cual la aplicación cliente se comunicaba por socket enviando los datos de localización, pero después he averiguado que con la plataforma Azure se pueden crear servidores remotos y bases de datos SQL para almacenar los datos de los usuarios y sus ubicaciones, lo cual da una mayor utilidad a la aplicación.

Después de la realización de este proyecto a nivel personal he mejorado mi formación aprendiendo el lenguaje C# y XML, además he tenido que aportar soluciones a los problemas que surgían e investigar cómo resolver ciertas dificultades que he ido encontrando, descubriendo herramientas que nos proporciona el entorno de desarrollo de visual studio como la ejecución de análisis de código, así como el saber interpretar los debug y el uso de trazas para poder analizar el código paso a paso.

7.1. – Posibles mejoras.

La aplicación desarrollada cumple con la función y requisitos especificados, sin embargo, se pueden realizar futuras mejoras en cuanto a seguridad, ya que los datos se envían sin cifrar al servidor remoto de Azure.

Otra mejora sería que una tercera persona pudira seguir el seguimiento de la ubicación geográfica con alguna opción de la misma app. o bien mediante página web.

Además otra mejora que podría realizar es que muestre la ruta seguida por el usuario al ir de un lugar geográfico a otro, así como su velocidad y distancia.

Como Windows Phone hoy en día en el Mercado no tiene mucho éxisto ya que tanto Android como IOS lo abarcan todo, tengo pensado en un futuro desarrollarlo con Xamarin ya que permite desarrollar con lenguaje de programación C# aplicaciones para Android, IOS y Windows Phone pudiendo compartir código.

Parte II.

8. - Anexos.

8.1. – Código del archivo IService1.cs.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.ServiceModel.Web;
using System.Text;

namespace WcfService
{
    [DataContract]
    public class Usuario
    {
        int idUsuario;
        string nombre, password, email, telefono;
        [DataMember]
        public int IdUsuario
        {
            get { return idUsuario; }
            set { idUsuario = value; }
        }
        [DataMember]
        public string Nombre
        {
            get { return nombre; }
            set { nombre = value; }
        }
        [DataMember]
        public string Password
        {
            get { return password; }
            set { password = value; }
        }
        [DataMember]
        public string Email
        {
            get { return email; }
            set { email = value; }
        }
        [DataMember]
        public string Telefono
        {
            get { return telefono; }
            set { telefono = value; }
        }
    }
    [DataContract]
    public class Ubicacion
    {
        int idUbicacion, idUsuario;
        float latitud, longitud;
```

```
        DateTime data;
        [DataMember]
        public int IdUbicacion
        {
            get { return idUbicacion; }
            set { idUbicacion = value; }
        }
        [DataMember]
        public int IdUsuario
        {
            get { return idUsuario; }
            set { idUsuario = value; }
        }

        [DataMember]
        public float Latitud
        {
            get { return latitud; }
            set { latitud = value; }
        }
        [DataMember]
        public float Longitud
        {
            get { return longitud; }
            set { longitud = value; }
        }
        [DataMember]
        public DateTime Data
        {
            get { return data; }
            set { data = value; }
        }
    }

    [ServiceContract]
    public interface IServiceTFG
    {
        [OperationContract]
        void InsertarUsuario(string nombre, string password, string email, string telefono);

        [OperationContract]
        void BorrarUsuario(int idUsuario);

        [OperationContract]
        List<Usuario> BuscarUsuario1(string nombre);

        [OperationContract]
        IEnumerable<Usuario> BuscarUsuario(string nombre);

        [OperationContract]
        List<Usuario> ListaUsuarios();

        [OperationContract]
        void InsertarUbicacion(float latitud, float longitud, int idUsuario);

        [OperationContract]
        void BorrarUbicacion(int idUbicacion);

        [OperationContract]
        List<Ubicacion> BuscarUbicacion(int idUsuario);
    }
}
```

8.2. – Código del archivo Service1.svc.cs.

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.ServiceModel.Web;
using System.Text;
using WcfService;

namespace WcfService
{
   public class Service1 : IServiceTFG
   {
      DataGPSDataContext datos = new DataGPSDataContext();
      public void BorrarUbicacion(int idUbicacion)
      {
         var BorrarUbicacion = (from ubi in datos.TB_UBICACION
                                where ubi.idUbicacion == idUbicacion
                                select ubi).Single();
         datos.TB_UBICACION.DeleteOnSubmit(BorrarUbicacion);
         datos.SubmitChanges();
      }
      public IEnumerable<Usuario> BuscarUsuario(string nombre)
      {
         IEnumerable<Usuario> usuarioQuery =
         from user in datos.TB_USUARIO
         where user.nombre == nombre
         select new Usuario
         {
            IdUsuario = user.idUsuario,
            Nombre = user.nombre,
            Password = user.password,
            Telefono = user.telefono
         };

         return usuarioQuery.AsEnumerable(); ;
      }
      public void BorrarUsuario(int idUsuario)
      {
         var BorrarUsuario = (from user in datos.TB_USUARIO
                              where user.idUsuario == idUsuario
                              select user).Single();
         datos.TB_USUARIO.DeleteOnSubmit(BorrarUsuario);
         datos.SubmitChanges();
      }

      public List<Ubicacion> BuscarUbicacion(int idUsuario)
      {
         var qry = from ubi in datos.TB_UBICACION
                   where ubi.idUsuario == idUsuario
                   select new Ubicacion
                   {
                      IdUbicacion = ubi.idUbicacion,
                      Latitud = (float)ubi.latitud,
                      Longitud = (float)ubi.longitud,
                      IdUsuario = (int)ubi.idUsuario
                   };
         return qry.ToList();
```

```
    }

    public List<Usuario> BuscarUsuario1(string nombre)
    {
        SqlConnection con = new SqlConnection("BD_GPSConnectionString");
        SqlDataReader rdr = null;
        List<Usuario> list = new List<Usuario>();

        try
        {
            // 2. Abro conexión
            con.Open();

            // 3. Paso la conexión a un objeto de SqlCommand
            SqlCommand cmd = new SqlCommand("select * from USUARIO where nombre==" +
nombre, con);

            //
            // 4. Use the connection
            //

            rdr = cmd.ExecuteReader();

            Usuario user = new Usuario();
            while (rdr.Read())
            {
                user.IdUsuario = rdr.GetInt32(0);
                user.Nombre = rdr.GetValue(1).ToString();
                user.Password = rdr.GetValue(2).ToString();
                user.Email = rdr.GetValue(3).ToString();
                user.Telefono = rdr.GetValue(4).ToString();
                list.Add(user);
            }
        }
        finally
        {
            // Cierro reader
            if (rdr != null)
            {
                rdr.Close();
            }

            // 5. Cierro conexión
            if (con != null)
            {
                con.Close();
            }
        }
        return list.ToList();
    }

    public void InsertarUbicacion(float latitud, float longitud, int idUsuario)
    {
        TB_UBICACION ubi = new TB_UBICACION
        {
            latitud = latitud,
            longitud = longitud,
            idUsuario = idUsuario
        };
        datos.TB_UBICACION.InsertOnSubmit(ubi);
```

```
            datos.SubmitChanges();
        }

        public void InsertarUsuario(string nombre, string password, string email, string telefono)
        {
            TB_USUARIO user = new TB_USUARIO
            {
                nombre = nombre,
                password = password,
                email = email,
                telefono = telefono
            };
            datos.TB_USUARIO.InsertOnSubmit(user);
            datos.SubmitChanges();
        }

        public List<Usuario> ListaUsuarios()
        {
            SqlConnection con = new SqlConnection("BD_GPSConnectionString");
            con.Open();
            var qry = from user in datos.TB_USUARIO
                      select new Usuario
                      {
                          IdUsuario = user.idUsuario,
                          Nombre = user.nombre,
                          Password = user.password,
                          Email = user.email,
                          Telefono = user.telefono
                      };
            con.Close();
            return qry.ToList();
        }
    }
}
```

8.3. – Código del archivo MainPage.xaml.cs.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Navigation;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;
using AppTFG.Resources;
using System.IO.IsolatedStorage;
using System.Windows.Media;
using AppTFG.ServicioGPS;

namespace AppTFG
{
    public partial class MainPage : PhoneApplicationPage
    {
        IsolatedStorageFile ISOFile = IsolatedStorageFile.GetUserStoreForApplication();
        List<Usuario> ObjUserDataList = new List<Usuario>();
        ServicioGPS.ServiceTFGClient proxy = new ServicioGPS.ServiceTFGClient();
        // Constructor
        public MainPage()
```

```
        {
            InitializeComponent();
            ApplicationBar.Mode = ApplicationBarMode.Default;
            ApplicationBar.Opacity = 1.0;
            ApplicationBar.IsVisible = true;
            ApplicationBar.IsMenuEnabled = true;
            proxy.BuscarUsuarioCompleted += new
EventHandler<BuscarUsuarioCompletedEventArgs>(buscar);
        }

        private void SignUp_Click(object sender, RoutedEventArgs e)
        {
            NavigationService.Navigate(new Uri("/Registro.xaml", UriKind.Relative));
        }

        protected override void OnBackKeyPress(System.ComponentModel.CancelEventArgs e)
        {
            var Result = MessageBox.Show("¿ Seguro que quieres salir ?", "",
MessageBoxButton.OKCancel);
            if (Result == MessageBoxResult.OK)
            {
                if (NavigationService.CanGoBack)
                {
                    while (NavigationService.RemoveBackEntry() != null)
                    {
                        NavigationService.RemoveBackEntry();
                    }
                }
            }
            else
            {
                e.Cancel = true;
            }
        }
        private void UserName_GotFocus(object sender, RoutedEventArgs e)
        {
            tbxUname.BorderBrush = new SolidColorBrush(Colors.LightGray);
            tbxpwd.BorderBrush = new SolidColorBrush(Colors.LightGray);
        }

        private void cancel_Click(object sender, RoutedEventArgs e)
        {
           Debuger.Break().
        }

        private void signin_Click(object sender, RoutedEventArgs e)
        {

            if (tbxUname.Text != "" && tbxpwd.Password != "")
            {
                int Temp = 0;

                try
                {
                    proxy.BuscarUsuarioAsync(tbxUname.Text);
                }
                catch
                {
                    Temp = 0;
                }
```

```
            }
        }

        private void buscar(object sender, BuscarUsuarioCompletedEventArgs e)
        {
            int idUsuario = 0; string nombre = "", password = "", email = "", telefono = "";
            var user = e.Result;
            var usu = user.GetEnumerator();
            while (usu.MoveNext())
            {
                idUsuario = usu.Current.IdUsuario;
                nombre = usu.Current.Nombre.ToString();
                password = usu.Current.Password.ToString();
                email = usu.Current.Email;
                telefono = usu.Current.Telefono.ToString();
            }
            if (nombre == tbxUname.Text && password == tbxpwd.Password.ToString())
            {
                new DetalleUsuario(nombre);
                NavigationService.Navigate(new Uri("/DetalleUsuario.xaml?parametro=" + nombre +
":" + idUsuario.ToString(), UriKind.Relative));
            }
            else
            {
                textBox1.Text = "Usuario no registrado";
            }
        }

        private void Ver_usuarios(object sender, EventArgs e)
        {
            string s = tbxUname.Text;
            this.Content = new DetalleUsuario(s);
            NavigationService.Navigate(new Uri("/DetalleUsuario.xaml", UriKind.Relative));
        }

        private void Mapa(object sender, EventArgs e)
        {
            NavigationService.Navigate(new Uri("/Mapa.xaml", UriKind.Relative));
        }
    }
}
```

8.4. – Código del archivo Registro.xaml.cs.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Navigation;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;
using System.IO.IsolatedStorage;
using System.IO;
using System.Runtime.Serialization;
using System.Windows.Media;
using System.Text.RegularExpressions;
using AppTFG.ServicioGPS;
```

```
namespace AppTFG
{
   public partial class Registro : PhoneApplicationPage
   {
      IsolatedStorageFile ISOFile = IsolatedStorageFile.GetUserStoreForApplication();
      List<Usuario> ObjUserDataList = new List<Usuario>();
      public Registro()
      {
         InitializeComponent();
         this.Loaded += Registro_Loaded;
      }
      private void Registro_Loaded(object sender, RoutedEventArgs e)
      {
         if (ISOFile.FileExists("RegistrationDetails"))
         {
            MessageBox.Show("El fichero RegistrationDetails existe");
            using (IsolatedStorageFileStream fileStream =
ISOFile.OpenFile("RegistrationDetails", FileMode.Open))
            {
               DataContractSerializer serializer = new
DataContractSerializer(typeof(List<Usuario>));
               ObjUserDataList = (List<Usuario>)serializer.ReadObject(fileStream);
            }
         }
      }
      private void enviar(object sender, RoutedEventArgs e)
      {
         //Validación nombre
         if (!Regex.IsMatch(TxtUserName.Text.Trim(), @"^[A-Za-z_][a-zA-Z0-9_\s]*$"))
         {
            MessageBox.Show("Invalido nombre usuario");
         }

         // longitud Password
         else if (TxtPwd.Password.Length < 4)
         {
            MessageBox.Show("Longitud del Password debe ser minimo de 4 caracteres!");
         }

         //validación telefono
         else if (TxtMovil.Text.Length != 9)
         {
            MessageBox.Show("Telefono inválido");
         }
         // validación email
         else if (!Regex.IsMatch(TxtEmail.Text.Trim(), @"^([a-zA-Z_])([a-zA-Z0-9_\-
\.]*)@(\[((25[0-5]|2[0-4][0-9]|1[0-9][0-9]|[1-9][0-9]|[0-9])\.){3}|((([a-zA-Z0-9\-]+)\.)+))([a-zA-
Z]{2,}|(25[0-5]|2[0-4][0-9]|1[0-9][0-9]|[1-9][0-9]|[0-9])\])$"))
         {
            MessageBox.Show("Email invalido");
         }

         // Despues de validar
         else if (TxtUserName.Text != "" && TxtPwd.Password != "" && TxtMovil.Text != "" &&
TxtEmail.Text != "")
         {
            try
            {
               List<Usuario> list = new List<Usuario>();
               ServicioGPS.ServiceTFGClient proxy = new ServicioGPS.ServiceTFGClient();
```

```
                proxy.InsertarUsuarioAsync(TxtUserName.Text, TxtPwd.Password, TxtMovil.Text,
TxtEmail.Text);
            }
            catch
            {
                MessageBox.Show("No se ha registrado");

            }

        }
    }

    private void insertar()
    {
        Usuario ObjUserData = new Usuario();
        ObjUserData.Nombre = TxtUserName.Text;
        ObjUserData.Password = TxtPwd.Password;
        ObjUserData.Telefono = TxtMovil.Text;
        ObjUserData.Email = TxtEmail.Text;
        int Temp = 0;
        foreach (var UserLogin in ObjUserDataList)
        {
            if (ObjUserData.Nombre == UserLogin.Nombre)
            {
                Temp = 1;
            }
        }
        // Compruebo si existe usuario en la base de datos
        if (Temp == 0)
        {
            ObjUserDataList.Add(ObjUserData);
            if (ISOFile.FileExists("RegistrationDetails"))
            {
                ISOFile.DeleteFile("RegistrationDetails");
            }
            using (IsolatedStorageFileStream fileStream =
ISOFile.OpenFile("RegistrationDetails", FileMode.Create))
            {
                DataContractSerializer serializer = new
DataContractSerializer(typeof(List<Usuario>));

                try
                {
                    serializer.WriteObject(fileStream, ObjUserDataList);
                    MessageBox.Show("Se ha guardado los datos en el fichero.");
                    Usuario us = new Usuario();

                    foreach (var User in ObjUserDataList)
                    {
                        us.Nombre = User.Nombre;
                        us.Password = User.Password;
                        us.Telefono = User.Telefono;
                        us.Email = User.Email;
                    }

                }
                catch
                {
                    throw new SerializationException();
                }
```

```
            }
        }
        else
        {
            MessageBox.Show("El nombre de usuario ya existe.!");
        }
    }

    private void Txt_GotFocus(object sender, RoutedEventArgs e)
    {
        TextBox tb = (TextBox)sender;
        tb.BorderBrush = new SolidColorBrush(Colors.LightGray);
    }
    private void pwd_GotFocus(object sender, RoutedEventArgs e)
    {
        PasswordBox pbox = (PasswordBox)sender;
        pbox.BorderBrush = new SolidColorBrush(Colors.LightGray);
    }
  }
}
```

8.5. – Código del archivo DetalleUsuario.xaml.cs.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Navigation;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;
using AppTFG.ServicioGPS;

namespace AppTFG
{
  public partial class DetalleUsuario : PhoneApplicationPage
  {
    ServicioGPS.ServiceTFGClient proxy = new ServicioGPS.ServiceTFGClient();
    string nombre = "", idUsuario = "";
    public DetalleUsuario()
    {
        InitializeComponent();

    }
    public DetalleUsuario(string s)
    {
        InitializeComponent();
        ServicioGPS.ServiceTFGClient svc = new ServicioGPS.ServiceTFGClient();
        svc.BuscarUsuarioCompleted += new
EventHandler<BuscarUsuarioCompletedEventArgs>(buscar);
        svc.BuscarUsuarioAsync(s);
    }
    protected override void
OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e)
    {
        base.OnNavigatedTo(e);

        string parametro = string.Empty;
```

```
            if (NavigationContext.QueryString.TryGetValue("parametro", out parametro))
            {
                string[] param = parametro.Split(':');
                nombre = param[0];
                idUsuario = param[1];
            }

            ServicioGPS.ServiceTFGClient proxy = new ServicioGPS.ServiceTFGClient();
            proxy.BuscarUsuarioCompleted += new
EventHandler<BuscarUsuarioCompletedEventArgs>(buscar);
            proxy.BuscarUsuarioAsync(nombre);
        }
        private void buscar(object sender, BuscarUsuarioCompletedEventArgs e)
        {
            lstUsuario.ItemsSource = e.Result;
        }

        private void listado(object sender, ListaUsuariosCompletedEventArgs e)
        {
            lstUsuario.ItemsSource = e.Result;
        }

        private void LogOut_Click(object sender, RoutedEventArgs e)
        {
            SignOut();
        }

        private void ImgSignOut_Tap(object sender, System.Windows.Input.GestureEventArgs e)
        {
            SignOut();
        }
        public void SignOut()
        {
            throw new NotImplementedException();
        }
        private void ImgUserProfile_Tap(object sender, System.Windows.Input.GestureEventArgs e)
        {
            StckUserDetailsUI.Visibility = Visibility.Visible;
        }

        private void Links_Tap(object sender, System.Windows.Input.GestureEventArgs e)
        {
            StckUserDetailsUI.Visibility = Visibility.Collapsed;
        }

        private void ir_inicio(object sender, EventArgs e)
        {
            throw new NotImplementedException();
        }

        private void Localizar(object sender, RoutedEventArgs e)
        {
            NavigationService.Navigate(new Uri("/Page1.xaml?parametro=" + nombre + ":" +
idUsuario, UriKind.Relative));
        }
    }
}
```

8.6. – Código del archivo CoordinateConverter.cs.

```
using System;
using System.Collections.Generic;
using System.Device.Location;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Windows.Devices.Geolocation;

namespace AppTFG
{
    static class CoordinateConverter
    {
        public static GeoCoordinate ConvertGeocoordinate(Geocoordinate geocoordinate)
        {
            return new GeoCoordinate
                (
                geocoordinate.Latitude,
                geocoordinate.Longitude,
                (double)geocoordinate.Altitude,
                geocoordinate.Accuracy,
                geocoordinate.AltitudeAccuracy ?? Double.NaN,
                geocoordinate.Speed ?? Double.NaN,
                geocoordinate.Heading ?? Double.NaN
               );
        }
    }
}
```

Parte III.

9. - Bibliografía.

- Libro de C#:

  Microsoft visual C# 2012 step by step.

- Para el diseño de la aplicación:

  http://appdesignbook.com/es/contenidos/diseno-visual-apps-nativas/

- Para probar aplicaciones que utilizan datos de localización:

  https://msdn.microsoft.com/en-us/library/windows/apps/hh202933(v=vs.105).aspx

- Mapas y Ubicacion Windows phone 8:

  https://juliofarfan.wordpress.com/2013/01/15/mapas-y-ubicacion-en-windows-phone-8/

- Aplicación de las Guidelines de Windows Phone, consejos de WPDPS para Windows Phone:

  https://blogs.msdn.microsoft.com/africaapps/2014/03/08/uxui-guidelines-for-windows-phone-8/

- Como obtener la ubicación del usuario:

  https://msdn.microsoft.com/windows/uwp/maps-and-location/get-location

- Probar la aplicación Windows Phone 8 incluye un simulador de sensor de ubicación:

  https://msdn.microsoft.com/en-us/library/windows/apps/hh202933(v=vs.105).aspx

- Cómo obtener la ubicación actual del teléfono para Windows Phone 8:

  https://msdn.microsoft.com/en-us/library/windows/apps/jj206956(v=vs.105).aspx

- Seguimiento continuo de la ubicación de un teléfono para Windows Phone 8:

  https://msdn.microsoft.com/es-es/library/windows/apps/jj247548(v=vs.105).aspx

- Consultar e insertar datos Azure Mobile service en UWP

  https://www.youtube.com/watch?v=smfdSAHGAEI#t=123.133618

- Metodología de desarrollo de software
  https://es.wikipedia.org/wiki/Metodolog%C3%ADa_de_desarrollo_de_software

10. – Acrónimos.

Servidor remoto: Es una combinación de hardware y software que permite el acceso remoto a herramientas o información que residen en una red de dispositivos. Está instalado en otro equipo, en él se puedes almacenar un sitio web, o almacenar datos en una base de datos.

Modelo relacional: En este modelo todos los datos son almacenados en relaciones, y como cada relación es un conjunto de datos, el orden en el que estos se almacenen no tiene relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar por un usuario no experto. La información puede ser recuperada o almacenada por medio de consultas que ofrecen una amplia flexibilidad y poder para administrar la información.

Referencia de servicio: Una referencia de servicio habilita a un proyecto para obtener acceso a uno o más servicios de Windows Communication Foundation (WCF).

Interfaz de usuario: La interfaz de usuario es el medio con que el usuario puede comunicarse con una máquina, equipo, computadora o dispositivo, y comprende todos los puntos de contacto entre el usuario y el equipo.

Autenticación: Procedimiento informático que permite asegurar que un usuario de un sitio web u otro servicio similar es auténtico o quien dice ser.

Implementación: Es la realización de una especificación técnica o algoritmos como un programa, componente software, u otro sistema de cómputo. Muchas implementaciones son dadas según a una especificación o un estándar.

Emulador: Es un software que permite ejecutar programas o videojuegos en una plataforma. A diferencia de un simulador, que solo trata de reproducir el comportamiento del programa, un emulador trata de modelar de forma precisa el dispositivo de manera que este funcione como si estuviese siendo usado en el aparato original.

Guidelines: Un conjunto de recomendaciones destinadas a mejorar la experiencia para los usuarios, haciendo interfaces de uso más intuitivos, aprendibles, y constantes. La mayoría de las guías se limitan a definir una apariencia común para las aplicaciones, y se centran en los usos en un ambiente de escritorio particular.

API: Es un conjunto de funciones, métodos o procedimientos que ofrece una biblioteca para ser utilizado por otro software como una capa de abstracción.

Smartphone: Es un tipo de teléfono móvil construido sobre una plataforma informática móvil, con mayor capacidad de almacenar datos y realizar actividades, semejante a la de una minicomputadora, y con una mayor conectividad que un teléfono móvil convencional.

11. – Índice figuras.