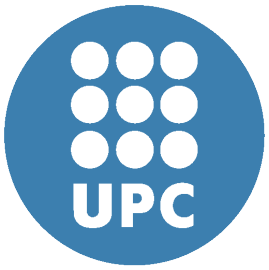


Aplicación de la realidad aumentada para un sistema de entrenamiento

Ingeniería Técnica Informática de Gestión, Junio 2011

Proyectista: Carlos Machado Marcos
Director: Pau Fonseca i Casas

Índice

Anexos

1. Introducción general i objetivo

Desde tiempos inmemoriales el ser humano ha hecho uso de la tecnología para hacerse la vida más fácil y cómoda. El descubrimiento de nuevos conocimientos ha permitido crear una serie de elementos de carácter tecnológico y a su vez gracias a la tecnología se han podido realizar estudios para avances científicos.

La tecnología avanza cada día más rápido, con la aparición de Internet en el año 1969 ha surgido un nuevo mundo de recursos y ventajas. En la actualidad "cualquier" persona puede acceder a una gran cantidad de información de casi todos los temas y disciplinas sin moverse de casa.

Unos años más tarde, en el 1983, sin saber que más adelante se convertiría en una auténtica revolución, aparece el teléfono móvil. Este dispositivo que previamente se inventó solo con la intención de hacer y recibir llamadas, en la actualidad con la integración de Internet, ya sea por WIFI o por 3G, el hecho de hacer llamadas ha pasado a un segundo plano y las verdaderas protagonistas son la inmensa cantidad de aplicaciones que existen para éstos, que mediante Internet y otros componentes ofrecen casi las mismas prestaciones que un ordenador de sobremesa.

Pero la tecnología va más allá y cada vez se orienta más a dar información al usuario sea cual sea su ubicación y a representar situaciones virtuales en la propia realidad.

El objetivo de este proyecto es la de implementar el software necesario para que un sistema de realidad aumentada pueda mostrar información de un sistema de entrenamiento. Se quiere crear un sistema que permita representar, a través de la realidad aumentada, la información que se obtiene de una traza de un modelo de simulación.

Se decidió realizar este proyecto porque hoy en día la realidad aumentada ofrece muchas posibilidades y es una tecnología muy potente que aún no está explotada. Con este proyecto se aprenderán nuevos conocimientos y se instalaran las bases para poder construir otros sistemas similares al de este proyecto que hagan uso de la realidad aumentada.

2. Realidad Aumentada (RA)

La realidad aumentada, como su propio nombre sugiere, es una tecnología que mezcla la propia realidad física con elementos virtuales, obteniendo como resultado una realidad mixta a tiempo real. No se tiene que confundir con el término realidad virtual, ya que en ésta no interviene la realidad física, es decir, la realidad aumentada sobrepone los elementos virtuales en la propia realidad física, y en el caso de la realidad virtual todos sus elementos son ficticios.

2.1. Elementos necesarios

Para poder hacer uso de la realidad aumentada se necesitan una serie de elementos y dispositivos. Los imprescindibles son los siguientes:

- **Monitor**: es el elemento básico dónde se muestra la mezcla de realidad y componentes virtuales. Hay distintos tipos de monitores que van desde los más futuristas (Img. 1) a los más convencionales. Se dará una explicación más detallada sobre técnicas y métodos de visualización en un apartado posterior. (Ver 2.2)

Img. 1: Dispositivo “headset” para realidad aumentada

- **Cámara**: dispositivo que toma la información del mundo real y la transmite al monitor. En el mayor número de casos, los elementos virtuales dependen de los elementos físicos, es por eso que se usaran cámaras web (Img. 2) para poder transmitir la imagen al monitor. En el resto de casos, será la propia vista la que jugará este papel.

Img. 2: Cámara web convencional

- **Software**: programa que toma los datos reales y los transforma en realidad aumentada. Hay distintos programas que permiten trabajar con realidad aumentada, uno de ellos muy conocido es Flash. Otros software disponibles son:

    - **ARviewer**[1]**:** editor de libre distribución que permite crear aplicaciones fácilmente integrables en dispositivos con sistema operativo Android

    - **ARToolkit**[2]**:** librería muy extensa de software para desarrollar aplicaciones de realidad aumentada.

    - **D.A.R.T**[3]**:** sistema de programación que ayuda a añadir a los software la funcionalidad de visualizar la mezcla de objetos reales y virtuales. Además incorpora librerías de acciones ante estímulos que permiten seguir a objetos con un marcador en un video y reaccionar con información a tiempo real según sea su marcador.

[1] http://www.androidzoom.com/android_applications/social/arviewer-10_syge.html

[2] http://www.hitl.washington.edu/artoolkit/download/

[3] http://www.cc.gatech.edu/dart/download.htm

- **Marcadores**: los marcadores básicamente son imágenes de símbolos que el software interpreta y de acuerdo a un marcador específico realiza una respuesta específica. Es un recurso muy utilizado y que se está imponiendo actualmente en la publicidad, de lo cual se hablará más adelante (Ver 2.3.2)

Hay diversos tipos de marcadores, los más comunes son los llamados marcadores de códigos matriciales (Img. 3). Este tipo de códigos no fueron diseñados para el uso en la realidad aumentada, si no que fueron inventados para ser el equivalente al código de barras. Se decidió crear, porque el código de barras al ser leído genera una serie de números que se buscan en una base de datos para relacionarlo con el producto. Pero el problema reside en que hay muchos productos y sus números pueden repetirse de código mundialmente. Por esto, en el 1994 en se inventaron los códigos matriciales o código QR (Quick Response).

Img. 3: Clásico marcador del tipo QR

Aunque no todos los marcadores tienen esta apariencia. Como los marcadores los lee el software (a través de la imagen que ve la webcam) pueden ser de cualquier tipo, como por ejemplo cromos.

Img. 4: Cromo de realidad aumentada, donde el propio cromo hace el papel de marcador

2.2. Métodos de visualización

A pesar de ser conceptos diferentes, en cuanto a métodos de visualización la realidad aumentada y la virtual se parecen bastante. Ambas necesitan un dispositivo para poder visualizar los elementos virtuales. En el caso de la RA, el monitor sería el dispositivo de uso más frecuente para poder mezclar realidad y ficción. Aunque cada día la lista es más larga incluyéndose muchos más dispositivos como pueden ser cascos (headsets), gafas, móviles (smartphones) e incluso displays espaciales. Sea cual sea el dispositivo de visualización, todos se rigen por el mismo esquema, el esquema de la RA:

Img. 5: Esquema sencillo del concepto realidad aumentada

A continuación se explican con más detalle algunos de estos dispositivos:

- **Headsets:** los headsets son cascos o gafas (HMD Head-Mounted Display) compuestos de pantallas de cristal líquido que se colocan en frente de los ojos para que el software plasme en ellas las imágenes virtuales. Estas pantallas no son opacas, ya que deben dejar que la vista pueda apreciar la realidad física y a su vez que los elementos virtuales se puedan reproducir y verse como si estuvieran sobrepuestos. Algún ejemplo de headset de tipo casco y gafas podrían ser estos:

Img. 6: Distintos dispositivos "headset". Pueden ser más sofisticados como el primer (izquierda) hasta de carácter más artesanal como el que se muestra en la última imagen (derecha)

- **Smartphones:** Otra manera de poder experimentar la realidad aumentada es a través de los móviles de tipo Smartphone, ya que éstos poseen cámara, una pantalla y la posibilidad de descargar e instalar aplicaciones para el funcionamiento de la RA. Aunque como se dijo en la introducción, la RA se está orientando mucho a proporcionar información en tiempo real al usuario. Este tipo de información suele ser acerca del lugar donde se encuentra, es por esto que además de los elementos mencionados anteriormente, también necesitará:

    - **GPS:** Sera necesario que el dispositivo móvil tenga un gps para poder localizar al usuario y más importante aún poder cargar la información vía Internet del lugar donde se encuentra.

    - **Brújula:** Además de localizar el dispositivo también se necesita saber la orientación de éste. Esto se puede realizar gracias a una brújula digital, su funcionamiento consiste en la de un pequeño chip que permite mediante una calibración inicial detectar la posición que ocupa el móvil con respecto al campo magnético terrestre.

    - **Acelerómetro:** Es un dispositivo que se usa para medir aceleraciones, en el caso de un móvil se usa para saber cuál es la posición de éste respecto a los ejes X, Y, Z. Los móviles que tienen incorporado un acelerómetro, suelen tener un nivel (Img. 7) en cada uno de los ejes, parecido a los que se usan en el ámbito de la construcción para saber si una pared esta recta, donde la burbuja de aire en este caso es de gas y mediante un sensor se sabe cuál es su posición.

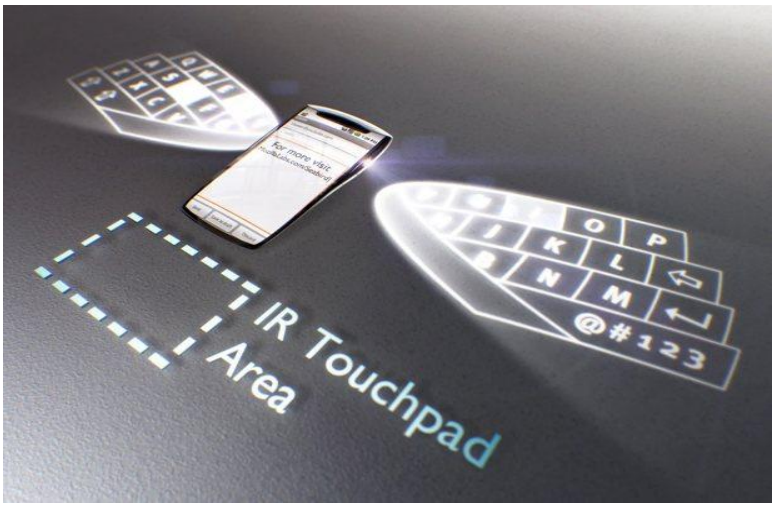
Img. 7: Nivel de burbuja para saber la posición del dispositivo, en este caso del eje X.

- **Displays espaciales:** La Realidad Aumentada Espacial (SAR) hace uso de proyectores digitales para mostrar información gráfica sobre los objetos físicos. La principal ventaja es que el display no está asociado a ningún usuario y así pueden hacer uso todos a la vez y trabajar en grupo, y también que no depende de la resolución de pantalla donde vaya a reproducirse. Además SAR permite al usuario no tener la necesidad de llevar equipo encima como los anteriormente mencionados, ya que gracias al proyector, la imagen virtual se puede reflejar en una pared, una mesa, o incluso en la propia piel.

A continuación se muestran y explican una serie de proyectos y aplicaciones que trabajan con la tecnología SAR como máxima protagonista.

Seabird[4]

Este es el nombre del prototipo de móvil de la compañía Mozilla. En este móvil las pantallas táctiles son secundarias frente a la tecnología SAR que emplea. El Seabird, a parte de una cámara de 8 mpx también poseerá dos pico-proyectores de 45 lúmenes y resolución de 960 x 600 píxeles uno a cada lado con los que se reflejará un teclado con el que se podrá escribir como si de un teclado normal se tratará, y en la parte inferior del móvil aparece un touchpad virtual.



Img. 8: Prototipo Seabird proyectando un teclado QWERTY y un touchpadh

[4] Este móvil fue diseñado por **Billy May**, el cual trabaja en un en un proyecto dentro de la organización llamado *Open Web Concept Phone*. El sistema operativo del seabird se basaría en Andorid. Cabe recordad, que este diseño es solo un prototipo, una idea, de cómo Mozilla Labs ve los teléfonos móviles en el futuro.

Mozilla también propone poder navegar por internet sin la incomodidad de hacerlo en una pantalla pequeña, pues los dos proyectores, aparte de reflejar un teclado también pueden reflejar el contenido de la pantalla.

Img. 9: Modalidad de proyección de pantalla y teclado.

Además de estas novedades, incluirá un manos libres Bluetooth que también se podrá usar como mouse para controlar el móvil.

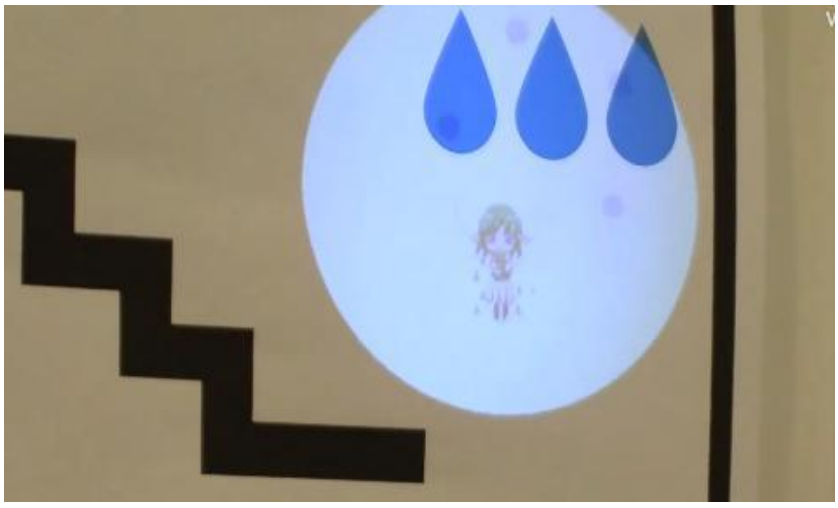
Img. 10: Manos libres del Seabird con capacidad de ser utilizando como ratón.

Twinkle

Es una interfaz diseñada por investigadores de Tokyo y Keio de realidad aumentada que usa un proyector (Img. 11), a modo de linterna, y una cámara. Su funcionamiento es simple, mediante el proyector reflejan una imagen de un personaje en alguna superficie y la cámara detecta la posición de éste y los obstáculos que tiene a su alrededor. Este tipo de obstáculos pueden ser objetos dibujados en la superficie o incluso sombras reflejadas, ya que también distingue proximidad.

Img. 11: Proyector con cámara incorporada para captar las imágenes de éste.

La imagen, una vez proyectada puede interactuar con los objetos o las sombras que haya en la superficie tal como esté programado. La cámara detecta que tipo de objeto es y el software se encarga de hacer que el personaje se mueva en consecuencia.



Img. 12: El objeto interactúa con las gotas de agua y se moja.

Skinput

Con la colaboración del investigador Chris Harrison de la Universidad de Carneige Mellon y Dan Morris y Desney Tan del laboratorio de Microsoft Research de la ciudad de Redmon se presentó el proyecto de realidad aumentada skinput, el cual trata sobre un sistema en el que el cuerpo humano se convierte en pantalla táctil.

A diferencia de los otros dispositivos, la idea de éste se basa en unos sensores acústicos (Img. 13) que captan los sonidos de baja frecuencia cuando hay contactos con la piel. Luego, según unos patrones el software sabe dónde se ha tocado y la acción que tiene que realizar.

Img. 13: Brazalete con sensores acústico

El sistema es capaz de detectar hasta 5 interacciones próximas entre sí con una precisión del 95%, incluso estando el sujeto en movimiento. El dispositivo tiene forma de brazalete el cual colocado en el bíceps proyecta los elementos virtuales en el antebrazo. En este caso, no se necesita cámara para captar la interacción del usuario ya que este brazalete cuenta con los sensores. Pero sí que se necesita enviar los datos recibidos a un ordenador o móvil que tenga la "base de datos" de tipos de toques con la piel. Esto se realiza gracias a la conexión Bluetooth de la que dispone el dispositivo.

Img. 14: Proyección de un menú en el antebrazo

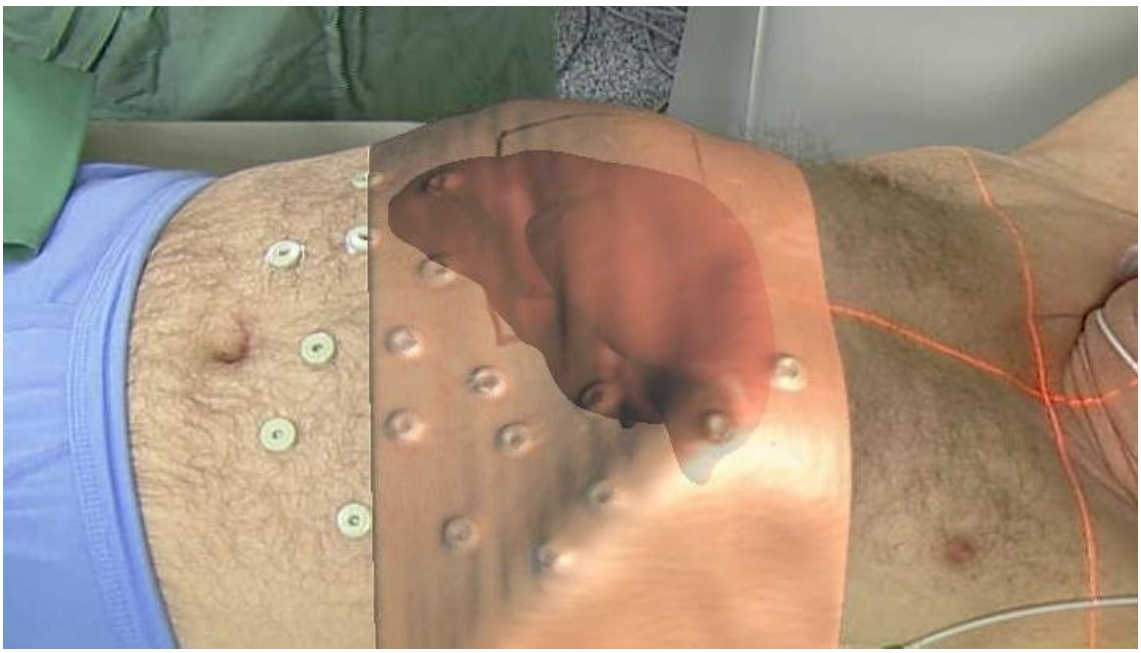
2.3. Aplicaciones de la RA

Después de la invención de una tecnología surgen muchas aplicaciones para darle uso. El caso más actual es el Iphone, el cual aparte de tener la funcionalidad de teléfono, gracias a sus prestaciones (conexión a Internet, GPS, Brújula, pantalla multitouch, cámara, etc.) dispone de centenares de aplicaciones que hacen uso de ellas. Aplicaciones de ocio, entretenimiento, navegación, calendarios, juegos, etc.

En el caso de la realidad aumentada sucede lo mismo, existen muchas aplicaciones de distintos ámbitos. A continuación se describen algunas de ellas:

2.3.1 Cirugía

En el ámbito de la cirugía la realidad aumentada resulta de lo más útil ya que en el mismo paciente se podría tener información como por ejemplo su tensión arterial, sus pulsaciones y muchos más datos sin levantar la vista a un monitor. Además, también se podría obtener información sobre qué zona es mejor para realizar una incisión, o delimitar los bordes limpios de un tumor invisibles a simple vista, o incluso sobreponer una radiografía. Con todo esto, se minimizaría mucho el impacto de la cirugía en el paciente (Img. 15). Además en ámbitos educativos los alumnos podrán ver desde todos los ángulos el funcionamiento de un cuerpo humano pudiéndolo incluso atravesar.



Img. 15: Operación donde se sobrepone a la piel del paciente sus órganos internos

2.3.2 Publicidad

Cada día son más las empresas que hacen uso de la realidad aumentada para anunciar sus productos. Es mucho más visual y entendedor para un consumidor poder ver lo que desea comprar si lo ve desde cualquier perspectiva y en 3D. Ofreciendo además la posibilidad de interactuar con el objeto. Este tipo de publicidad, además de ser más atractiva, despierta más interés y curiosidad en el usuario, y es más probable que éste se interese en el producto y lo compre.

Por ejemplo la marca italiana de automóviles FIAT, para anunciar su Fiat 500 propuso a los usuarios crear su propio anuncio con el que solo necesitaban una webcam e imprimirse el marcador de FIAT. Enfocando la webcam al marcador aparecía en el monitor el automóvil y al mismo tiempo el usuario.

Otro ejemplo es el de la revista Esquire, en la que se publican muchos códigos QR que se podían visualizar con la ayuda de una webcam y el software adecuado, pudiendo así ver los productos publicitados.

Y por último, la empresa danesa de cerveza Carlberg, para mostrar su nueva botella, también hizo uso de un marcador (Img. 16) que gracias al software descargable de su página web podías visualizar.

Img. 16: Marcador de la nueva botella de Calsberg

2.3.3. Redes Sociales

La RA también se refleja en el impacto que han tenido las redes sociales. Gracias al elevado número de aplicaciones que existen para dispositivos móviles y gracias también a la geolocalización, surgen muchas más aplicaciones que usan la realidad aumentada para proporcionar información de contactos, lugares, etc. Algunas de ellas son:

AugmentedID es una aplicación en la que el usuario que la utiliza debe hacerse una foto para que sea cargada en la base de datos del programa. Luego tiene que decidir qué información (perfil, facebook, twitter,…) desea compartir con el resto de usuarios. A partir de ese momento, cualquier persona que disponga también del software solo tendrá que enfocarlo con la cámara de su dispositivo móvil para poder acceder a su información.

Img. 17: Al enfocar a una persona registrada, se muestran sus participaciones en redes sociales

TwittARound: Twitter es una red social que tiene alrededor de 100 millones de usuarios en el mundo, y la realidad aumentada no ha pasado por alto esa cifra. TwittAround es una aplicación que sobrepone en la pantalla los twits que han sido publicados en la dirección donde enfoca la cámara del dispositivo. Para ello usa el acelerómetro, la brújula y el GPS (también Internet). Es una aplicación que según muchos usuarios no tiene mucha utilidad, pues no consigues nada sabiendo la posición de los usuarios que publican twits. Pero está claro que es otro ejemplo de lo que realidad aumentada puede hacer.

Hay otras aplicaciones que usan el mismo método pero para usos más útiles. **Layar** es una de ellas, y al igual que TwittARound hace uso del GPS para localizar el dispositivo, de la brújula para orientarse y del acelerómetro para saber el ángulo con respecto al suelo. Ésta aplicación da información de edificios, restaurantes, farmacias, paradas de autobús, etc. así como de su localización. También tiene la opción de poder llevar al usuario al punto de interés escogido. Además permite añadir nuevas[5] capas como Wikipedia o Tweetmondo, que tiene la misma función que TwittAround. Es una gran aplicación y sobre todo de gran utilidad ya que en un futuro se podrán añadir nuevas capas con zonas de interés para los usuarios.

Otra aplicación muy parecida a Layar es Wikitude, tiene más o menos las mismas funcionalidades pero con otra interfaz y diferentes capas. Con solo un vistazo a la pantalla del dispositivo, se pueden, por ejemplo, localizar hoteles y hostales y saber el número de habitaciones disponibles.

[5] Para la adición de nuevas capas se tiene que esperar a que salgan actualizaciones de la versión de la aplicación. No se pueden conseguir de forma individual.

2.3.4. Simulación y entrenamiento

El concepto de simulación es muy amplio y existen muchas definiciones para este término, aunque R.E. Shannon en su libro *IEEE Transactions on Systems, Man and Cybernetics*[6] da una definición muy completa, la describe como "*el proceso de diseñar un modelo de un sistema real y llevar a término experiencias con él, con la finalidad de comprender el comportamiento del sistema o evaluar nuevas estrategias -dentro de los límites impuestos por un cierto criterio o un conjunto de ellos - para el funcionamiento del sistema*".

Para diseñar el modelo del sistema real, la simulación también hace uso de la realidad aumentada. Gracias a ella el modelo se acerca aún más a la realidad ya que ésta interviene en la simulación del proceso, y además ofrece la posibilidad de que el ser humano pueda interactuar también. Esto es una gran ventaja puesto que en un modelo de simulación lo que más dificultad conlleva es simular el comportamiento de las personas, no pudiendo llegar nunca a una aproximación completa a la realidad.

Existen infinidad de situaciones que se pueden simular, pero una de las más interesantes es la de sistemas de entrenamiento. Es muy importante el hecho poder practicar antes de enfrentarse a situaciones reales y es aquí donde la realidad aumentada juega un papel muy importante. Actualmente muchos de los simuladores de entrenamiento hacen uso de la realidad virtual, lo cual está bien ya que recrean de manera muy precisa la realidad y los posibles problemas que pueda haber. El problema irradia en que recrean la realidad, nunca trabajan con ella directamente, cosa que la realidad aumentada sí que hace.

Algún ejemplo donde el entrenamiento con realidad aumentada aportaría mucho y sería de gran ayuda sería en la policía. Ya que actualmente disponen de campos de tiro y circuitos de entrenamiento donde para practicar disparan a blancos inanimados, sin la posibilidad de que éstos disparen contra ellos. En cambio con realidad aumentada, podrían entrenan contra personas virtuales que tuvieran inteligencia artificial y pudieran interactuar con el entorno e incluso disparar contra los agentes (de forma virtual).

[6] Shannon, Robert; Johannes, James D. (1976). «Systems simulation: the art and science». *IEEE Transactions on Systems, Man and Cybernetics* **6(10)**. pp. 723-724.

Otro ejemplo es la aplicación de la realidad aumentada al entrenamiento de los bomberos. Siempre se ha asociado a los bomberos con la función de apagar fuegos, aunque tengan muchos otros tipos de trabajos. Uno de estas tareas, por ejemplo, es la de rescatar personas de estructuras, para esta labor pueden entrenarse en cualquier tipo de estructura que incluso ellos mismos pueden escoger. Pero a la hora de apagar un incendio, solo pueden practicar de manera casi exacta a la realidad con fuegos en edificios o casas, pero ¿qué sucede cuando el incendio es en un bosque? No pueden escoger parcelas de bosque y prenderles fuego para luego apagarlo y así entrenar. Es aquí donde entra la realidad aumentada, pudiendo conseguir un sistema para poder incendiar una parcela de bosque real con llamas virtuales.

3. Sistema operativo (SO) y dispositivos

Tuvieron que pasar 37 años desde la aparición del primer computador, el ENIAC, para que un ordenador contara con un sistema operativo. Éste vino de la mano de Microsoft con Windows 1.0 en el 1983, el cual era distribuido en 5 diskettes de 5¼ de 360 KB cada uno.

Por otro lado, en el 1983, Richard Stallman creó el proyecto GNU con el objetivo de desarrollar un sistema operativo libre, ya que Windows se tenía que comprar. Fue así como nació el sistema operativo Linux y sus múltiples distribuciones: Ubuntu, Kubuntu, Debian, Suse, etc.

Por último, Steve Jobs y Steve Wozniak con la ayuda económica de Mike Markkula fundaron en el 1 de abril de 1976 Apple Computer, teniendo como máximo protagonista al Apple I, una computadora por la que Jobs tuvo que vender su camioneta y Wozniak su calculadora HP para crear el prototipo.

Estos tres sistemas operativos son los que actualmente rivalizan por ganar usuarios de ordenadores personales y portátiles. ¿Pero qué sucede con los móviles? Desde hace unos años, todos los móviles disponen de un sistema operativo, y al igual que con los ordenadores se pelean entre ellos para tener la mayor cuota de mercado, actualmente en EE UU Android lidera la lista de SOs con un 31,2 %, le sigue RIM de Blackberry con un 30,4%, luego Apple con un 24,7% y finalmente Microsoft con su Windows Phone y Symbian con un 8% y 3,2% respectivamente. A continuación se hará una pequeña explicación de cada uno de estos SO:

3.1 Android

Android fue desarrollado por Android Inc, una empresa que en el año 2005 compró Google. Este sistema operativo es el producto estrella de la Open Handset Alliance, un conjunto de desarrolladores de software y hardware, así como también de fabricantes y operadores de servicio. Tiene un núcleo Linux, lo que dio un soplo de aire fresco a la industria de la telefonía móvil y el hecho de que sea abierto fue una de las claves de su éxito, aunque no es completamente libre. Es la primera vez que Android se convierte en el SO más usado en smartphones en Estado Unidos. Un dato curioso es que una gran parte de la gente desconoce el sistema operativo Linux, y un tercio de ésta lo usa diariamente en su dispositivo Android.

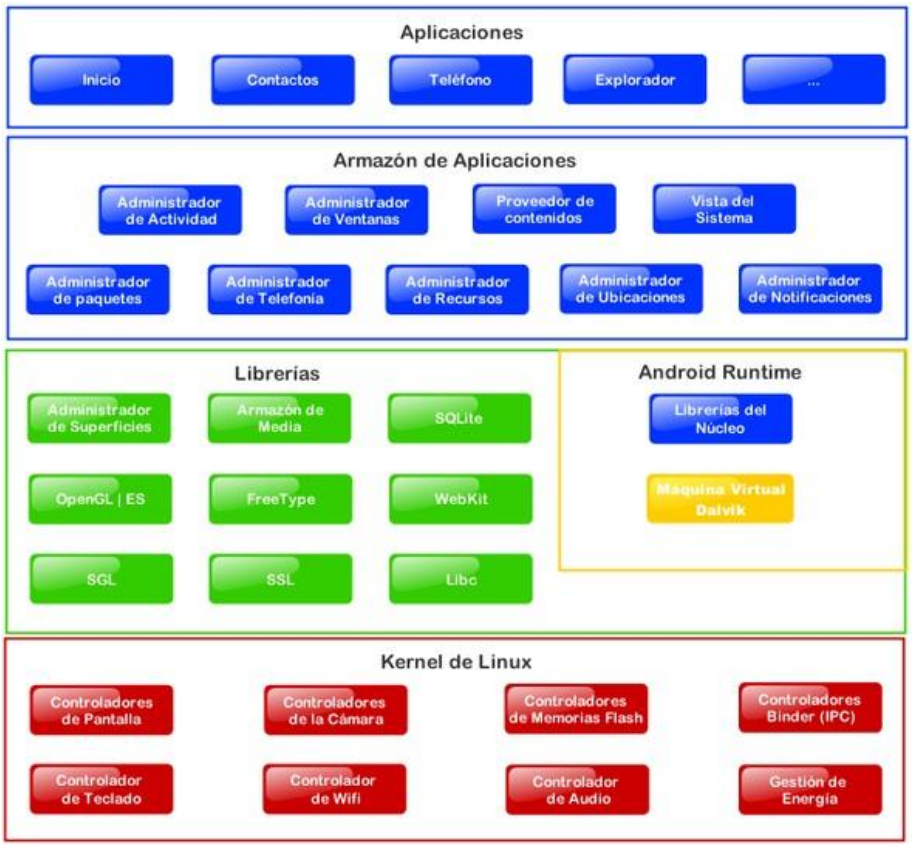
Además posee una gran comunidad de desarrolladores de aplicaciones, actualmente el número está sobre las 200.000. Al ser open source, Android tiene una página web donde se puede descargar el kit de desarrollo de software Android [http://developer.android.com/sdk/index.html]. Android Market es la tienda administrada por Google responsable de distribuirlas en línea, aunque también se puede obtener el software de manera externa. Todos estos programas tienen en común a Java como su lenguaje de programación, al cual se le puede sacar mucho partido gracias a los procesadores de 1 GHz como el del Sony Ericcson Xperia X10 o el de la HTC Desire.

Img. 19: Sony Ericsson Xperia X10

Img. 18: HTC Desire

La arquitectura de Android se compone por diversos elementos, los más importantes son los que se muestran a continuación:

- Las **aplicaciones** base de cualquier dispositivo con SO Android incluyen un cliente de correo electrónico, capacidad de enviar SMS, calendario, agenda de contactos, mapas, etc.
- La arquitectura del **armazón de aplicaciones** está diseñada para simplificar la reutilización de componentes, es decir, una aplicación puede mostrar sus capacidades y otra luego poder usarlas. El usuario puede reemplazar componentes gracias a este mecanismo.
- En cuando a las **librerías**, Android incluye un conjunto de éstas de C y C++. Algunas de estas librerías son SQLite, librerías para el uso del 3D así como de gráficos, etc.
- **Runtime de Android**: cada aplicación Android tiene su propia instancia en la máquina virtual Dalvik, es decir, que cada una corre su propio proceso. Dalvik, compuesto de registros y con clases compiladas en Java, está escrito de tal forma que optimiza el uso de memoria mínima.
- Por último, como se ha mencionado anteriormente, los sistemas Android poseen en su interior un **núcleo Linux**, que se encarga de los servicios bases de seguridad, gestión de memoria y de procesos, pila de red, etc. Además el núcleo actúa de capa de abstracción entre el hardware y el resto de software.



Img. 20: Arquitectura de un SO basdo en Android

3.2. RIM Blackberry

RIM (Research In Motion) es una compañía canadiense fundada en 1984 de telecomunicaciones más conocida por ser la creadora de los dispositivos Blackberry. Antes de embarcarse en este proyecto, RIM trabajaba con la compañía Ericsson y con memoria RAM de datos móviles para competir contra el SkyTel de Motorola.

Los dispositivos Blackberry cuentan con un sistema operativo que trabaja en un entorno multitarea. El SO está diseñado para dar soporte a Java MIDP 1.0 y WAP 1.2. El actual SO, el 5.0, proporciona un subconjunto de MIDP 2.0 y permite la sincronización con Microsoft Exchange Server para el correo electrónico, calendario, citas, tareas, etc y además añade soporte para Lotus Notes y Novell GroupWise. Está todo escrito en Java, así pues se pueden crear y ejecutar aplicaciones con el estándar J2ME de java.

En cuanto a la CPU, los primeros dispositivos Blackberry utilizaban un chip Intel 80386. La serie 8000 de Blackberry inlcuido el Pearl ya utilizaban un ARM XScale, ARMv5TE y PXA 900 de 312 MHz, a excepción del 8707 que usaba un chip Qualcomm 3250 ya que el PXA 900 no soportaba la tecnología 3G. En Mayo de 2008, la Research In Motion introdujo en las series 9000 de Blackberry un XScale de 624 HMz. En la actualidad, los smartphones de Blackberry más comprados son el modelo Storm (Img. 21) con una CPU Qualcomm a 528 MHz igual que el modelo Curve 8530 con otro Qualcomm, el MSM7600, el modelo Bold 9000 (Img. 22) con un procesador algo superior, un Intel XScale PXA270 con 624MHz. Y finalmente el modelo no tan visto, el Torch (9800) que aunque es de una gama más alta que el Bold, conserva el mismo procesador a 624 MHz.

Img. 21: Blackberry Storm

Img. 22: Blackberry Bold

3.3. Apple

Apple Inc., con su sede en California, siempre se ha caracterizado por tener un software y hardware muy novedosos y con las últimas tecnologías, así como también por sorprender con unos diseños inverosímiles. Un buen ejemplo de esto es uno de sus nuevos productos, el MacBook Air, un finísimo ordenador portátil, pero con unas prestaciones muy potentes.

Apple no quiso quedarse fuera del mercado de la telefonía móvil, esperándose así hasta la aparición de los teléfonos inteligentes, para lanzar al mercado lo que aún a día de hoy es una auténtica revolución, el iPhone. Este dispositivo posee un sistema operativo llamado iOS que ocupa menos de 500 MB. Es una variante del Mac OS X que se encuentra en los ordenadores Mac. El iOS tiene 4 capas de abstracción:

- La capa del núcleo del sistema operativo
- La capa de servicios principales
- La capa de medios de comucación
- La capa de chocolate touch

Al igual que en el caso de Android, Apple tiene la tienda virtual Apple Store para poder descargarse aplicaciones, algunas son de pago, de todo tipo: juegos, entretenimiento, navegación, utilidades, de cámara, bolsa, climatología, etc. Y entre estas aplicaciones también están las de realidad aumentada, como pueden ser Layar o Wikitude comentadas en un apartado anterior. Por otra parte, como sucede en el caso de Android, el propio usuario puede crear sus propias aplicaciones teniendo conocimientos de Objective-C, variante del C, que es el lenguaje de programación que usan las aplicaciones de iPhone.[7]

Además, existe el proceso Jailbreak, comúnmente conocido por “piratear el iPhone”, que permite instalar aplicaciones no autorizadas por Apple, aunque sean las mismas que están en Apple Store. El mayor inconveniente es que al utilizar este recurso, se pierde toda garantía y soporte de la casa Apple, pero en contraposición se puede hacer un restaurado del sistema dejándolo tal y sale de fábrica. A pesar de que parezca un procedimiento ilegal, tras una resolución judicial, el Jailbreak es declarado totalmente legal.

[7] El SDK se puede encontrar en [http://developer.apple.com/xcode/].

Igual que otras compañías, Apple ha ido actualizando su único dispositivo móvil. A continuación se muestran y explican cada una de las versiones con sus principales características:

3.3.1. iPhone

Este fue el primer dispositivo móvil que Apple lanzó al mercado. Con una apariencia muy similar al iPod, el iPhone hizo que millones de personas salieran a la calle para hacer largas horas de cola con tal de conseguir uno de éstos. Algunos elementos a destacar son la pantalla capacitiva, el altavoz y micrófono, una humilde cámara de 2 MPX, un jack de 3,5 mm para auriculares y conectividad EDGE y WI-Fi. En cuanto a procesador contaba con un ARM11 a 412 MHz.

3.3.2. iPhone 3G

Muy parecido a su antecesor con la única diferencia de incorporar conectividad 3G y una nueva carcasa enteramente de plástico a diferencia del primero que tenía la parte delantera metálica y un procesador ARM11 a 667 MHz, solo que funcionaba a 412 MHz ya que el iPhone 3G se underlockea para ahorrar batear.

3.3.3. iPhone 3GS

En esta actualización se aprecian muchas más diferencias que del primero al segundo. Es el doble de rápido que el 3G, de ahí viene la “S”, de Speed. Además de la velocidad, posee grabación de video, algo que se echó mucho en falta en las anteriores versiones, con una cámara de 3 MPX con autofocus y balance de blancos. También posee brújula, comandos de voz, y la capacidad de copiar y pegar. Otro punto a favor es la prolongación de la batería. Cuenta además con un procesador ARM Cortex-A8 a 620 Mhz a pesar de que su frecuencia normal fuera de 833 MHz.

3.3.4. iPhone 4

La última versión del smartphone de Apple ya presenta muchas mejoras respecto a sus hermanos pequeños. A simple vista se nota un cambio considerable en el diseño: tanto la parte frontal como dorsal están hechas de vidrio con silicato de aluminio que hace mucho más resistente la pantalla contra ralladuras y golpes. Además todo él está rodeado por un marco de acero inoxidable que también hace a la vez función de antena.

En cuanto la pantalla, decir que es de 3,5 pulgadas y con 960x640 PX, denominada por Apple como "retina display" tiene el cuádruple de resolución que la del 3GS. Algo que también se echaba de menos era poder realizar videoconferencias y es por esto que el iPhone 4 ya incorpora una cámara frontal, además de la trasera de 5 MGX con un led flash.

Otras características son la integración de GPS, memoria RAM de 512 MB, un procesador Apple A4 a 1 GHz (pero debido al underlock para ahorrar batería, igual que en el 3G, funciona a 800 MHz), y la capacidad de reproducir y grabar videos en HD 720p, y al igual que todos sus predecesores, pantalla multitouch. A pesar de un pequeño problema de cobertura que se solucionó con unas fundas que regalaba Apple y el despido del ingeniero responsable de tal error, el iPhone 4 es un dispositivo muy completo.

Img. 23: Iphone 3G

Img. 24: Iphone 3GS

Img. 25: Iphone 4

3.4. Symbian

Symbian Ltd. es una empresa de desarrollo de software con sede en Londres. Aunque desarrolla otros tipos de software, es conocida por ser la creadora del sistema operativo Symbian OS. En sus inicios, en el 1998, estableció una alianza con Ericsson, Nokia, Motorola y Psion, para poder competir con los sistemas operativos de Palm y Windows Mobile. Nokia tenía el 56,3% de la empresa, por lo que finalmente en el 2008 compró las acciones que no poseía haciéndose así con la totalidad de la empresa.

Img. 26: Nokia 5800 xPress Music

A partir de ese momento todos los dispositivos móviles Nokia pasaron a tener Symbian OS. Éste cuenta con 6 tipos distintas de interfaces de usuario o plataformas para su sistema operativo:

- Serie 60, Serie 80, Serie 90: Son las series que usan la mayoría de los móviles con Symbian. La Serie 60 en concreto, fue el sistema operativo que propuso Symbian para sus Smartphones. Está compuesto por un conjunto de bibliotecas y aplicaciones informáticas estándar.
- UIQ: desarrollada por UIQ Technology, es la que usan principalmente Sony Ericsson y Motorola
- MOAP: usada por algunos móviles con tecnología 3G de NTT-Docomo.

El SDK de Symbian es el estándar de C++ aunque este tipo de sistema operativo, soporta aplicaciones escritas en Java MIDP 2.0, flash y pyhton.

En cuanto al kernel de Symbian, es un microkernel que contiene solo lo mínimo, órdenes primitivas básicas y de funcionalidad para ofrecer máxima robustez y una rápida capacidad de respuesta. Además contiene un planificador, un gestor de memoria y controladores de dispositivos, con servicios de apoyo a la creación de redes.

Symbian se creó en base a la integridad y seguridad de los datos del usuario. Para esto el microkernel de Symbian mantiene la separación entre la interfaz de usuario y el motor del sistema. Además el SO está optimizado para el rendimiento en dispositivos de bajo consumo alimentados por batería, por lo que la CPU del dispositivo se cambia a modo de bajo consumo cuando no hay aplicaciones en foreground.

Por otro lado, el sistema operativo sigue un modelo de capas, desde la más alta hasta la más baja, esta sería su composición:

- Marco de la interfaz del usuario
- Capa de servicios de la aplicación
  - Java ME
- Capa de servicio del SO
  - Servicios genéricos del SO
  - Servicio de comunicaciones
  - Servicio de gráficos y multimedia
  - Servicios de conectividad
- Capa de servicios base
- Capa de interfaz hardware y servicios del kernel

Igual que en los otros sistemas operativos, los teléfonos Nokia disponían de la tienda OVI (OVI Suite) donde poder comprar y descargar aplicaciones de todo tipo, aunque igual que en Android había otros métodos para descargarlas e instalarlas. En la actualidad y tal como adelantaba The Wall Street Journal, la compañía Nokia integrará en sus dispositivos Windows Phone, así como algunas de sus aplicaciones como el Microsoft Marketplace y los mapas de Bing.

Img. 27: Sony Ericsson Vivaz

3.5. Windows Phone

Este es el sistema operativo en el que más énfasis se pondrá en la explicación, ya que el objetivo de este proyecto es la creación de una aplicación de realidad aumentada para un dispositivo móvil con Windows Phone.

Anteriormente llamado Windows Mobile, Windows Phone es la propuesta de la multitudinaria empresa dirigida por Bill Gates por hacerse un sitio entre los primeros en el mercado de los smartphones. Este SO se basa en el sistema operativo Windows CE y como el resto cuenta con unas aplicaciones básicas, solo que estas utilizan las API de Windows.

Han sido muchas las versiones que han ido apareciendo de Windows para móviles, desde Pocket PC 2000 hasta el actual Windows Phone 7. A continuación se hará un breve recorrido por cada una de ellas mostrando además una imagen de la pantalla inicial para que se pueda apreciar un poco más los cambios que ha habido:

3.5.1. Pocket PC 2000

Como su nombre indica se empezó a comercializar en Abril del 2000 bajo el sobrenombre de "Rapier", y se lanzó con la intención de que fuera el sucesor del sistema operativo que funcionaba en los equipos Palm. La única resolución compatible con este SO era de 240x320 (QVGA) y, en cuanto a la utilización de memorias externas, solo permitía el uso de tarjetas CompactFlash y MultiMedia. Además, como en ese momento no se había normalizado aun una CPU para los dispositivos Pocket PC, se publicaron múltiples arquitecturas de CPU como SH-3, MIPS y ARM.

Img. 28: Pantalla inicial de Pocket PC 2000

Algunas de las características y aplicaciones integradas en Pocket PC 2000 fueron:

- Pocket Office
- Pocket IExplorer
- Windows Media Player
- Microsoft Reader

3.5.2. Pocket PC 2002

Originalmente con nombre "Merlin" el Pocket PC 2002 fue lanzazo al mercado en abril de 2001. Igual que su antecesor fue impulsado por Windows CE 3.0 y la máxima resolución que soportaba era 240x320 (QVGA). Fue la primera vez que también se podía usar este SO en smartphones, básicamente los dispositivos GSM. En cuanto a la apariencia, intentó parecerse al recien lanzado Windows XP.

Algunas de sus aplicaciones y características eran:

- Interfaz de usuario mejorada
- Corrector ortográfico
- Posibilidad de guardar las descargas
- Sincronización con carpetas
- MSN Messenger
- Servicios de Terminal Server
- Mejora en Pocket Outlook
- Microsoft Reader 2

Img. 29: Pantalla inicial de Pocket PC 2002

3.5.3. Windows Mobile 2003 y SE

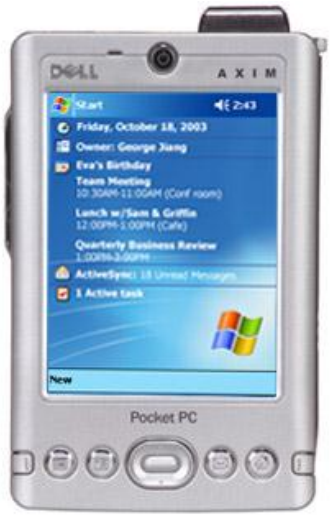
La tercera y renombrada versión del sistema operativo de Windows fue lanzada en abril de 2003 con el nombre en código de "Ozono", y llego en cuatro ediciones distintas: para pocket PC (Premium Edition, Professional Edition y Phone Edition) y para Smartphone. Las diferencias más significativas están en que la Phone Edition es una edición diseñada especialmente para teléfonos móviles, y luego la edición Premium tenía un cliente de VPN L2TP/IPSec. que la Professional no tenía.

Algunas características:

- Apoyo a la extensión de teclados
- Bluetooth para auriculares
- Mejores en el Bluetooth
- Mejora de IExplorer
- Pocket Outlook con apoyo vCard y vCal
- Windows Media Player 9.0
- Soporte para archivos MIDI

Se tuvo que esperar hasta Marzo de 2004 para ver una nueva versión del SO de Microsoft. El Windows Mobile 2003 Second Edition se mostró en un Dell Axim x30. Esta fue la última versión que permitía hacer copias de seguridad y restaurar todo el sistema a través de ActiveSync. Alguna de las novedades fue la posibilidad de poner la pantalla apaisada (en horizontal), el diseño en columna única del IExplorer, resoluciones de pantalla VGA (640x480), (176x220), (240x240) y (480x480), y por último protección WPA para conexiones Wi-Fi.

Img. 30: Pantalla inicial de Windows Mobile 2003



Img. 31: Dell Axim X3

3.5.4. Windows Mobile 5

La versión de Windows Mobile 5 fue presentada en el Microsoft's Mobile and Embedded Developers Conference en Las Vegas el 9 de Mayo de 2005. Se mostró rodando dentro de un dispositivo Dell Axim x51. Entre algunas mejoras, este SO incluye mejoras de funcionalidad en Microsoft Exchange en comparación con la versión de 2003, además con las actualizaciones de WM 5.0 AKU2 todos los dispositivos podían soportar y ser compatibles con DirectPush.

Otra mejora importante en este sistema operativo es la prolongación de la batería gracias a la capacidad de almacenaje persistente, ya que en las versiones anteriores hasta el 50% de la carga de la batería se encargaba de mantener los datos en la memoria RAM volátil. A partir de ese momento todas las versiones de Windows Mobile hacían uso de la memoria RAM como medio de almacenamiento primario combinado con una memoria flash. Los programas y datos de uso más frecuente se almacenaban en la RAM y el resto de datos en la memoria flash.

Por último cabe a destacar que cualquier copia de seguridad que se hiciera, se guarda en la memoria flash a diferencia de las versiones anteriores que lo hacían en la RAM, así pues se conservaran todos los datos en todo caso.

Algunas aplicaciones y actualizaciones:

- Nueva versión de Office, "Office Mobile"
- Windows Media Player 10
- Identificación de llamada por fotografía
- Mejorado el soporte Bluetooth
- GPS
- ActiveSync 4.2 con un incremento del 15% en la velocidad de sincronización.

Img. 32: Pantalla inicial de Windows Mobile 5

3.5.5. Windows Mobile 6

Se tuvo que esperar hasta el 12 de febrero de 2007 para que en la 3GSM World Congress se presentara el nuevo sistema operativo de Microsoft, el Windows Mobile 6 bajo el sobrenombre de "Crossbow", en un SPV E650. Se lanzó en tres versiones diferentes, una para smartphones (Windows Mobile 6 Standard), otra para PDAs con función de teléfono (Windows Mobile 6 Professional) y la última para PDAs sin función de teléfono (Windows Mobile 6 Classic), todos ellos en base a Windows CE 5.2 y con claros dejes con Windows Vista, Windows Live, Microsoft Office y Exchange 2007.

Img. 33: Pantalla inicial de Windows Mobile 6

Igual que en la versión de Pocket PC 2002 que intentó parecerse a la versión de Windows XP, el Windows Mobile 6 hizo lo mismo pero con la última versión del sistema operativo de Windows, el Vista. Funcionalmente se asemeja mucho a Windows Mobile 5 pero con varias mejoras en estabilidad.

En cuanto a especificaciones:

- Soporte para resoluciones 800x480 y 320x320
- Desarrollo y distribución de aplicaciones más rápido
- Windows Live para Windows Mobile
- Operating System Live Update
- Outlook con soporte para HTML
- Soporte para AJAX, JavaScript y XMLDOM
- .NET Compact Framework v2 SP1 en la ROM
- SQL server Compact Edition en la ROM
- Opción de 1:1 en páginas web
- Soporte VoIP con los códec ce audio AEC y MSRT

Img. 34: SPV E650

Casi un año más tarde, el 1 de abril de 2008, salió la primera actualización de Windows Mobile 6. Esta actualización, **la 6.1**, trae consigo varias mejoras de rendimiento y en la Standar Edition una nueva pantalla inicial con fichas horizontales que se expanden al hacer clic para tener más información. También se añadió un programa de "primeros pasos", como una especie de "Paseo por Windows".

Algo importante a destacar es que la Standar Edition tiene la característica de crear vínculos automáticos para los números de teléfono almacenados en "Tareas" y "Citas" que permite la fácil comunicación con Outlook. Además se mejoró el ancho de banda optimizando en el protocolo de push-email ActiveSync.

Img. 35: Pantalla inicial de Windows Mobile 6.1

En el año 2009, más concretamente el 7 de Mayo, apareció **la actualización 6.5**. En esta actualización considerada menor, se aprecia una nueva interfaz gráfica de usuario (GUI) y un nuevo navegador web con una interfaz mejorada y más intuitiva. Esta versión fue diseñada para que el uso de la pantalla táctil fuera más sencillo. Por otro lado, "SkyMarket" pasó a llamarse a Windows Marketplace, y aparece la Microsoft My Phone que permite disponer de 200 MB para hacer copias de backup.

Img. 36: pantalla inicial de Windows Mobile 6.5

Más tarde apareció la **versión 6.5.1** la cual trae una interfaz de usuario mucho más amena y amigable para el uso con los dedos además de los botones basados en imágenes que hay en la pantalla. A parte de esto da apoyo para el A-GPS por parte de Microsoft en lugar de por parte del operador telefónico, mejoras en cuanto a la escritura de mensajes de texto (SMS) y optimizaciones de rendimiento.

El 2 de febrero de 2010, presentado en el Sony Ericsson Aspen apareció la **versión 6.5.3**, aunque extraoficialmente ya circulaba por algunos dispositivos. Esta nueva versión, igual que en las anteriores, trae mejoras en la interfaz de usuario como el uso aún más fácil de la pantalla táctil, soporte multitouch y la capacidad de manejarla con los dedos en vez de con un lápiz, además incorpora la capacidad de poder arrastrar elementos al menú inicio. Por otro lado Internet Explorer Mobile 6 también trae mejoras reduciendo el tiempo de carga de la página y de gestión de memoria. En cuanto a Microsoft Office, esta nueva versión trae Office Mobile 2010.

Img. 37: Pantalla inicial de Windows Phone 6.5.3

Al margen de esta última actualización reconocida por Windows, han parecido de manera no oficial algunas otras versiones desde la aparición de la 6.5. Como no son oficiales, se denominan como versión 6.5.x donde "x" puede cambiar dependiendo del autor de la actualización. De manera resumida, a todas estas versiones se les llama **6.5.5**

3.5.6. Windows Phone 7

Windows Phone 7 fue anunciado el 15 de Febrero de 2010 en el Mobile World Congress en Barcelona, bajo el nombre en clave de "Photon". Este sistema operativo, a diferencia de sus sucesores está más enfocado para el consumo comercial y no tanto para el empresarial. Además con esta edición Microsoft buscará unos estándares mínimos de calidad así como unas configuraciones de hardware muy concretas, y anuncian que para finales de 2011 habrá una gran actualización que será capaz de competir con los sistemas operativos iOS y Android.

Igual que pasó en la transición de Pocket PC 2002 a Windows Mobile 2003, se vuelve a renombrar al nuevo sistema operativo como Windows Phone Serie 7, pero tras una serie de polémicas y quejas de lo largo y complicado que era el nombre, la multinacional decidió prescindir de la palabra "Serie" quedándose solo en Windows Phone 7. Microsoft lo justificó diciendo que: "*Los clientes quieren poderlo decir de una manera simple y coherente, pero que lo importante era mantener la marca Windows Phone".*

Una de las novedades más interesantes es la nueva interfaz de usuario que se aprecia en la imagen posterior, conocida como "Metro", que comparte características visuales con la interfaz de Zune HD **(PONER RESEÑA).** A primera vista lo que más sorprende es la pantalla inicial, la cual está compuesta por mosaicos que cambian de manera dinámica mostrando información personalizable por el usuario. Dicha información puede ser desde SMS o mails pendientes, citas, juegos, etc. Además estos mosaicos pueden arrastrarse y colocarse donde más se desee.

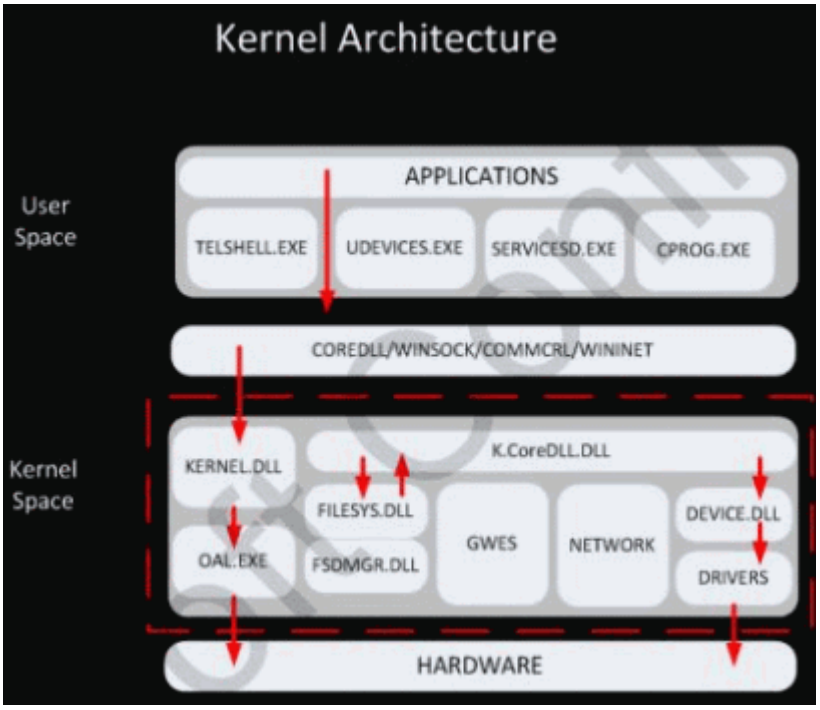
Img. 38: Dispositivo con sistema operativo Windows Phone 7

Estos son los requisitos hardware para instalar un WP7 en un dispositivo, según Microsoft son “altos, pero justos”:

Requisitos hardware para WP7
Pantalla capacitiva con resolución WVGA (800x480) con 4 puntos multitouch
1 GHz ARM v7 “Cortex/Scorpion”
GPU DirecX 9 con capacidad de representación
256 MB de RAM con al menos 8 GB de memoria flash
Acelerómetro con brújula, sensor de luz, sensor de proximidad y A-GPS
Cámara de 5 MPX con flash
FM Radio
6 botones dedicados: Volver, Inicio, Buscar, On/Off, cámara, volumen up/down

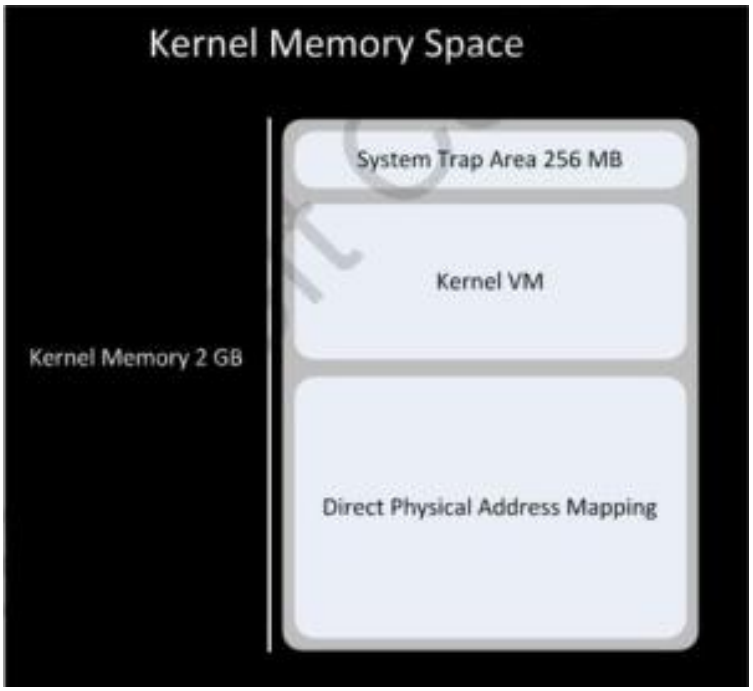
Otra gran novedad es la denominada “hub”. La función de los “hubs” es clasificar y agrupar acciones y aplicaciones en grupos que tengan una actividad determinada, es decir, se tendrá un “hub” para contactos, otro para redes sociales, otro para imágenes y videos, etc. La gran ventaja de este sistema es que, en el “hub” de imágenes y videos por ejemplo, se tendrán juntos videos, fotografías y aplicaciones para edición de éstos /-as. Así pues, sucede lo mismo en el “hub” de contactos, donde además de tener la agenda de contactos, se pueden tener almacenados los comentarios que han hecho en Facebook o Windows Live. Con el “hub” de música sucede lo mismo, dentro de éste se tienen las listas de reproducción y todas las canciones y en adición se pueden tener podcasts, el servicio Zune, y multiples aplicaciones de edición o distribución de contenido multimedia.

La plataforma de este sistema operativo está basada en ARM, más concretamente con instrucciones ARMv7. En cuanto a la arquitectura, decir que WP7 es un sistema operativo de 32 bits de doble capa y está basado en Windows Embedded CE 6.0 a diferencia de los Windows Mobile 6.x que tienen un kernel construido en Windows CE 5.0. Una de las principales ventajas de que sea de 32 bits es que esto permite un direccionamiento de hasta 4 GB.



Img. 39: Arquitectura del kernel del sistema operativo Windows Phone 7

En esta primera imagen se puede apreciar la arquitectura del kernel compuesta de dos capas distinas: la del espacio de usuario y la capa del núcleo. Dentro del espacio de usuario se encuentran los servicios del sistema operativo, el Shell, etc. Y dentro de la capa del núcleo se pueden ver el kernel, el sistema de archivos, el renderizado de gráficos, la radio, los drivers, etc.

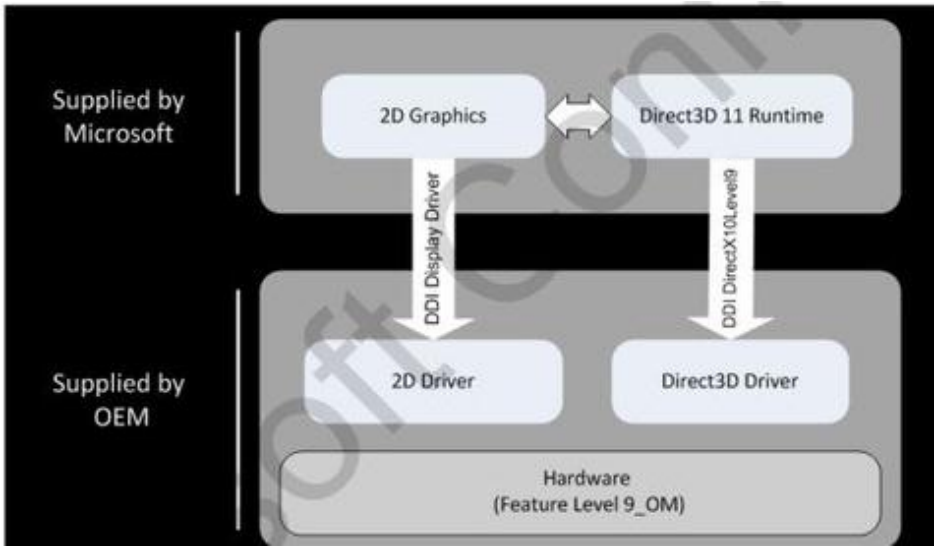


Img. 40: Espacio de memoria del kernel

En esta segunda imagen se puede apreciar el espacio de memoria del kernel de 2GB, la cual está dividida en tres partes: el área de System Trap de 256 MB, una memoria virtual del kernel que puede llegar hasta 1GB y el área de mapeo directo a la memoria física. Además, WP7 contará con dos tipos de archivos distintos: IMGFS para archivos de sistema y TextFAT para archivos de usuario. Estos últimos son una versión ampliada del sistema de ficheros FAT capaz de hacer frente a archivos de 4 GB.

Sin embargo Microsoft ha optado por un enfoque de almacenamiento unificado, es decir, que las aplicaciones y los usuarios no distinguen entre los archivos de almacenamiento local o de una tarjeta de memoria. Por lo que surge el problema que si una tarjeta de memoria contiene archivos clave y se formatea o se pierde, el teléfono quedaría inutilizado solamente pudiendo realizar llamadas de emergencia.

Por último se hará un breve comentario sobre los gráficos, a continuación se muestra el esquema.



Img. 41: Esquema sobre el sistema de gráficos

Windows Phone 7 utiliza por defecto una versión de Direct3D 11, basado en DirectX 10, para trabajar con 3D, lo que permite a los fabricantes poder escribir sus propios drivers en 2D y 3D a través del marco de trabajo que ofrece Microsoft. Lo único que sorprende es que el hardware trabaja con la versión Direct3D 9 que es anterior, aunque a pesar de ello, como muestra la imagen, la API permite trabajar sin problema con una versión anterior.

4. Creando la aplicación

4.1. Objetivo de la aplicación

Tal y como se ha estado explicando a lo largo de todo este documento, se desea implementar una aplicación para Windows Phone 7 que use la realidad aumentada. El objetivo es sobreponer a la visión en pantalla de la cámara una imagen concreta que será recibida a través de un socket cliente TCP/IP así como las coordenadas y los datos necesarios para ubicarla.

El uso de la aplicación será sencillo, al iniciarse aparecerá una pantalla en negro con un botón que será el que iniciará la cámara y abrirá el socket que espera conexiones entrantes. Una vez viéndose la imagen que muestra la cámara, por cada conexión entrante la aplicación enviará un mensaje por pantalla diciendo si se desea aceptar o no. En caso que se diga que "si", la aplicación mostrara la imagen recibida en las coordenadas recibidas, en caso contrario, se hará caso omiso a la conexión entrante.

Img. 42: Posible pantalla inicial de la aplicación

Img. 43: Posible pantalla de ejecución de la aplicación

4.2. Instalar el entorno de trabajo

Para poder trabajar y crear aplicaciones de Windows Phone 7, lo primera que se tiene que hacer es descargarse el software necesario, en este caso el *Microsoft Phone Developer Tools*, el cual incluye:

- Visual Studio 2010 Express for Windows Phone
- Windows Phone Emulator
- Silverlight for Windows Phone
- XNA Game Studio 4.0
- Expression Blend 4.0 for Windows Phone

Para ello se describirán a continuación los pasos necesarios para poder descargarlo e instalarlo:

4.2.1. Paso 1: Requisitos del sistema

Antes de instalar el software es necesario saber que solo se podrá instalar si se cumplen una serie de premisas:

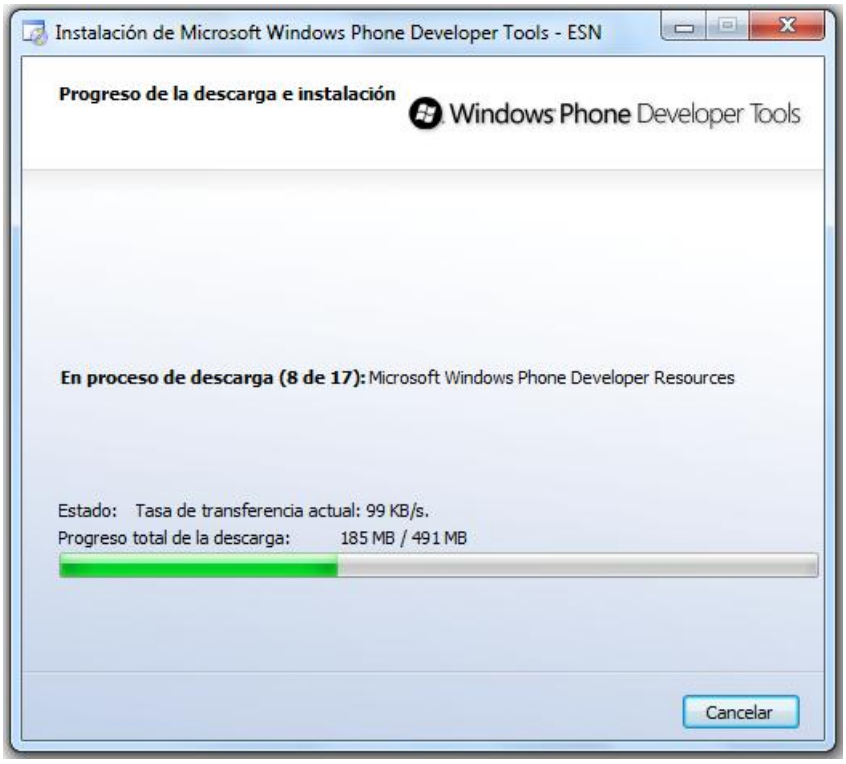
- Sistema operativo:
  - Windows 7 (x84 y x64) exceptuando la versión Starter Edition.
  - Windows Vista (x84 y x64) con Service Pack 2 exceptuando la versión Starter Edition
- Hardware:
  - Memoria mínima de 2GB de RAM
  - La instalación requiere de 3 GB de disco duro
  - Tarjeta gráfica: DirectX 10 o posterior con el driver WDDM 1.1
- NO SOPORTA:
  - Windows XP y Windows Server
  - Virtual PC e Hyper-V

4.2.2. Paso 2: Descarga del Software

Una vez cumplidos los requisitos de hardware y software, se puede proceder a descargarse los programas necesarios para poder desarrollar aplicaciones de WP7. Para ello se debe entrar en la siguiente página [http://www.microsoft.com/express/Phone/] y clicar en "*Get Started*". Aunque se recomienda bajárselo de aquí [http://www.microsoft.com/express/Downloads/#2010-Visual-Phone] ya que permite escoger el idioma del software, en cambio en el primer link está por defecto en inglés.

4.2.3. Paso 3: Instalación del software

La instalación es muy sencilla, haciendo doble clic en el archivo ejecutable que se ha descargado en el paso anterior, aparecerán las típicas ventanas extrayendo el contenido del paquete, y a continuación aparecerá esta ventana:



Img. 44: Ventana de descarga de software

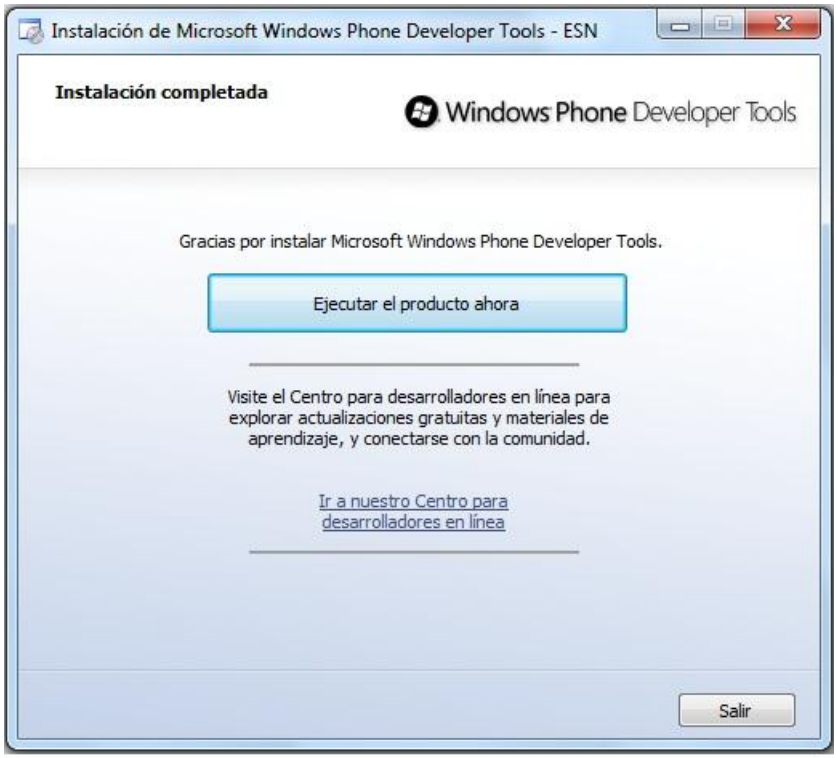
Este proceso automatizado descargará todos los programas anteriormente mencionados pertenecientes al *Microsoft Phone Developer Tools*. Son 491 MB, dependiendo de la conexión la descarga será más o menos rápida.

Una vez descargados todos los paquetes necesarios, comenzará el proceso de instalación con una ventana muy similar a la anterior:

Img. 45: Ventana de instalación de software

4.2.4. Paso 4: Ejecutar el software

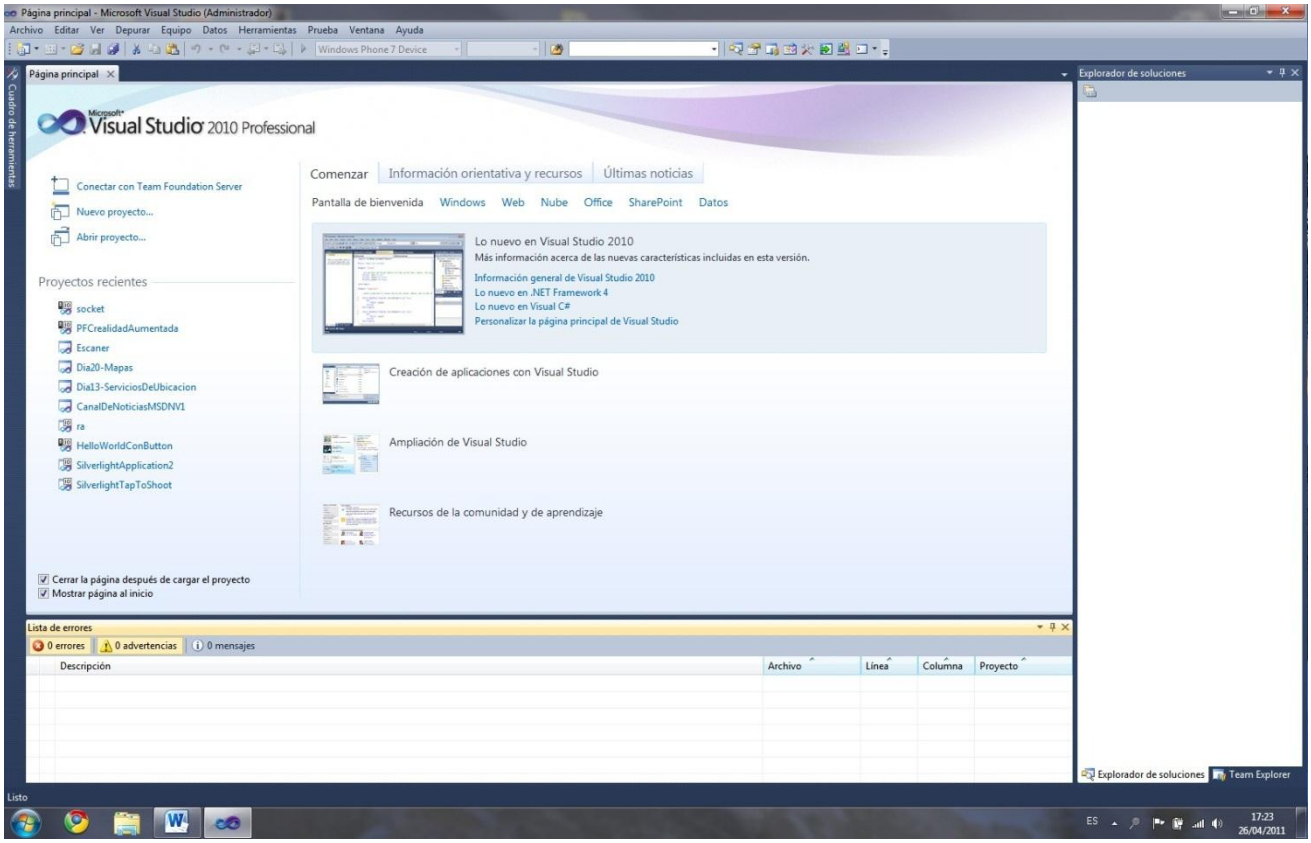
Al finalizar la instalación aparecerá la ventana conforme el proceso ha finalizado y el software ya está listo para usarse:



Img. 46: Ventana de instalación finalizada

4.3. Microsoft Visual Studio Express for Windows Phone

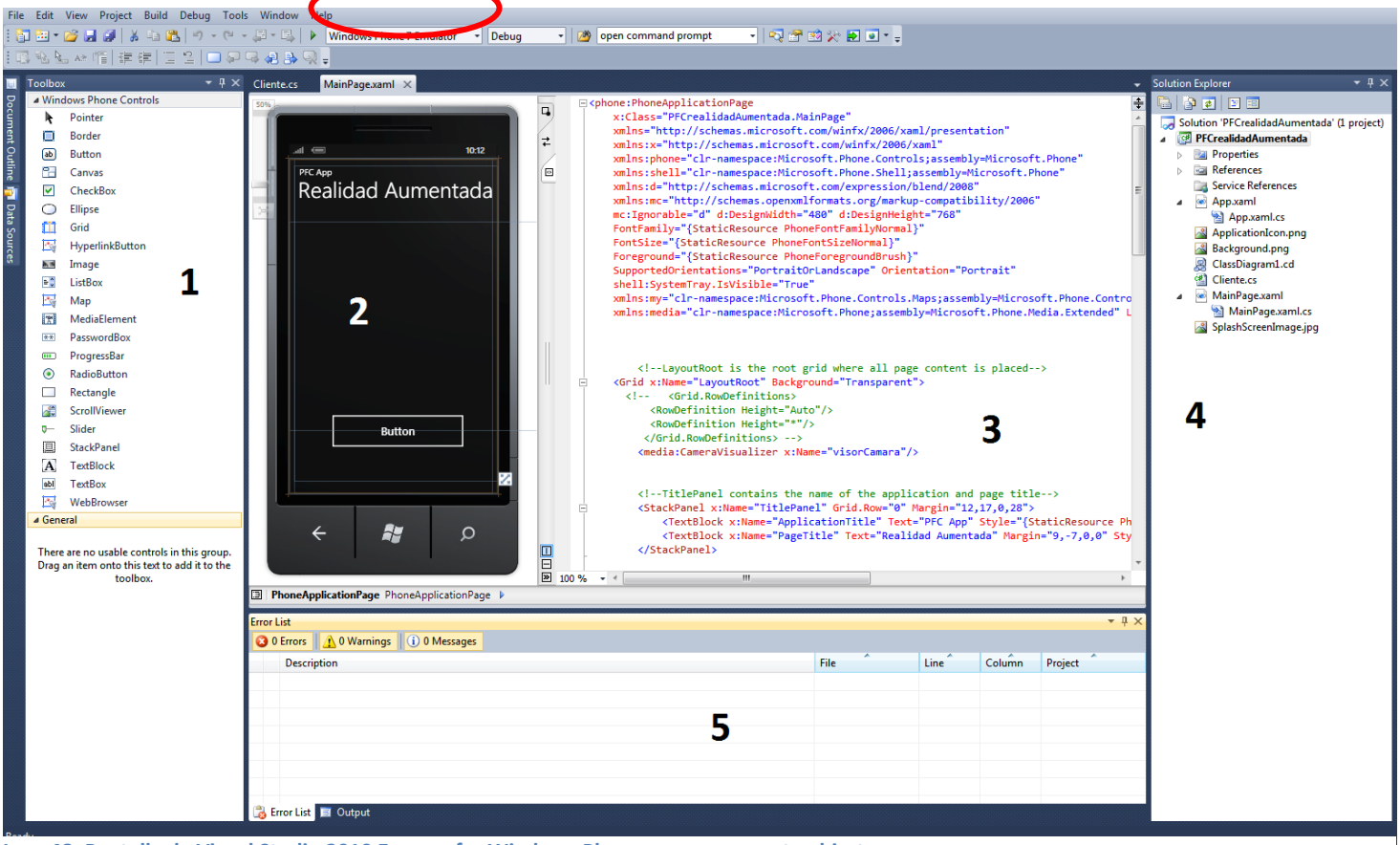
Después de finalizar la instalación del software, ya se puede proceder a abrir la aplicación con la que se realizará la aplicación, la cual no deja de ser el propio Visual Studio 2010 pero especializado para Windows Phone. La pantalla inicial se muestra a continuación:



Img. 47: Pantalla inicial de Visual Studio 2010 Express for Windows Phone

Ésta permite iniciar un nuevo proyecto, abrir uno de existente, y también permite la navegación entre sus pestañas para obtener información orientativa y recursos así como las últimas noticias acerca del producto. Además, dado la novedad del nuevo sistema operativo de Windows para móviles y a su vez la creación de este software, en esta pantalla inicial también se puede obtener información acerca del nuevo .NET Framework 4 y lo nuevo de Visual C#.

En cuanto a la distribución de las secciones que componen el editor es similar a la de cuando se inicia un nuevo proyecto, a continuación se van a describir las cuatro secciones que lo componen:



Img. 48: Pantalla de Visual Studio 2010 Express for Windows Phone con un proyecto abierto

Sección 1

En esta sección se muestra la lista de herramientas, es decir, los elementos que se pueden añadir al emulador, como por ejemplo: botones, cuadros te texto, imágenes, pases de diapositivas, etc. Una vez seleccionado el elemento deseado se debe arrastrar a la pantalla del emulador de la sección 2 para poderlo insertar en el diseño y poderlo editar. Al mismo tiempo que se añade el elemento de manera visual a la pantalla del dispositivo, se añade en formato código en la sección 3.

Sección 2

En ésta se muestra la parte frontal de un dispositivo móvil con Windows Phone 7, el cual tiene los 3 botones obligados (*atrás*, *menú* y *búsqueda)* y muestra la estructura que tienen las aplicaciones de Windows Phone 7: un título, un subtítulo y un espacio más grande para el cuerpo de la aplicación. También incluye detalles como la cobertura, la batería restante y la hora (inanimados). Permite además, poder insertar y editar (cambio de posición, tamaño, etc.) los elementos del cuadro de herramientas, al mismo tiempo, cualquier cambio realizado en la pantalla del emulador se realizará simultáneamente en código reflejado de la sección 3.

Sección 3

Esta es la sección más importante de todas ya que en ella se escribe el código de la apariencia (archivo .xaml) y funcionalidad (archivo de C#) de la aplicación. De la misma manera que insertando elementos a la pantalla del emulador se añade el código correspondiente, ocurre de manera recíproca al realizarlo al revés. De todos modos, se recomienda diseñar la apariencia de la aplicación añadiendo los elementos desde la caja de herramientas que no insertando código, salvo que ya se tenga cierto dominio.

Por otro lado, en esta misma sección también se pueden visualizar los códigos del funcionamiento en sí de la aplicación. Estos códigos están escritos en C# (.cs), y pueden ligarse a las funcionalidades de los elementos de la pantalla, es decir, en el archivo .xaml se tiene la apariencia pero no la funcionalidad. Para darle una determinada acción a un elemento se usan los archivos de C# relacionando las acciones con los elementos visuales.

Sección 4

Se muestra en forma de árbol la estructura del proyecto. Esta estructura incluye las clases, los paquetes (*references*) a los que se hace referencia para el uso de las diferentes características del dispositivo como puede ser la cámara, las imágenes que se muestran en la aplicación, el archivo .xaml de apariencia, etc.

Sección 5

En esta última sección se muestran los errores, las advertencias (*warnings*) y los mensajes que se puedan producir a la hora de compilar y ejecutar el código, indicando el archivo, la fila y la columna donde se ha producido el error. También tiene la característica de mostrar el output si es que así lo establece el código de la aplicación.

Por último, en la imagen anterior donde están señaladas todas las secciones, también hay una elipse roja que rodea una lista desplegable. En esta lista se puede seleccionar donde se desea ejecutar la aplicación, ya sea en el emulador o en un dispositivo real. Para poder ejecutar el código en este último, es necesario descargarse el software *Zune*.

Este es el software que se perfila para hacerle la competencia a Itunes de Apple. Análogamente *Zune* tendrá su Zune MarketPlace donde se podrá descargar y comprar música y videos. Aunque no solo se tratará de un software, *Zune HD* también es el nombre del nuevo reproductor de Microsoft que combatirá en el mercado con el Ipod.

Algunas de sus características son:

- Pantalla OLED de 3,3 pulgadas multitouch con resolución de 480x272
- Conexión WIFI
- Navegador web con teclado en pantalla
- HD Radio
- Conectividad con TV con posibilidad de visualizar videos en HD 720p

Img. 49: Reproductor multimedia Zune HD

4.4. Windows Phone Emulator

Dentro del paquete de software *Windows Phone Developer Tools* se encuentra la aplicación *Windows Phone Emulator*. Esta aplicación, como su propio nombre indica, es un emulador de un dispositivo móvil dotado de sistema operativo Windows Phone. Para poderlo usar, es imprescindible tener una seria de requisitos, los más importantes se citan a continuación:

- Sistema operativo: Windows 7 (32-bit y 64-bit), Windows Vista (32-bit y 64-bit)
- Memoria: 2GB de RAM y a su vez, 1.5GB libres
- Emulador GPU: Tarjeta gráfica DirectX 10 o DirectX 11 con WDDM 1.1 o posterior y DDI 10
- Paquetes .xap: El tamaño máximo de un paquete .xap es 225MB para aplicaciones de Silverlight y aplicaciones XNA Framework
- Multi-Touch: La simulación de multi-touch con el ratón no es compatible, se requeriría un equipo host que admitiera entrada multi-touch
- Almacenamiento: Durante el funcionamiento del emulador los datos guardados se mantienen, una vez cerrado el emulador, se pierde todo.
- **Acelerómetro, GPS y cámara: Actualmente no soportado**

Por otra parte, permite las siguientes características:

- Emulación de dispositivos periféricos
- Emulación del procesador, de la memoria RAM, de la pantalla y GPU
- Almacenamiento persistente (mientras está en funcionamiento el emulador)
- Creación de redes
- Uso de teclado
- Implementación
- Depuración (Debuggin)

4.5. Desbloqueo del dispositivo

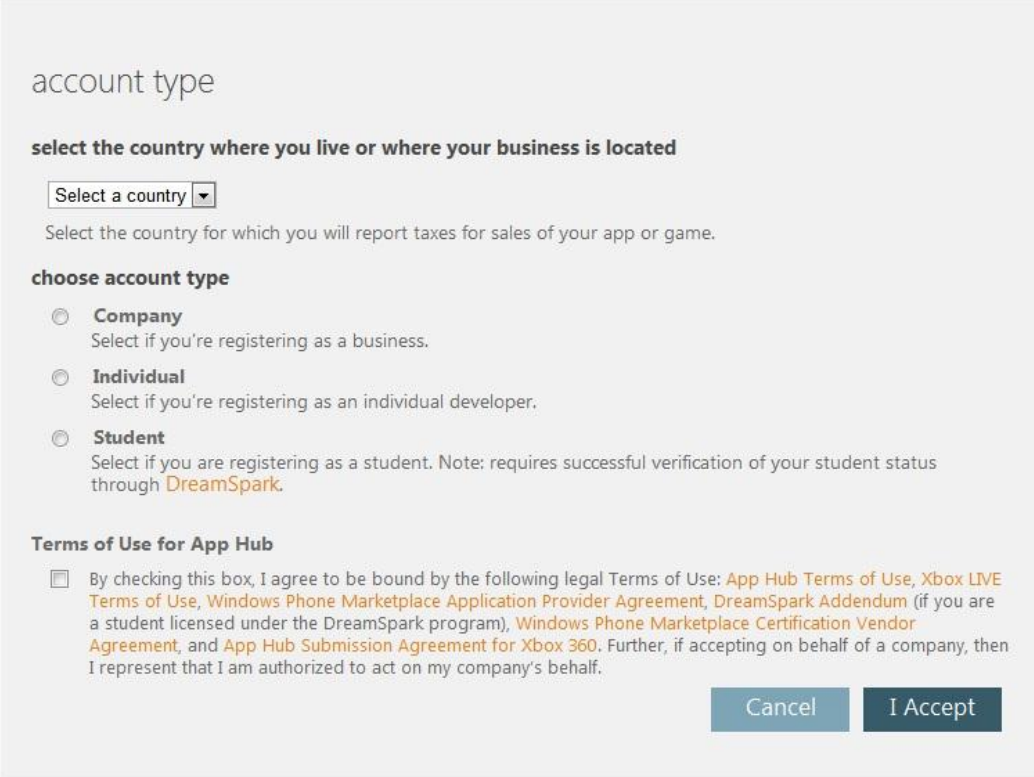
Para poder probar las aplicaciones creadas en un dispositivo, primero de todo se tiene que desbloquear (es decir, **hacerse administrador del dispositivo**). En el software *Windows Phone Developer Tools* viene un pequeño programa llamado *Windows Phone DeveloperRegistration* con el cual, habiendo cumplido una serie de premisas, se puede desbloquear el terminal.

A continuación se hará una descripción de los pasos que se tienen que seguir para cumplir estas premisas ya que todo el proceso es algo confuso:

4.5.1. Paso 1: Darse de alta como desarrollador de aplicaciones de WP7

Para darse de alta se ha de entrar en la siguiente página web [http://developer.windowsphone.com]. En la esquina superior derecha hay un apartado que pone "*Sign in"* en el cual se tendrá que acceder mediante una ID de Windows Live (la cuenta de Hotmail por ejemplo), si no se tiene uno, también permite la opción de crearse una cuenta nueva.

Una vez iniciado sesión, aparecerá una pantalla como esta:



Img. 50: Pantalla de selección de país y tipo de cuenta para darse de alta como desarrollador

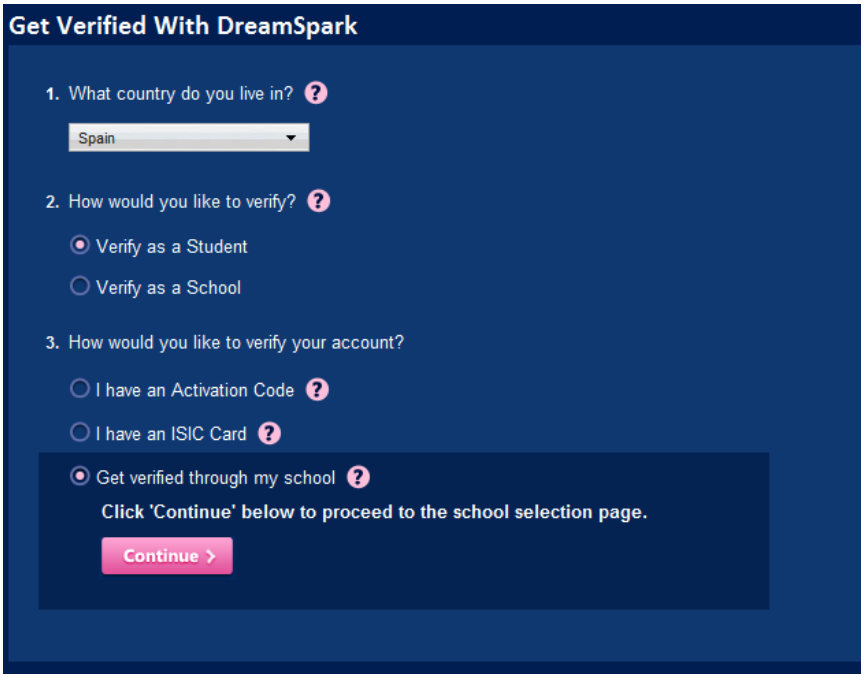
En esta pantalla primero de todo se tiene que seleccionar el país donde se reside o en el caso de ser una empresa el país donde está ubicada ésta. Después se ha de seleccionar el tipo de cuenta que se desea, existen tres tipos**: compañía** (empresa), **desarrollador individual**, o **estudiante**. Para la realización de este proyecto se ha escogido el tipo estudiante, lo cual conlleva tener que registrarse en otra web llamada *DreamSpark*. Además, es la única gratuita ya que tanto para compañías y desarrollador individual se tiene que pagar una cierta cantidad.

4.5.2. Paso 2: Registrarse en DreamSpark

DreamSpark es un programa de ayuda de Microsoft, que provee gratuitamente herramientas de diseño y desarrollo a estudiantes de múltiples países del mundo, su página web es [https://www.dreamspark.com]. En esta web hay un apartado llamado "*Get Started"* en el que en tres pasos, se puede dar de alta como estudiante.

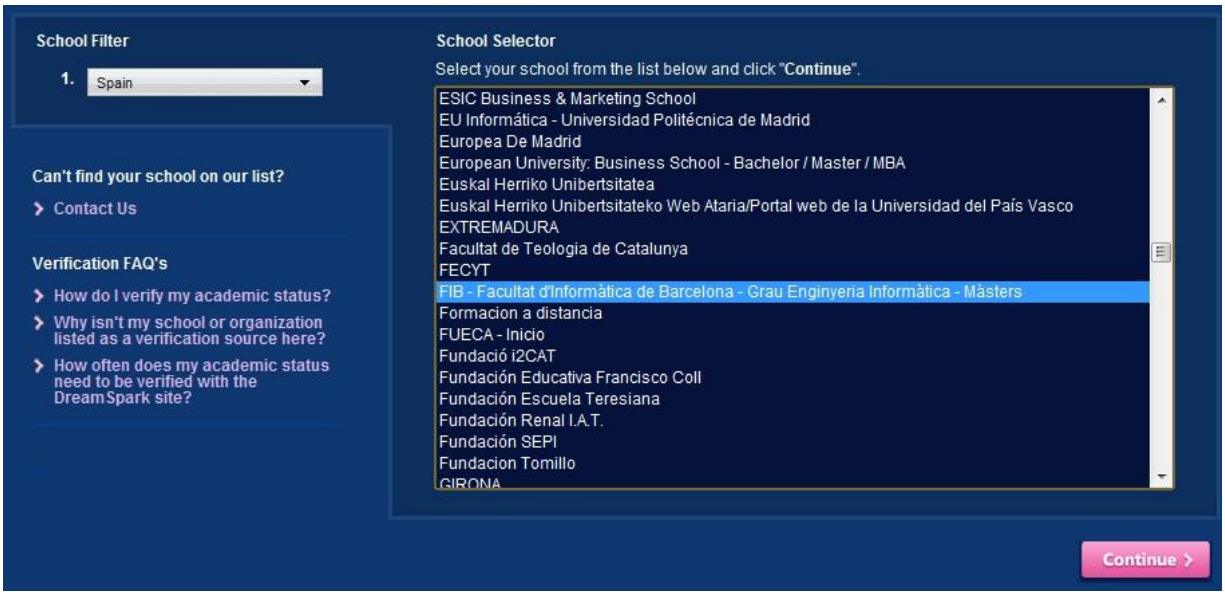
Img. 51: Sección donde se ven los pasos que ya están completos

Igual que se ha hecho en la web anterior de desarrolladores de WP7, lo primero de todo es identificarse con el Windows Live ID. Una vez registrado, se procederá a realizar el paso 2, en el cual Microsoft verifica que realmente la persona que va a darse de alta es estudiante. Para ello aparece la siguiente pantalla:



Img. 52: Pantalla de elección de país, y tipo de cuenta de DreamSpark

Lo primero que se ha de hacer en esta pantalla es seleccionar el país en el que vivimos. Luego seleccionar si el registro se hace a nivel de escuela (instituto, universidad…) o se hace a nivel de estudiante, en este caso se ha seleccionado a nivel de estudiante. Y por último, escoger el método con el que Microsoft verificará el status de estudiante. Para esto último existen tres opciones: verificar mediante un código de activación, mediante la tarjeta ISIC (carnet de estudiante) o seleccionando la escuela, o en este caso universidad, de una lista y mediante el correo electrónico de la universidad verificar que realmente es un estudiante. Aparece una pantalla como esta:



Img. 53: Pantalla de selección de universidad o centro de estudios

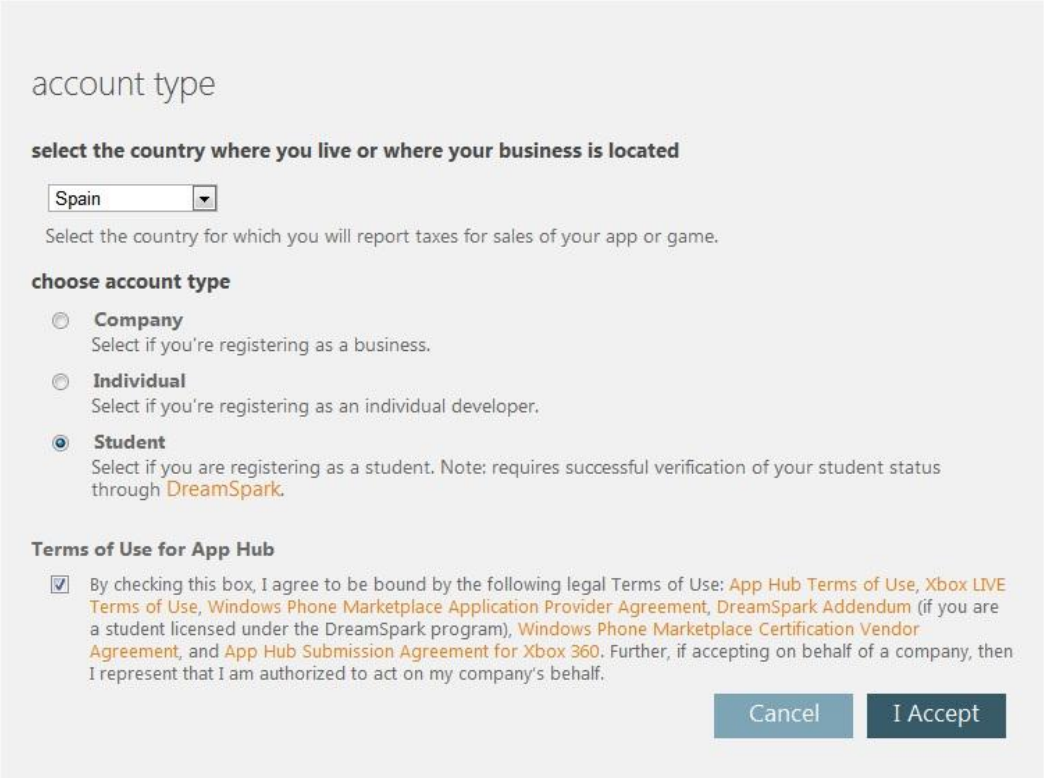
En este caso se seleccionó, tal y como aparece en la imagen, la verificación por selección de universidad. Una vez seleccionada, aparecerá otra pantalla donde se ha de poner la dirección de correo proporcionada por la universidad, en este caso *carlos.machado@est.fib.upc.edu* para que Microsoft lo verifique.

Una vez terminado todo el proceso aparecerá en el apartado "*Get Started"* de la pantalla principal (Img. 54) que ya están todos los pasos completos, y ya se podrá volver a la página de desarrolladores de aplicaciones de WP7 para continuar el registro.

Img. 54: Pantalla con todos los pasos completos

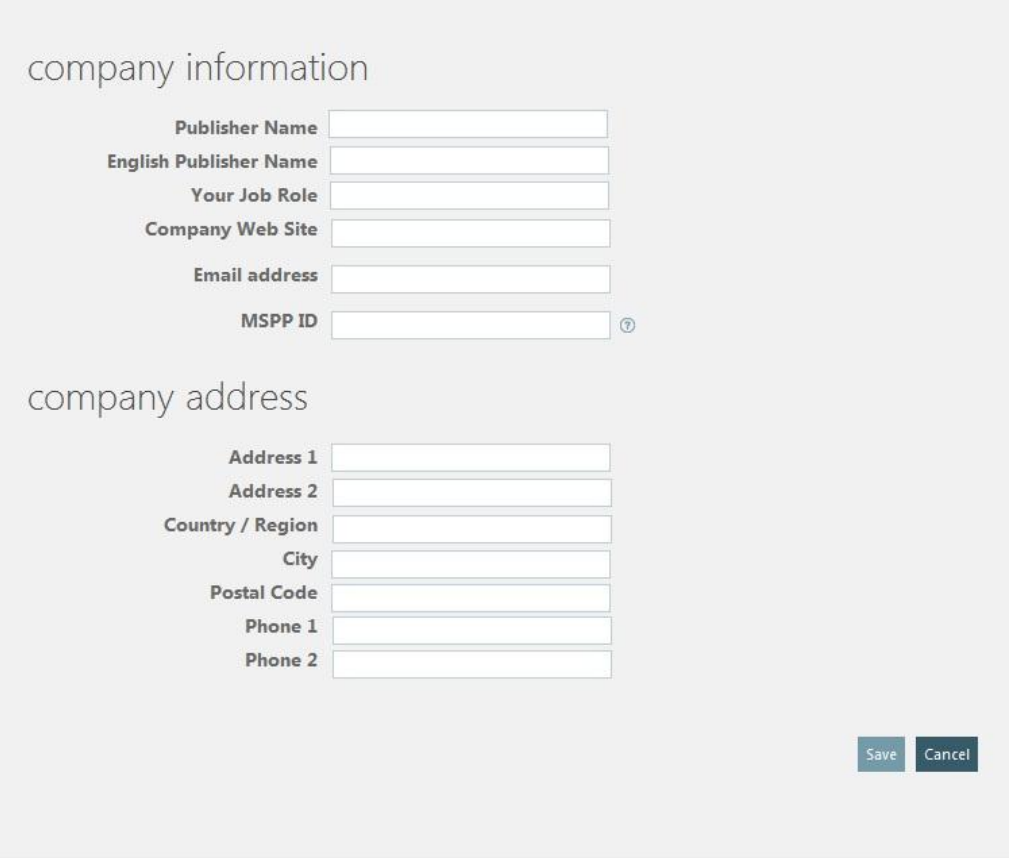
4.5.3. Paso 3: Finalizar registro como desarrollador

Después de finalizar el registro en *DreamSpark*, se volverá a la web de desarrolladores de aplicaciones de WP7 para continuar el registro donde se dejó:



Img. 55: Pantalla de selección de país y tipo de cuenta una vez finalizado el registro en DreamSpark

Ahora ya se puede seleccionar el tipo de cuenta *Student*, aceptar los términos de uso y clicar en aceptar. A continuación aparecerá la siguiente pantalla (Img. 56) donde el usuario deberá rellenar con sus datos personales: nombre para sus publicaciones, dirección de correo electrónico, dirección, país, código postal, etc.



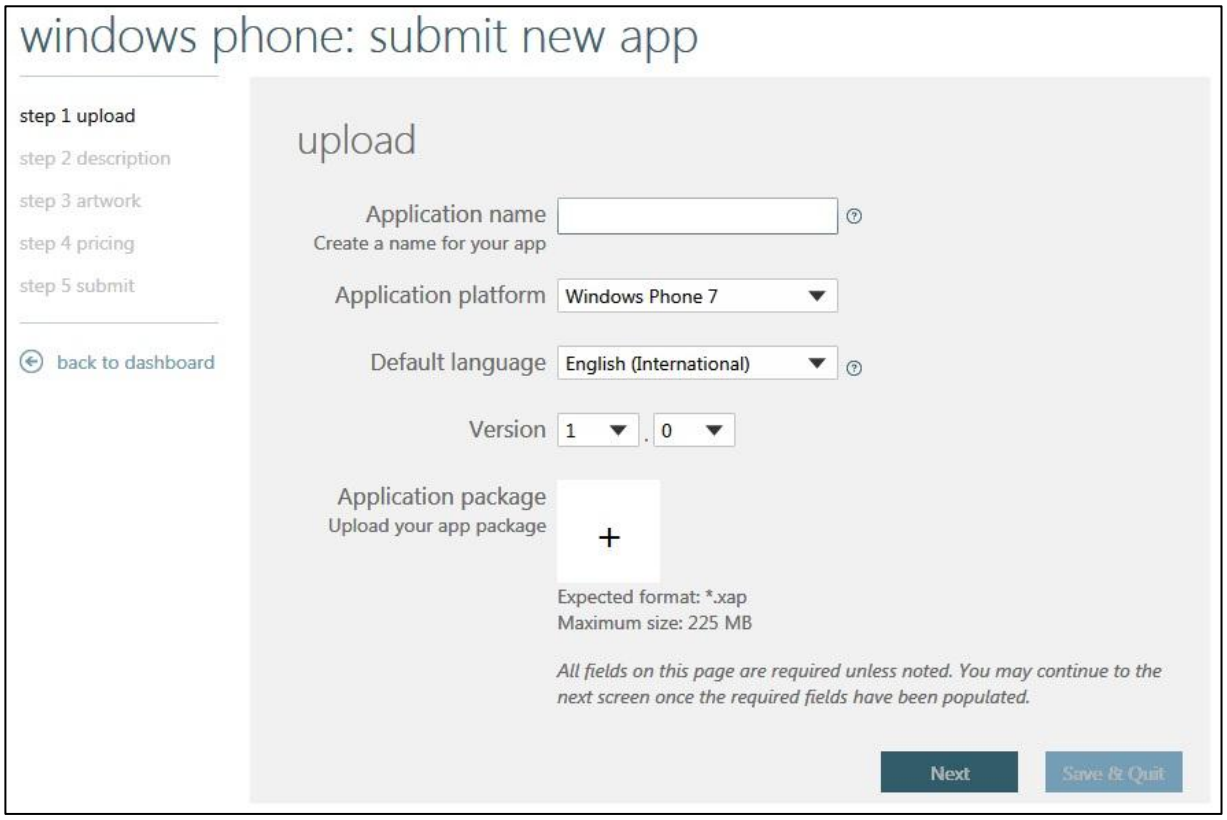
Img. 57: Pantalla de registro de datos personales

Después de rellenar la página anterior con los datos personales de usuario, aparecerán tres pantallas más: una para editar el perfil de usuario, otra sobre datos de pago (al ser estudiante no se debe pagar nada) y una última de confirmación.

A pesar de estar registrado como desarrollador de aplicaciones para WP7, y estar también registrado en *DreamSpark* como estudiante, aún falta un paso más para poder llevar a cabo el desbloqueo del terminal.

4.5.4. Paso 4: Envío de aplicación

Para poder finalizar el proceso, y que Microsoft verifique que realmente quien ha hecho todos los trámites es el usuario de la dirección de correo introducida como Windows Live ID, faltará realizar un último paso que consta de enviar una aplicación sencilla (tanto como una aplicación que simplemente muestre un mensaje) desde la página de desarrolladores. Al enviarla, aparecerá una pantalla como esta:



Img. 58: Pantalla para subir la aplicación para que GeoTrust verifique la identidad del usuario

Una vez enviada, se habrá de esperar a que la empresa GeoTrust lo verifique, y mande confirmación. Si se desea acelerar el proceso, es aconsejable enviar un email a GeoTrust pidiendo que aceleren el proceso de verificación, entonces la empresa envía un correo con una carta, la cual se debe rellenar adjuntando una fotocopia de la tarjeta de identidad, en el caso de España el DNI, y enviársela de nuevo.

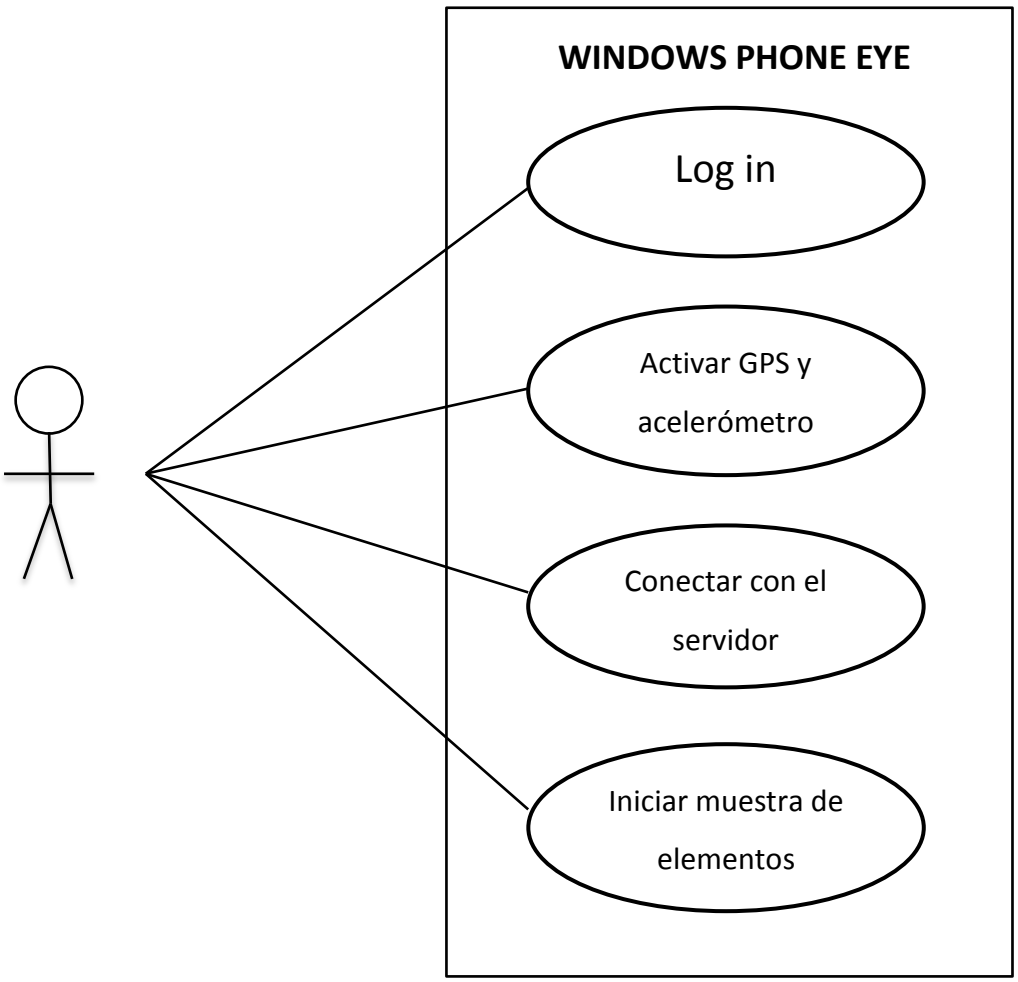
Después de todos estos pasos, ya solo queda volver al *Windows Phone Developer Registration*, conectar el dispositivo, introducir el Windows Live ID que se ha estado utilizando durante todo el proceso de registro en las distintas webs, y completar el proceso. Aparecerá una pantalla como esta:

Img. 59: Pantalla final para desbloquear el dispositivo

5. Detalles de la aplicación

En este apartado se expondrán los detalles de la aplicación. Como se ha explicado en apartados anteriores, existen dos tipos de archivos: los de extensión .xaml y los de extensión .cs. De extensión .xaml suele haber solo uno, y es el que usualmente contiene el código, muy similar al html, la interfaz de la aplicación. En cuanto a los archivos .cs, están escritos en C# y pueden estar solos o hacer referencia a un archivo .xaml como en el caso de los proyectos de aplicaciones para Windows Phone. La página inicial de la aplicación se llama MainPage.xaml y la asociada a ésta escrita en C# se llama MainPage.xaml.cs, desde la cual se pueden referenciar objetos de la página inicial.

5.1. Diagrama de caso de uso



5.2. Pantalla Inicial

A continuación se podrá ver una imagen de la apariencia de la pantalla inicial de la aplicación. En cuanto al código del archivo MainPage.xaml y MainPage.xaml.cs con sus respectivas aclaraciones de sus distintas partes se pueden encontrar en los anexos 1 y 2 respectivamente.

Img. 60: Pantalla inicial definitiva de la aplicación

Se realizará la explicación de la pantalla inicial (Img. 60) de arriba abajo. Después del título del proyecto de la aplicación (Windows Phone EYE) y del título de la aplicación (Aumented Reality), lo primer que se encuentra es un botón con el título de "*Activar GPS/Acc*" el cual al apretarlo pone en funcionamiento la localización del dispositivo vía GPS y la puesta en marcha del acelerómetro. Una vez activado, el estado de ambos elementos pasa de "*Disabled*" a "*Ready*" y en los campos "*Longitude*" y "*Latitude*" aparecen la longitud y latitud actuales del dispositivo en ese momento, así como su orientación en los campos "*X*", "*Y*", "*Z*", que es un valor que oscila entre -1 y 1. Además, aparece una tachuela o indicador, de donde se encuentra el móvil en el mapamundi.

El botón "*Comenzar*" hará que aparezca la cámara para que se empiecen a visualizar elementos. Y por último el botón "*Parar*" parará el GPS y el acelerómetro. En cuanto a las opciones del menú inferior, de izquierda a derecha, la primera despliega una ventana de ayuda, la del medio una ventana para iniciar sesión mediante usuario y contraseña, y la última permite introducir una IP y un puerto para conectarse al servidor que enviará los datos de los elementos a mostrar.

5.3. Menú

Estas son las tres ventanas de cada una de las opciones del menú:

Img. 61: Pantalla del menú de ayuda

Img. 62: Pantalla del menú Log in

Img. 63: Pantalla del menú IP/Port

La primera imagen (Img. 61) es la del menú ayuda, es simplemente una ventana en la que salen las instrucciones a seguir para que la aplicación funcione correctamente. Para cerrar solo hay que presionar el botón "*Cerrar*".

La segunda imagen (Img. 62) corresponde a la ventana del menú que permite iniciar sesión. Contiene dos campos de texto donde escribir el nombre de usuario y la contraseña que aparecerá con asteriscos. Una vez introducidos los datos requeridos se apretará al botón "*Log in*" para iniciar sesión.

Y por último, la imagen de la derecha (Img. 63) es la más importante de todas ya que es la del menú de selección de IP y puerto del servidor al cual se conectará la aplicación para recibir los datos a mostrar en pantalla. Es de geometría idéntica a la ventana de LOG IN, salvo por el detalle que el teclado que aparece cuando se quiere introducir la IP y el puerto, es solo numérico, a diferencia de la de LOG IN que es el estándar alfanumérico.

5.4. Funcionamiento

A continuación se describirán una serie de pasos a seguir para el correcto funcionamiento de la aplicacione Windows Phone EYE:

Paso 1

Una vez abierta la aplicación, se debe entrar en el menú LOG IN para introducir el usuario y contraseña y presionar en el botón “Log in” para que la aplicación guarde los datos introducidos.

Paso 2

Abrir en el ordenador el servidor llamado SDLPS, cargar el modelo de simulación y ponerlo en funcionamiento apretando el botón “Initialize”. Una vez puesto en funcionamiento aparecerá una ventana diciendo que el servidor espera conexiones en el puerto #8686. A continuación, abrir el menú de “IP/Port” e introducir la IP y el puerto y pulsar el botón “Conectar” para conectarse al servidor que será el que enviará los datos para representar en pantalla los elementos. Lo que se hace es crear un socket con dirección y puerto los introducidos con el servidor.

Paso 3

Aunque se haya conectado con el servidor desde la aplicación, seguirá apareciendo la ventana de esperando conexiones. Para finalizar el estado de conexión y el servidor, en la pantalla inicial se debe iniciar el GPS y el acelerómetro pulsando en el botón “Activar GPS/Acc” y a continuación pulsar sobre “Comenzar”. Haciendo esto, la aplicación envía un mensaje al servidor para finalizar la conexión, y a partir de ese momento ya se puede iniciar la transmisión de datos pulsando el botón “Run” del servidor SDLPS. Para la transmisión de datos, se abre otro socket por el que cliente y servidor se envían mensajes.

6. Informe económico y sostenibilidad

En este apartado se hará un pequeño análisis de costes económicos de la aplicación así como su impacto medioambiental por su realización. Al ser una aplicación de teléfono móvil la mayoría de los gastos serán los del software y hardware utilizado y el precio del terminal con el que se ha hecho la aplicación.

A diferencia de iPhone o android, tal vez debido a su reciente lanzamiento, no se ha de pagar para obtener las herramientas de desarrollo para Windows Phone 7, lo cual es un ahorro a la hora de hacerse con el software. En cuanto al hardware, existen unos requisitos para poder instalar el SDK 7.0 o 7.1. Son los siguientes:

- Windows 7 o Windows Vista
- Windows ® Vista ® (x86 y x64) con Service Pack 2 - todas las ediciones salvo Starter Edition
- Windows 7 (x86 y x64) - todas las ediciones salvo Starter Edition
- La instalación requiere 4 GB de espacio libre en disco en la unidad del sistema.
- 3 GB de RAM
- Emulador de Windows Phone requiere un DirectX 10 o superior Tarjeta gráfica compatible con un driver WDDM 1.1

Con estos requisitos, para que el software vaya fluido, el ordenador debe tener un procesador mínimo de doble núcleo a 2GHz. Actualmente se pueden encontrar ordenadores en el mercado con estas características por 300 euros.

La descarga del software (*Windows Phone Developer Tools*) como se ha mencionado anteriormente es totalmente gratuita desde la página de Microsoft, y en cuanto al registro como desarrollador de aplicaciones de WP7, como se ha explicado en el apartado 4.5, tampoco tiene coste alguno si se registra como estudiante. En cuanto al terminal móvil, para este proyecto se ha utilizado un Smartphone HTC Mozart de la compañía Orange, el precio del cual puede variar dependiendo del tipo de contrato que se desee, aunque para dar un importe, se situaría entre los 169 euros. Por otra parte, el tiempo empleado para el desarrollo y finalización del proyecto han sido 4 meses.

En cuanto al coste económico en horas trabajadas, se empezó a realizar el proyecto el día 14 de febrero y se ha finalizado el día 17 de junio. Lo que en total son 90 días (excluidos sábados y domingos) a 4 horas por día salen un total de 360 horas de trabajo.

En resumen, el coste de la aplicación sería:

169€ → dispositivo

300€ → hardware

360h * 35€ = 12.600€ →sueldo analista

169€ + 300€ + 12.600€ = **13.069€**

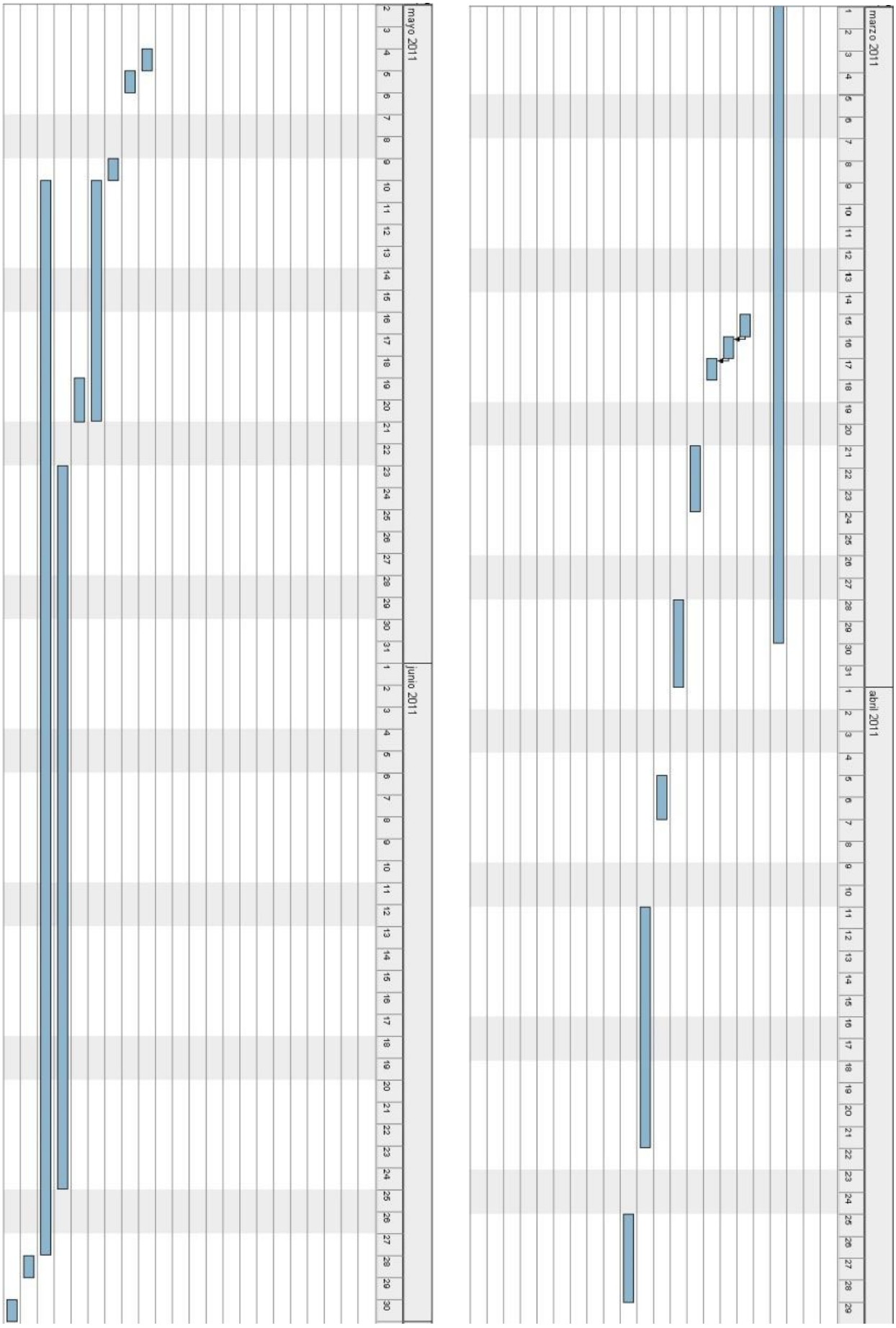
Finalmente, cabe decir que la realización de la aplicación en si no ha tenido ningún impacto medioambiental ni efecto contaminante para el planeta de forma directa. No obstante, el uso de la aplicación requiere de un terminal móvil, el cual se tiene que cargar conectándolo a la corriente, electricidad de la cual proviene principalmente de combustibles fósiles, que en el proceso de generar electricidad se queman produciendo dióxido de carbono y otros gases de efecto invernadero en la atmosfera, lo que significa que la temperatura media de la Tierra aumente.

La vida útil de un móvil en media se sitúa en 1 año y para la fabricación de un teléfono móvil se necesitan elementos como petróleo crudo, cobre, oro, paladio, silicio, aluminio, litio, plomo, antimonio, etc. Todos estos elementos al extraerlos de la tierra se produce un impacto en el medioambiente con el desgaste de montañas, pozos,... Igual que cuando estos teléfonos son arrojados a vertederos o son incinerados, estas sustancias pueden filtrarse en la tierra y llegar a pozos subterráneas de agua. Varios estudios demuestran que si se alargara la vida útil de un móvil de 1 a 4 años, se podría reducir el impacto medioambiental en un 40%.

Por otro lado, está el uso del GPS y conexión a internet. Según un estudio realizado por el programa *Panorama* de la cadena BBC, las ondas electromagnéticas que generan las redes WIFI son tres veces más potentes que las que generan los teléfonos móviles. Así también el Ministerio Federal Alemán de Medio Ambiente señala que es preferible siempre que se pueda la instalación de cables como la fibra óptica en escuelas para la transmisión de datos ya que la exposición de electromagnetismo a edades tempranas puede ser muy perjudicial.

7. Calendario

Para la mayor claridad en la explicación del calendario que se siguió a la hora de hacer el proyecto, se ha decidido mostrarlo como un diagrama de Gantt, el cual contiene todas las tareas y los días que se ha tardado en realizarlas.


marzo 2011
abril 2011
mayo 2011
junio 2011

8. Conclusiones

Una vez finalizado el proyecto, la primera sensación que se tiene es la de haber trabajado con un software y unos recursos limitados debido al escaso tiempo que lleva Windows Phone 7 en el mercado. También, habiendo tenido más tiempo se hubieran podido solucionar algunos de los problemas que fueron surgiendo a la hora de ir programando la aplicación. El principal de esos problemas fue la falta de soporte a sockets de la versión 7.0 del SDK de Windows Phone, lo cual es un elemento fundamental e imprescindible a la hora de comunicarse por internet.

A pocos días de la fecha límite de la entrega del proyecto, Microsoft lanzó la versión Beta del SDK 7.1 que incluye entre otras cosas, el soporte a sockets. Pero a continuación, surgió otro problema, y es que Microsoft no ha sacado aún el firmware 7.1 para el dispositivo móvil. Y todas las pruebas se tuvieron que realizar con el emulador de Windows Phone, el cual, no puede visualizar la cámara.

La aplicación creada, gracias al GPS localiza y muestra en un mapa la posición del dispositivo. Con el acelerómetro detecta la posición del móvil respecto a los ejes x, y, z. Permite además la conexión vía sockets a un servidor, el cual le enviará trazas de un modelo de simulación con los datos pertinentes para poder mostrar posteriormente elementos sobre las imágenes que capturará la cámara del dispositivo.

Pasados unos meses, Microsoft lanzará la versión definitiva del SDK 7.1 y el firmware correspondiente para los dispositivos con lo que se podrá finalizar la aplicación y ejecutarla en los móviles. Así que en resumen, lo que faltaría por hacer sería:

- La recepción de los datos del servidor.
- El procesado de los datos.
- Mostrar los elementos recibidos sobre las imágenes de la cámara.

9. Bibliografía

- http://es.wikipedia.org/wiki/Realidad_aumentada
- http://www.nexusmania.info/2010/09/mozilla-seabird-prototipo-de-movil.html
- http://logit42.com/archives/4706
- http://www.xataka.com/otros/skinput-microsoft-convierte-tu-brazo-en-una-pantalla-tactil
- http://msdn.microsoft.com/en-us/library/ff431744(v=vs.92).aspx
  De esta web se sacaron varios códigos de ejemplo (se muestran en el anexo III)
- http://maromasdigitales.net/category/blog/windows-phone/
  De esta web se sacaron códigos para el aprendizaje del lenguaje C# (se muestran en el anexo III)
- http://msdn.microsoft.com/en-us/library/ff431744(v=vs.92).aspx
  De esta web se sacaron códigos para el aprendizaje del lenguaje C# (se muestran en el anexo III)
- http://www.maestrosdelweb.com/editorial/que-es-realidad-aumentada/
- http://msdn.microsoft.com/en-us/library/hh202858(v=vs.92).aspx#Y2754

ANEXOS

ANEXO I

Código archivo MainPage.xaml

Código MainPage.xaml

```
<phone:PhoneApplicationPage
    x:Class="PFCFinal.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:media="clr-
namespace:Microsoft.Phone;assembly=Microsoft.Phone.Media.Extended"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait"
    shell:SystemTray.IsVisible="True"
    xmlns:map="clr-
namespace:Microsoft.Phone.Controls.Maps;assembly=Microsoft.Phone.Controls.Maps">

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle" Text="Windows Phone EYE"
Style="{StaticResource PhoneTextNormalStyle}"/>
            <TextBlock x:Name="PageTitle" Text="Aumented Reality" Margin="9,-
7,0,0" FontSize="59" Style="{StaticResource PhoneTextTitle1Style}"/>
        </StackPanel>



        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">



        <map:Map Height="627"
                 HorizontalAlignment="Left"
                 Name="miMapa"
                 VerticalAlignment="Top"
                 Width="456"
                 Opacity="0.2"
                 CredentialsProvider="Am_lyCh3hii
                FRT7DEZPiEWWaVbBo
                8ciCo_uOychGW9FAFSUc6Lzl
                BIAZT8iSkP2U"
                    >
        </map:Map>
```

Código correspondiente al mapa que hay en segundo plano (translúcido). El campo "*CredentialsProvider*" contiene el código credencial para usar mapas de "Bing" y no aparezca el logo en medio del mapa.

```
<Button x:Name="GPSACCButton"
        Content="Activar GPS/Acc"
        Height="100"
        HorizontalAlignment="Center"
        VerticalAlignment="Top"
        Width="300"
        Click="GPSACCButton_Click"
        FontSize="30" />
```

Código correspondiente al primer botón con el texto "*Activar GPS/Acc*"

```
<TextBlock Height="30"
           HorizontalAlignment="Left"
           Margin="130,110,0,0"
           Name="estadoGPSAcc"
           Text="Disabled"
           FontSize="23"
           VerticalAlignment="Top"
           Width="180"
           TextAlignment="Center" />
```

Código correspondiente al estado del GPS y el aceleróemtro. El texto va cambiando según el estado entre: "*Disabled*", "*Initializin*", "Ready" y "NoData"

```
<TextBlock Text="Longitude: "
           Margin="20,150,0,362"/>

<TextBlock Text="Latitude: "
           Margin="20,180,0,332"/>

<TextBlock Name="Longit"
           Text="0"
           Margin="125,150,0,0"/>

<TextBlock Name="Lat"
           Text="0"
           Margin="125,180,0,0"/>

<TextBlock Text="X: "
           Margin="20,210,0,302"/>

<TextBlock Text="Y: "
           Margin="20,240,0,272"/>

<TextBlock Text="Z: "
           Margin="20,270,0,242"/>

<TextBlock Name="x"
           Text="0"
           Margin="50,210,0,0"/>

<TextBlock Name="y"
           Text="0"
           Margin="50,240,0,0"/>

<TextBlock Name="z"
           Text="0"
           Margin="50,270,0,0"/>
```

Código correspondiente a los campos de texto "*Longitude*", "*Latitude*", "*X*", "*Y*", "*Z*" y sus respectivos campos de textos para colocar el valor

```
        <Button x:Name="StartButton"
                Content="Comenzar"
                Height="100"
                HorizontalAlignment="Left"
                Margin="130,312,0,150"
                VerticalAlignment="Center"
                Width="200"
                Click="StartButton_Click"
                IsEnabled="False"
                FontSize="30" />
```

Código correspondiente al botón con el texto "*Comenzar*". El cual inicia la cámara

```
        <TextBlock Height="30"
                   HorizontalAlignment="Left"
                   Margin="73,370,0,50"
                   Name="Status"
                   Text="Inactivo"
                   FontSize="23"
                   VerticalAlignment="Center"
                   Width="306"
                   TextAlignment="Center" />
```

Código correspondiente al estado de la cámara al pulsar el botón comenzar. Pasa de "*Inactivo*" a "*Activo*"

```
        <Button Content="Parar"
                Height="72"
                HorizontalAlignment="Left"
                Margin="150,0,0,30"
                Name="StopButton"
                VerticalAlignment="Bottom"
                Width="160"
                Click="StopButton_Click"
                IsEnabled="False"
                FontSize="22" />
```

Código correspondiente al botón con el texto "*Parar*" el cual detiene el GPS y el acelerómetro

```
    </Grid>
</Grid>

<!--Sample code showing usage of ApplicationBar-->
<!--<phone:PhoneApplicationPage.ApplicationBar>
     <shell:ApplicationBar IsVisible="True" IsMenuEnabled="True">
         <shell:ApplicationBarIconButton IconUri="/Images/appbar_button1.png"
Text="Button 1"/>
         <shell:ApplicationBarIconButton IconUri="/Images/appbar_button2.png"
Text="Button 2"/>
         <shell:ApplicationBar.MenuItems>
             <shell:ApplicationBarMenuItem Text="MenuItem 1"/>
             <shell:ApplicationBarMenuItem Text="MenuItem 2"/>
         </shell:ApplicationBar.MenuItems>
     </shell:ApplicationBar>
  </phone:PhoneApplicationPage.ApplicationBar>-->

</phone:PhoneApplicationPage>
```

Código correspondiente a la barra de menús, en este proyecto se ha creado la barra de menú desde el fichero MainPage.xaml.cs. Es por eso que esta parte de código está comentada.

ANEXO II

Código archivo MainPage.xaml.cs
y
SocketClient.cs

Código MainPage.xaml.cs

Se procederá a explicar de manera individual cada una de las funciones y elementos que componen el archivo MainPage.xaml.cs.

Librerias

- Microsoft.Devices.Sensors
- Microsoft.Phone
- Microsoft.Phone.Controls.Maps
- Microsoft.Phone.Interop
- Microsoft.Phone.Media.Extended
- Microsoft.Xna.Framework
- mscorlib
- System
- Syetem.Core
- System.Device
- System.Net
- System.Windows
- System.Xml
- System.Xml.Linq

Una vez instalado *Microsoft Visual Studio 2010 Express for Windows Phone,* estas librerías se pueden encontrar en el siguiente path:

C:\ProgramFiles\ReferenceAssemblies\Microsoft\Framework\Silverlight\Profile\4.0\WindowsPhone

Espacios de nombre

```
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;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;
using System.Device.Location;
using System.Diagnostics;
using System.Windows.Media.Imaging;
using System.IO;
using System.Text;
using System.Net.Sockets;
using System.Xml.Linq;
using System.Xml;
using System.Windows.Controls.Primitives;
using System.IO.IsolatedStorage;
using Microsoft.Phone.Controls;
using Microsoft.Devices.Sensors;
using Microsoft.Phone;
using Microsoft.Phone.Tasks;
using Microsoft.Phone.Controls.Maps;
using Microsoft.Phone.Controls.Maps.Platform;
using Microsoft.Phone.Shell;
using Microsoft.Xna.Framework.Media;
using Microsoft.Phone.Net.NetworkInformation;
```

Variables Globales

```
Accelerometer acc = new Accelerometer();
GeoCoordinateWatcher GeoWatcher;
CameraCaptureTask camera;
String posX, posY, posZ;
String lat, longit;
String user, contrasena;
PasswordBox password;
SocketClient client;
Button conectar;
ApplicationBarIconButton btnHelp;
ApplicationBarIconButton btnConfig;
ApplicationBarIconButton btnIport;
TextBox porText;
TextBox ipText;

Popup help = new Popup();
Popup config = new Popup();
Popup iport = new Popup();
```

Variables correspondientes para la creación de la barra de menú

MainPage() (constructor)

```
/* Función Main que inicializa los componentes gráficos básicos así como los
botones de menú y sus eventos al pulsar sobre éstos. */
public MainPage()
{

// Función que inicializa todos los componentes gráficos básicos de una
aplicación de Windows Phone.
     InitializeComponent();


// Creación de la barra de menú y configuración la visibilidad y las propiedades.
     ApplicationBar = new ApplicationBar();
     ApplicationBar.IsVisible = true;
     ApplicationBar.IsMenuEnabled = true;


// Creación de los botones de la barra de menú.
     btnHelp = new ApplicationBarIconButton(new
Uri("/Iconos/appbar.questionmark.rest.png", UriKind.Relative));
     btnConfig = new ApplicationBarIconButton(new
Uri("/Iconos/appbar.feature.settings.rest.png", UriKind.Relative));
     btnIport = new ApplicationBarIconButton(new
Uri("/Iconos/appbar.download.rest.png", UriKind.Relative));


 // Etiquetas para los botones de la barra de menú.
     btnHelp.Text = "Ayuda";
     btnConfig.Text = "Config";
     btnIport.Text = "IP/Port";


  // Adición de los botones a la barra de menú.
     ApplicationBar.Buttons.Add(btnHelp);
     ApplicationBar.Buttons.Add(btnConfig);
     ApplicationBar.Buttons.Add(btnIport);


  // Creación de los eventos al pulsar los botones.
     btnHelp.Click += new EventHandler(btnHelp_Click);
     btnConfig.Click += new EventHandler(btnConfig_Click);
     btnIport.Click += new EventHandler(btnIport_Click);
}
```

BtnConfig_Click

```
/* Función que crea la ventana, el marco, y todos los componentes del evento
cuando se aprieta el botón config. */
void btnConfig_Click(object sender, EventArgs e)
{
     // Crear una ventana con el fondo negro.
     StackPanel panelConfig = new StackPanel();
     panelConfig.Background = new SolidColorBrush(Colors.Black);
     panelConfig.Width = 410;
     panelConfig.Height = 565;

     // Crear un marco blanco.
     Border border = new Border();
     border.BorderBrush = new SolidColorBrush(Colors.White);
     border.BorderThickness = new Thickness(7.0);

     // Crear el título de la ventana y los botones.
     TextBlock textblockConfig = new TextBlock();
     textblockConfig.FontSize = 24;
     textblockConfig.Foreground = new SolidColorBrush(Colors.White);
     textblockConfig.Text = "LOG IN";
     textblockConfig.Margin = new Thickness(160, 10, 0, 0);

     // Crear textbox para usuario.
     TextBlock user = new TextBlock();
     user.Text = "User: ";
     user.TextWrapping = TextWrapping.Wrap;
     user.Foreground = new SolidColorBrush(Colors.White);
     user.FontSize = 20;
     user.Margin = new Thickness(20, 20, 0, 0);

     //Crear textbox para introducir el usuario.
     TextBox usuario = new TextBox();
     usuario.Margin = new Thickness(10, 20, 10, 0);

     // Crear textbox para contraseña.
     TextBlock passwd = new TextBlock();
     passwd.Text = "Password: ";
     passwd.TextWrapping = TextWrapping.Wrap;
     passwd.Foreground = new SolidColorBrush(Colors.White);
     passwd.FontSize = 20;
     passwd.Margin = new Thickness(20, 40, 0, 0);

     // Crear textbox para introducir la contraseña.
     PasswordBox password = new PasswordBox();
     password.PasswordChar = '*';
     password.Margin = new Thickness(10, 20, 10, 0);

     // Crear el botón de log in.
     Button conectar = new Button();
     conectar.Content = "Log in";
     conectar.Margin = new Thickness(80, 40, 80, 0);
     conectar.Click += new RoutedEventHandler(loggin_Click);
```

```
        // Crear el botón cerrar.
        Button close = new Button();
        close.Content = "Cerrar";
        close.Margin = new Thickness(100, 40, 100, 0);
        close.Click += new RoutedEventHandler(cerrar_Click);

        // Adición de los elementos a la ventana.
        panelConfig.Children.Add(textblockConfig);
        panelConfig.Children.Add(user);
        panelConfig.Children.Add(usuario);
        panelConfig.Children.Add(passwd);
        panelConfig.Children.Add(password);
        panelConfig.Children.Add(conectar);
        panelConfig.Children.Add(close);
        border.Child = panelConfig;

        /* Establecer la ventana como propiedad de hijo del marco,
        el cual contendrá la ventana, los textboxs y los botones. */
        config.Child = border;

       // Ajustar la posición de la ventana en la pantalla.
       config.VerticalOffset = 150;
       config.HorizontalOffset = 40;

       // Abrir la ventana y cerrar las otras en caso de estar abiertas.
       if (help.IsOpen) help.IsOpen = false;
       if (iport.IsOpen) iport.IsOpen = false;
       config.IsOpen = true;
}
```

Cerrar_Click

```
// Función que cierra la ventana de configuración.
void cerrar_Click(object sender, RoutedEventArgs e)
{
     config.IsOpen = false;
}
```

Loggin_Click

```
/* Función que copia en dos variables globales los strings correspondientes al
usuario y contraseña introducidos*/
void loggin_Click(object sender, RoutedEventArgs e)
{
     this.user = this.usuario.Text;
     this.contrasena = this.password.Password;
     this.conectar.IsEnabled = false;

}
```

BtnIport_Click

```
/* Función que crea la ventana, el marco, y todos los componentes del evento
cuando se aprieta el botón Ip/port. */
void btnIport_Click(object sender, EventArgs e)
{
     // Crear una ventana con el fondo negro.
     StackPanel panelIP = new StackPanel();
     panelIP.Background = new SolidColorBrush(Colors.Black);
     panelIP.Width = 410;
     panelIP.Height = 565;

     // Crear un marco blanco.
     Border borderr = new Border();
     borderr.BorderBrush = new SolidColorBrush(Colors.White);
     borderr.BorderThickness = new Thickness(7.0);

     // Crear el título de la ventana y los botones.
     TextBlock textblockIP = new TextBlock();
     textblockIP.FontSize = 24;
     textblockIP.Foreground = new SolidColorBrush(Colors.White);
     textblockIP.Text = "IP/Port";
     textblockIP.Margin = new Thickness(160, 10, 0, 0);

     // Crear texblock para IP.
     TextBlock dirIp = new TextBlock();
     dirIp.Text = "IP: ";
     dirIp.TextWrapping = TextWrapping.Wrap;
     dirIp.Foreground = new SolidColorBrush(Colors.White);
     dirIp.FontSize = 20;
     dirIp.Margin = new Thickness(20, 20, 0, 0);

     // Crear textbox para introducir la IP.
     this.ipText = new TextBox();
     this.ipText.Margin = new Thickness(10, 20, 10, 0);
     InputScope Keyboard = new InputScope();
     InputScopeName ScopeName = new InputScopeName();
     ScopeName.NameValue = InputScopeNameValue.TelephoneNumber;
     Keyboard.Names.Add(ScopeName);
     this.ipText.InputScope = Keyboard;

     // Crear textblock para el puerto.
     TextBlock port = new TextBlock();
     port.Text = "Port: ";
     port.TextWrapping = TextWrapping.Wrap;
     port.Foreground = new SolidColorBrush(Colors.White);
     port.FontSize = 20;
     port.Margin = new Thickness(20, 40, 0, 0);

     // Crear textbox para introducir el puerto.
     this.porText = new TextBox();
     this.porText.Margin = new Thickness(10, 20, 10, 0);
     InputScope KeyboardPort = new InputScope();
     InputScopeName ScopeNamePort = new InputScopeName();
     ScopeNamePort.NameValue = InputScopeNameValue.TelephoneNumber;
     KeyboardPort.Names.Add(ScopeNamePort);
     this.porText.InputScope = KeyboardPort;
```

```
        // Crear el botón de conectar.
        Button conectar = new Button();
        conectar.Content = "Conectar";
        conectar.Margin = new Thickness(80, 40, 80, 0);
        conectar.Click += new RoutedEventHandler(conectar_Click);

        // Crear el botón de cerrar.
        Button close = new Button();
        close.Content = "Cerrar";
        close.Margin = new Thickness(100, 40, 100, 0);
        close.Click += new RoutedEventHandler(cortar_Click);

        // Adición de los elementos a la ventana.
        panelIP.Children.Add(textblockIP);
        panelIP.Children.Add(dirIp);
        panelIP.Children.Add(ipText);
        panelIP.Children.Add(port);
        panelIP.Children.Add(porText);
        panelIP.Children.Add(conectar);
        panelIP.Children.Add(close);
        borderr.Child = panelIP;

        /* Establecer la ventana como propiedad de hijo del marco,
        el cual contendrá la ventana, los textboxs y los botones. */
        iport.Child = borderr;

        // Ajustar la posición de la ventana en la pantalla.
        iport.VerticalOffset = 150;
        iport.HorizontalOffset = 40;

        // Abrir la ventana y cerrar las otras en caso de estar abiertas.
        if (help.IsOpen) help.IsOpen = false;
        if (config.IsOpen) config.IsOpen = false;
        iport.IsOpen = true;
}
```

Cortar_Click

```
// Función que cierra la ventana de Ip/port.
void cortar_Click(object sender, RoutedEventArgs e)
{
        iport.IsOpen = false;
}
```

Conectar_Click

```
/* Función que conecta con el servidor a través de un socket correspondiente a la
IP y el puerto introducidos en los campos Ip y puerto. */
void conectar_Click(object sender, RoutedEventArgs e)
{
     if (NetworkInterface.GetIsNetworkAvailable())
     {
          this.client = new SocketClient();
          int puerto = Convert.ToInt16(porText.Text);
          client.Connect(ipText.Text, puerto);

     }
 }
```

BtnHelp_Click

```
/* Función que crea la ventana, el marco y texto del botón help. */
void btnHelp_Click(object sender, EventArgs e)
{
     // Crear una ventana con el fondo negro.
     StackPanel panelHelp = new StackPanel();
     panelHelp.Background = new SolidColorBrush(Colors.Black);
     panelHelp.Width = 400;
     panelHelp.Height = 400;

     // Crear un marco blanco.
     Border border = new Border();
     border.BorderBrush = new SolidColorBrush(Colors.White);
     border.BorderThickness = new Thickness(7.0);

     // Crear el botón de cerrar.
     Button close = new Button();
     close.Content = "Cerrar";
     close.Margin = new Thickness(20, 50, 20, 0);
     close.Click += new RoutedEventHandler(close_Click);

     // Create config text and buttons.
     TextBlock textblockHelpTittle = new TextBlock();
     textblockHelpTittle.FontSize = 24;
     textblockHelpTittle.Foreground = new SolidColorBrush(Colors.White);
     textblockHelpTittle.Text = "AYUDA";
     textblockHelpTittle.Margin = new Thickness(160, 10, 0, 0);

     // Crear el título de la ventana, los botones y el texto de ayuda.
     TextBlock textblockHelp = new TextBlock();
     textblockHelp.FontSize = 24;
     textblockHelp.Foreground = new SolidColorBrush(Colors.White);
     textblockHelp.TextWrapping = TextWrapping.Wrap;
     textblockHelp.Text = "Aprieta <Activar GPS/Acc> para ver tu
       posición en el mapa. A continuación selecciona la IP y puerto en el menú
       de configuración y aprieta <Comenzar> para lanzar la aplicación.";
     textblockHelp.Margin = new Thickness(20, 20, 0, 0);
```

```
        // Adición de los elementos a la ventana.
        panelHelp.Children.Add(textblockHelpTittle);
        panelHelp.Children.Add(textblockHelp);
        panelHelp.Children.Add(close);
        border.Child = panelHelp;

        /* Establecer la ventana como propiedad de hijo del marco,
        el cual contendrá la ventana, los textboxs y los botones. */
        help.Child = border;

        // Ajustar la posición de la ventana en la pantalla.
        help.VerticalOffset = 150;
        help.HorizontalOffset = 40;

        // Abrir la ventana y cerrar las otras en caso de estar abiertas.
        if (config.IsOpen) config.IsOpen = false;
        if (iport.IsOpen) iport.IsOpen = false;
        help.IsOpen = true;
}
```

Close_Click

```
/* Función que cierra la ventana help. */
void close_Click(object sender, RoutedEventArgs e)
{
        help.IsOpen = false;
}
```

GPSACCButton_Click

```
/* Función que pone en funcionamiento el GPS y el acelerómetro. Además de
habilitar o deshabilitar ciertos botones. */
private void GPSACCButton_Click(object sender, RoutedEventArgs e)
{
        // Crear elemento GPS y puesta en marcha de éste y el acelerómetro
        GeoWatcher = new GeoCoordinateWatcher(GeoPositionAccuracy.Default);
        GeoWatcher.StatusChanged += GeoWatcher_StatusChanged;
        GeoWatcher.PositionChanged += GeoWatcher_PositionChanged;
        GeoWatcher.Start();
        acc.ReadingChanged += acc_ReadingChanged;
        acc.Start();

        // Modificación del estado de los botones de activo e inactivo.
        StartButton.IsEnabled = true;
        GPSACCButton.IsEnabled = false;
        StopButton.IsEnabled = true;
}
```

StartButton_Click

```
/* Función que envía el mensaje al servidor para empezar a recibir mensajes, pone
en funcionamiento la cámara y modifica los estados de varios botones */
   private void StartButton_Click(object sender, RoutedEventArgs e)
   {

        String szMsj = "    <?xml version='1.0' encoding='utf-8'?> \n" +
               "<!--  Mensaje de EVENTO del usuario sobre el servidor --> \n" +
               "<StatisticMsg> \n" +
               "        <Operacion ID ='Start'> \n" +
               "            <MaquinaID ID='PC202'/> \n" +
               "            <UsuarioID ID='" + this.user.Text + "'/> \n" +
               "            <Password PASS='" + this.contrasena + "%s'/> \n" +
               "        </Operacion> \n" +
               "</StatisticMsg>";
     this.client.Send(szMsj);

     Status.Text = "Activo";
     StartButton.IsEnabled = true;
     StopButton.IsEnabled = true;
     GPSACCButton.IsEnabled = false;

     camera = new CameraCaptureTask();
     camera.Show();
   }
```

StopButton_Click

```
/* Función que para el funcionamiento del GPS y el acelerómetro. Además de
habilitar o deshabilitar ciertos botones. */
private void StopButton_Click(object sender, RoutedEventArgs e)
{
     // Puesta a 0 de los valores de “Longitude” y “Latitude”.
     if (GeoWatcher != null)
     {
          Longit.Text = "0";
          Lat.Text = "0";
          GeoWatcher.Stop();
     }

     // Puesta a 0 de los valores de “X”, “Y” y “Z”
     if (acc != null)
     {
          x.Text = "0";
          y.Text = "0";
          z.Text = "0";
          acc.Stop();
     }
     // Modificación del estado de los botones
     StartButton.IsEnabled = false;
     StopButton.IsEnabled = false;
     GPSACCButton.IsEnabled = true;
 }
```

Acc_ReadingChanged

```
/* Función que detecta los cambios de posición por el aceleróemtro. */
void acc_ReadingChanged(object sender, AccelerometerReadingEventArgs e)
{
     Deployment.Current.Dispatcher.BeginInvoke(() =>
       ThreadSafeAccelerometerChanged(e));
}
```

ThreadSafeAccelerometerChanged

```
/* Función que modifica los valores “X”, “Y” y “Z” del acelerómetro y lo muestra
en pantalla. */
void ThreadSafeAccelerometerChanged(AccelerometerReadingEventArgs e)
{
    this.posX = e.X.ToString("0.000");
    this.posY = e.Y.ToString("0.000");
    this.posZ = e.Z.ToString("0.000");
    x.Text = this.posX;
    y.Text = this.posY;
    z.Text = this.posZ;
}
```

GeoWatcher_PositionChanged

```
/* Función que modifica los valores “Longitude” y “Latitude” y GPS y lo muestra
en pantalla. Además, crea un pushpin (tachuela) para mostrar la posición del
dispositivo en el mapa que está en segundo plano en la pantalla inicial. */
void GeoWatcher_PositionChanged(object sender,
GeoPositionChangedEventArgs<GeoCoordinate> e)
{
     this.lat = e.Position.Location.Latitude.ToString();
     this.longit = e.Position.Location.Longitude.ToString();
     Longit.Text = this.longit;
     Lat.Text = this.lat;

     // Crear pushpin y asignarle los valores recibidos del GPS
     Pushpin pushpin = new Pushpin();
     Location location = new Location();
     location.Latitude = e.Position.Location.Latitude;
     location.Longitude = e.Position.Location.Longitude;
     pushpin.Location = location;
     pushpin.Background = new SolidColorBrush(Colors.Orange);
     pushpin.Content = "1";
     pushpin.FontSize = 30;
     miMapa.Children.Add(pushpin);
}
```

GeoWatcher_StatusChanged

```
/* Función que capta el estado del GPS y lo muestra por pantalla. */
void GeoWatcher_StatusChanged(object sender, GeoPositionStatusChangedEventArgs e)
{
     estadoGPSAcc.Text = e.Status.ToString();
}
```

Código SocketClient.cs

```
using System;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Ink;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;
using System.Net.Sockets;
using System.Threading;
using System.Text;
using System.Net.NetworkInformation;


namespace PFCFinal
{
    public class SocketClient
    {
    /* Cached Socket object that will be used by each call for the lifetime of
       this class */
    Socket _socket = null;

    /* Signaling object used to notify when an asynchronous operation is
       completed. */
    static ManualResetEvent _clientDone = new ManualResetEvent(false);

    /* Define a timeout in milliseconds for each asynchronous call. If a response
       is not received within this. */
    // timeout period, the call is aborted.
    const int TIMEOUT_MILLISECONDS = 5000;

    /* The maximum size of the data buffer to use with the asynchronous socket
       methods. */
    const int MAX_BUFFER_SIZE = 2048;
```

```
    /// <summary>
    /// Attempt a TCP socket connection to the given host over the given port
    /// </summary>
    /// <param name="hostName">The name of the host</param>
    /// <param name="portNumber">The port number to connect</param>
    /// <returns>A string representing the result of this connection
       attempt</returns>
    public string Connect(string hostName, int portNumber)
    {
         string result = string.Empty;

         /* Create DnsEndPoint. The hostName and port are passed in to this
            method. */
         DnsEndPoint hostEntry = new DnsEndPoint(hostName, portNumber);

         /* Create a stream-based, TCP socket using the InterNetwork Address
            Family. */
         _socket = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);

        /* Create a SocketAsyncEventArgs object to be used in the connection
           request. */
        SocketAsyncEventArgs socketEventArg = new SocketAsyncEventArgs();
        socketEventArg.RemoteEndPoint = hostEntry;

        // Inline event handler for the Completed event.
        /* Note: This even handler was implemented inline in order to make this
           method self-contained. */
        socketEventArg.Completed += new
EventHandler<SocketAsyncEventArgs>(delegate(object s, SocketAsyncEventArgs e)
        {
        // Retrieve the result of this request
        result = e.SocketError.ToString();

        // Signal that the request is complete, unblocking the UI thread
        _clientDone.Set();
        });

        // Sets the state of the event to nonsignaled, causing threads to block
        _clientDone.Reset();

        // // Make an asynchronous Connect request over the socket
        _socket.ConnectAsync(socketEventArg);

        // Block the UI thread for a maximum of TIMEOUT_MILLISECONDS seconds.
        // If no response comes back within this time then proceed
        _clientDone.WaitOne(TIMEOUT_MILLISECONDS);

         return result;
   }
```

```
        /// <summary>
        /// Send the given data to the server using the established connection
        /// </summary>
        /// <param name="data">The data to send to the server</param>
        /// <returns>The result of the Send request</returns>
        public string Send(string data)
        {
        string response = "Operation Timeout";

        /* We are re-using the _socket object that was initialized in the Connect
           method */
        if (_socket != null)
        {
             // Create SocketAsyncEventArgs context object
             SocketAsyncEventArgs socketEventArg = new SocketAsyncEventArgs();

             // Set properties on context object
             socketEventArg.RemoteEndPoint = _socket.RemoteEndPoint;
             socketEventArg.UserToken = null;

             // Inline event handler for the Completed event.
             /* Note: This even handler was implemented inline in order to make
                this method self-contained.*/
            socketEventArg.Completed += new
EventHandler<SocketAsyncEventArgs>(delegate(object s, SocketAsyncEventArgs e)
            {
                 response = e.SocketError.ToString();

                 // Unblock the UI thread
                 _clientDone.Set();
            });

            // Add the data to be sent into the buffer
            byte[] payload = Encoding.UTF8.GetBytes(data);
            socketEventArg.SetBuffer(payload, 0, payload.Length);

            //Sets the state of the event to nonsignaled, causing threads to block
            _clientDone.Reset();

            // Make an asynchronous Send request over the socket
            _socket.SendAsync(socketEventArg);

            // Block the UI thread for a maximum of TIMEOUT_MILLISECONDS seconds.
            // If no response comes back within this time then proceed
            _clientDone.WaitOne(TIMEOUT_MILLISECONDS);
        }
        else
        {
             response = "Socket is not initialized";
        }

        return response;
    }
```

```
        /// <summary>
        /// Receive data from the server using the established socket connection
        /// </summary>
        /// <returns>The data received from the server</returns>
        public string Receive()
        {
        string response = "Operation Timeout";

        // We are receiving over an established socket connection
        if (_socket != null)
        {
             // Create SocketAsyncEventArgs context object
             SocketAsyncEventArgs socketEventArg = new SocketAsyncEventArgs();
             socketEventArg.RemoteEndPoint = _socket.RemoteEndPoint;

             // Setup the buffer to receive the data
             socketEventArg.SetBuffer(new Byte[MAX_BUFFER_SIZE], 0,
MAX_BUFFER_SIZE);

             // Inline event handler for the Completed event.
             /* Note: This even handler was implemented inline in order to make
                this method self-contained. */
             socketEventArg.Completed += new
EventHandler<SocketAsyncEventArgs>(delegate(object s, SocketAsyncEventArgs e)
             {
             if (e.SocketError == SocketError.Success)
             {
                  // Retrieve the data from the buffer
                  response = Encoding.UTF8.GetString(e.Buffer, e.Offset,
                 e.BytesTransferred);
                  response = response.Trim('\0');
             }
             else
             {
                  response = e.SocketError.ToString();
             }
                  _clientDone.Set();
             });

        // Sets the state of the event to nonsignaled, causing threads to block
        _clientDone.Reset();

        // Make an asynchronous Receive request over the socket
        _socket.ReceiveAsync(socketEventArg);

        // Block the UI thread for a maximum of TIMEOUT_MILLISECONDS seconds.
        // If no response comes back within this time then proceed
        _clientDone.WaitOne(TIMEOUT_MILLISECONDS);
        }
        else
        {
            response = "Socket is not initialized";
        }

            return response;
        }


    }
}
```

ANEXO III

Pequeños proyectos para el aprendizaje del lenguaje C#

Lector del timeline de Twitter

Código MainPage.xaml

```
<phone:PhoneApplicationPage x:Class="Dia25_ServiciosDeWeb.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:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"
FontFamily="{StaticResource PhoneFontFamilyNormal}"
FontSize="{StaticResource PhoneFontSizeNormal}"
Foreground="{StaticResource PhoneForegroundBrush}"
SupportedOrientations="Portrait" Orientation="Portrait"
shell:SystemTray.IsVisible="True">

    <!--LayoutRoot es la cuadrícula raíz donde se coloca todo el contenido de la
página-->
    <Grid x:Name="LayoutRoot"
          Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto" />
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>

        <!--TitlePanel contiene el nombre de la aplicación y el título de la
página-->
        <StackPanel x:Name="TitlePanel"
                    Grid.Row="0"
                    Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle"
                       Text="Prueba 1"
                       Style="{StaticResource PhoneTextNormalStyle}" />
            <TextBlock x:Name="PageTitle"
                       Text="Lector Twitter"
                       Margin="9,-7,0,0"
                       Style="{StaticResource PhoneTextTitle1Style}" />
        </StackPanel>

        <!--ContentPanel. Colocar aquí el contenido adicional-->
        <Grid x:Name="ContentPanel"
              Grid.Row="1"
              Margin="12,0,12,0">
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="350" />
                <ColumnDefinition Width="100" />
            </Grid.ColumnDefinitions>
            <Grid.RowDefinitions>
                <RowDefinition Height="100" />
                <RowDefinition Height="100" />
                <RowDefinition Height="*" />
            </Grid.RowDefinitions>
            <TextBox x:Name="TwitterNameBox" />
            <Button x:Name="GoButton"
                    Content="Ir"
                    Grid.Column="1"
```

```
                    Click="GoButton_Click" />


<StackPanel Orientation="Horizontal"
                        Grid.Row="1"
                        Grid.ColumnSpan="2">
                <Image x:Name="TwitterImage"
                       Width="100"
                       Height="100"
                       Margin="0,0,12,0" />
                <TextBlock x:Name="TwitterName"
                           Foreground="Orange"
                           FontSize="50" />
            </StackPanel>
            <ListBox x:Name="TwitterList"
                     Grid.Row="2"
                     Grid.ColumnSpan="2">
                <ListBox.ItemTemplate>
                    <DataTemplate>
                        <StackPanel>
                            <TextBlock Margin="0,0,40,0"
                                       Text="{Binding message}"
                                       TextWrapping="Wrap"
                                       Width="450" />
                            <Rectangle Height="20" />
                        </StackPanel>
                    </DataTemplate>
                </ListBox.ItemTemplate>
            </ListBox>
        </Grid>
    </Grid>

    <!--Código de ejemplo que muestra el uso de ApplicationBar-->
    <!--<phone:PhoneApplicationPage.ApplicationBar>
        <shell:ApplicationBar IsVisible="True" IsMenuEnabled="True">
            <shell:ApplicationBarIconButton IconUri="/Images/appbar_button1.png"
Text="Botón 1"/>
            <shell:ApplicationBarIconButton IconUri="/Images/appbar_button2.png"
Text="Botón 2"/>
            <shell:ApplicationBar.MenuItems>
                <shell:ApplicationBarMenuItem Text="Elemento de menú 1"/>
                <shell:ApplicationBarMenuItem Text="Elemento de menú 2"/>
            </shell:ApplicationBar.MenuItems>
        </shell:ApplicationBar>
    </phone:PhoneApplicationPage.ApplicationBar>-->

</phone:PhoneApplicationPage>
```

Código MainPage.xaml.cs

```
using System;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Xml.Linq;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Net.NetworkInformation;

namespace Dia25_ServiciosDeWeb
{
    public partial class MainPage : PhoneApplicationPage
    {
        // Constructor
        public MainPage()
        {
            InitializeComponent();
        }

        private void GoButton_Click(object sender, RoutedEventArgs e)
        {
            if (NetworkInterface.GetIsNetworkAvailable())
            {
                WebClient twitter = new WebClient();

              twitter.DownloadStringCompleted += twitter_DownloadStringCompleted;
              twitter.DownloadStringAsync(new
Uri("http://api.twitter.com/1/statuses/user_timeline.xml?screen_name=" +
TwitterNameBox.Text));
            }
        }

void twitter_DownloadStringCompleted(object sender,
DownloadStringCompletedEventArgs e)
        {
            if (e.Error != null) return;

            XElement xmlTweets = XElement.Parse(e.Result);

           string image =
xmlTweets.Element("status").Element("user").Element("profile_image_url").Value;
            ImageSource source = new BitmapImage(new Uri(image));
            TwitterImage.Source = source;

            string name =
xmlTweets.Element("status").Element("user").Element("screen_name").Value;
            TwitterName.Text = name;

 TwitterList.ItemsSource = from tweet in xmlTweets.Descendants("status")
                                      select new TwitterItem
                                      {
                                          message = tweet.Element("text").Value
                                      };
        }
    }
}
```

GPS

Código MainPage.xaml

```
<phone:PhoneApplicationPage x:Class="Dia13_ServiciosDeUbicacion.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:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
xmlns:maps="clr-namespace:Microsoft.Phone.Controls.Maps;
assembly=Microsoft.Phone.Controls.Maps" mc:Ignorable="d"
                          d:DesignWidth="480"
                          d:DesignHeight="768"
                          FontFamily="{StaticResource PhoneFontFamilyNormal}"
                          FontSize="{StaticResource PhoneFontSizeNormal}"
                          Foreground="{StaticResource PhoneForegroundBrush}"
                          SupportedOrientations="Portrait"
                          Orientation="Portrait"
                          shell:SystemTray.IsVisible="True">

    <!--LayoutRoot es la cuadrícula raíz donde se coloca todo el contenido de la
página-->
    <Grid x:Name="LayoutRoot"
          Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto" />
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>

        <!--TitlePanel contiene el nombre de la aplicación y el título de la
página-->
        <StackPanel x:Name="TitlePanel"
                    Grid.Row="0"
                    Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle"
                       Text="Prueba 2"
                       Style="{StaticResource PhoneTextNormalStyle}" />
            <TextBlock x:Name="PageTitle"
                       Text="GPS"
                       Margin="9,-7,0,0"
                       Style="{StaticResource PhoneTextTitle1Style}" />
        </StackPanel>

        <!--ContentPanel. Colocar aquí el contenido adicional-->
        <Grid x:Name="ContentPanel"
              Grid.Row="1"
              Margin="12,0,12,0">
            <CheckBox x:Name="HighBox"
                      Content="Usar GeoPositionAccuracy.High"
                      Height="72"
                      HorizontalAlignment="Left"
                      Margin="6,6,0,0"
                      VerticalAlignment="Top"
                      Width="444" />
```

```
<Button x:Name="StartButton"
        Content="Comenzar"
        Height="72"
        HorizontalAlignment="Left"
        Margin="12,84,0,0"
        VerticalAlignment="Top"
        Width="160"
        Click="StartButton_Click"
        FontSize="20" />
<Button Content="Parar"
        Height="72"
        HorizontalAlignment="Left"
        Margin="159,84,0,0"
        Name="StopButton"
        VerticalAlignment="Top"
        Width="160"
        Click="StopButton_Click"
        IsEnabled="False"
        FontSize="22" />
<TextBlock Height="30"
           HorizontalAlignment="Left"
           Margin="12,201,0,0"
           Name="LatitudeLabel"
           Text="Latitud"
           VerticalAlignment="Top" />
<TextBlock Height="30"
           HorizontalAlignment="Left"
           Margin="12,339,0,0"
           Name="LongitudeLabel"
           Text="Longitud"
           VerticalAlignment="Top" />
<TextBlock Height="30"
           HorizontalAlignment="Left"
           Margin="12,476,0,0"
           Name="AccuracyLabel"
           Text="Exactitud"
           VerticalAlignment="Top" />
<TextBlock Height="101"
           HorizontalAlignment="Left"
           Margin="12,237,0,0"
           Name="Latitude"
           Text=""
           VerticalAlignment="Top"
           Width="438"
           FontSize="72" />
<TextBlock Height="101"
           HorizontalAlignment="Left"
           Margin="12,375,0,0"
           Name="Longitude"
           Text=""
           VerticalAlignment="Top"
           Width="438"
           FontSize="72" />
```

```
            <TextBlock Height="101"
                       HorizontalAlignment="Left"
                       Margin="12,506,0,0"
                       Name="Accuracy"
                       Text=""
                       VerticalAlignment="Top"
                       Width="438"
                       FontSize="72" />
            <TextBlock Height="30"
                       HorizontalAlignment="Left"
                       Margin="73,162,0,0"
                       Name="Status"
                       Text="Inactivo"
                       VerticalAlignment="Top"
                       Width="306"
                       TextAlignment="Center" />
            <Button Content="Ver mapa"
                    Height="72"
                    HorizontalAlignment="Left"
                    IsEnabled="False"
                    Margin="308,84,0,0"
                    Name="MapButton"
                    VerticalAlignment="Top"
                    Width="160"
                    Click="MapButton_Click"
                    FontSize="20" />
            <maps:Map x:Name="Map"
                    Visibility="Collapsed"
                    Height="768"
                    HorizontalAlignment="Left"
                    Margin="-12,0,0,0"
                    VerticalAlignment="Bottom"
                    Width="480" />
        </Grid>
    </Grid>

    <!--Código de ejemplo que muestra el uso de ApplicationBar-->
    <!--<phone:PhoneApplicationPage.ApplicationBar>
        <shell:ApplicationBar IsVisible="True" IsMenuEnabled="True">
            <shell:ApplicationBarIconButton IconUri="/Images/appbar_button1.png"
Text="Botón 1"/>
            <shell:ApplicationBarIconButton IconUri="/Images/appbar_button2.png"
Text="Botón 2"/>
            <shell:ApplicationBar.MenuItems>
                <shell:ApplicationBarMenuItem Text="Elemento de menú 1"/>
                <shell:ApplicationBarMenuItem Text="Elemento de menú 2"/>
            </shell:ApplicationBar.MenuItems>
        </shell:ApplicationBar>
    </phone:PhoneApplicationPage.ApplicationBar>-->

</phone:PhoneApplicationPage>
```

Código MainPage.xaml.cs

```
using System.Device.Location;
using System.Windows;
using Microsoft.Phone.Controls;

namespace Dia13_ServiciosDeUbicacion
{
    public partial class MainPage : PhoneApplicationPage
    {
        private GeoCoordinateWatcher GeoWatcher;

        // Constructor
        public MainPage()
        {
            InitializeComponent();
        }

              void GeoWatcher_PositionChanged(object sender,
GeoPositionChangedEventArgs<GeoCoordinate> e)
              {
                     Latitude.Text = e.Position.Location.Latitude.ToString();
                     Longitude.Text = e.Position.Location.Longitude.ToString();
                     Accuracy.Text =
e.Position.Location.HorizontalAccuracy.ToString();
                     MapButton.IsEnabled = true;
              }

              void GeoWatcher_StatusChanged(object sender,
GeoPositionStatusChangedEventArgs e)
              {
                     Status.Text = e.Status.ToString();
              }

              private void StartButton_Click(object sender, RoutedEventArgs e)
              {
                     if (HighBox.IsChecked == true)
                     GeoWatcher = new
GeoCoordinateWatcher(GeoPositionAccuracy.High);
                     else
                            GeoWatcher = new
GeoCoordinateWatcher(GeoPositionAccuracy.Default);

                     GeoWatcher.StatusChanged += GeoWatcher_StatusChanged;
                     GeoWatcher.PositionChanged += GeoWatcher_PositionChanged;
                     GeoWatcher.Start();

                     StartButton.IsEnabled = false;
                     StopButton.IsEnabled = true;
              }
```

```
private void StopButton_Click(object sender, RoutedEventArgs e)
		{
			if (GeoWatcher != null) GeoWatcher.Stop();

			StartButton.IsEnabled = true;
			StopButton.IsEnabled = false;

			Latitude.Text = "";
			Longitude.Text = "";
			Status.Text = "Inactivo";
		}

		private void MapButton_Click(object sender, RoutedEventArgs e)
		{
			Map.Visibility = Visibility.Visible;
			Map.Center = new
GeoCoordinate(GeoWatcher.Position.Location.Latitude,
GeoWatcher.Position.Location.Longitude);
			Map.ZoomLevel = 17;
			Map.ZoomBarVisibility = Visibility.Visible;
			Map.ScaleVisibility = Visibility.Visible;
		}
	}
}
```

Acelerómetro

Código MainPage.xaml

```
<phone:PhoneApplicationPage x:Class="Dia11_Acelerometro.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:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"
FontFamily="{StaticResource PhoneFontFamilyNormal}"
FontSize="{StaticResource PhoneFontSizeNormal}"
Foreground="{StaticResource PhoneForegroundBrush}"
SupportedOrientations="Portrait" Orientation="Portrait"
shell:SystemTray.IsVisible="True">

    <!--LayoutRoot es la cuadrícula raíz donde se coloca todo el contenido de la
página-->
    <Grid x:Name="LayoutRoot"
          Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto" />
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>

        <!--TitlePanel contiene el nombre de la aplicación y el título de la
página-->
        <StackPanel x:Name="TitlePanel"
                    Grid.Row="0"
                    Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle"
                       Text="Prueba 3"
                       Style="{StaticResource PhoneTextNormalStyle}" />
            <TextBlock x:Name="PageTitle"
                       Text="Acelerómetro"
                       Margin="9,-7,0,0"
                       Style="{StaticResource PhoneTextTitle1Style}" />
        </StackPanel>

        <!--ContentPanel. Colocar aquí el contenido adicional-->
        <Grid x:Name="ContentPanel"
              Grid.Row="1"
              Margin="12,0,12,0">
            <TextBlock Text="X="
                       FontSize="100"
                       Margin="5,0,-5,0" />

            <TextBlock Text="Y="
                       FontSize="100"
                       Margin="5,106,-5,-106" />
            <TextBlock Text="Z="
                       FontSize="100"
                       Margin="5,206,-5,-206" />
```

```
            <TextBlock Name="X"
                       Text="0"
                       FontSize="100"
                       Margin="140,0,-138,0" />
            <TextBlock Name="Y"
                       Text="0"
                       FontSize="100"
                       Margin="140,106,-139,-106" />
            <TextBlock Name="Z"
                       Text="0"
                       FontSize="100"
                       Margin="140,206,-140,-206" />

        </Grid>
    </Grid>

    <!--Código de ejemplo que muestra el uso de ApplicationBar-->
    <!--<phone:PhoneApplicationPage.ApplicationBar>
        <shell:ApplicationBar IsVisible="True" IsMenuEnabled="True">
            <shell:ApplicationBarIconButton IconUri="/Images/appbar_button1.png"
Text="Botón 1"/>
            <shell:ApplicationBarIconButton IconUri="/Images/appbar_button2.png"
Text="Botón 2"/>
            <shell:ApplicationBar.MenuItems>
                <shell:ApplicationBarMenuItem Text="Elemento de menú 1"/>
                <shell:ApplicationBarMenuItem Text="Elemento de menú 2"/>
            </shell:ApplicationBar.MenuItems>
        </shell:ApplicationBar>
    </phone:PhoneApplicationPage.ApplicationBar>-->

</phone:PhoneApplicationPage>
```

Código MainPage.xaml.cs

```
using System;
using System.Windows;
using Microsoft.Devices.Sensors;
using Microsoft.Phone.Controls;

namespace Dia11_Acelerometro
{
    public partial class MainPage : PhoneApplicationPage
    {
        Accelerometer acc = new Accelerometer();

        // Constructor
        public MainPage()
        {
            InitializeComponent();
            acc.ReadingChanged += acc_ReadingChanged;
            acc.Start();
        }

        void acc_ReadingChanged(object sender, AccelerometerReadingEventArgs e)
        {
            Deployment.Current.Dispatcher.BeginInvoke(() =>
ThreadSafeAccelerometerChanged(e));
        }

        void ThreadSafeAccelerometerChanged(AccelerometerReadingEventArgs e)
        {
            X.Text = e.X.ToString("0.000");
            Y.Text = e.Y.ToString("0.000");
            Z.Text = e.Z.ToString("0.000");
        }
    }
}
```

Programa sencillo de fotografías

De esta aplicación solo se mostrará una parte del MainPage.xaml.cs que es de donde se sacaron las ideas para la aplicación del proyecto.

Código 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.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Media.Animation;
using System.Windows.Shapes;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;
using Microsoft.Phone.Tasks;
using Microsoft.Phone;
using System.IO;
using System.IO.IsolatedStorage;
using System.Windows.Controls.Primitives;

namespace PhotosSample
{
    public partial class MainPage : PhoneApplicationPage
    {

        //This is a variable for the help popup.
        Popup help = new Popup();

        //The application bar buttons that are used.
        ApplicationBarIconButton btnCamera;
        ApplicationBarIconButton btnCrop;
        ApplicationBarIconButton btnHelp;

        //The camera chooser used to capture a picture.
        CameraCaptureTask ctask;

        // Constructor
        public MainPage()
        {
            InitializeComponent();

            SupportedOrientations = SupportedPageOrientation.Portrait |
SupportedPageOrientation.Landscape;
```

```
 //Creates an application bar and then sets visibility and menu properties.
            ApplicationBar = new ApplicationBar();
            ApplicationBar.IsVisible = true;


            //This code creates the application bar icon buttons.
            btnCamera = new ApplicationBarIconButton(new
Uri("/Icons/appbar.feature.camera.rest.png", UriKind.Relative));
            btnCrop = new ApplicationBarIconButton(new
Uri("/Icons/appbar.edit.rest.png", UriKind.Relative));
            btnHelp = new ApplicationBarIconButton(new
Uri("/Icons/appbar.questionmark.rest.png", UriKind.Relative));

            //Labels for the application bar buttons.
            btnCamera.Text = "Camera";
            btnCrop.Text = "Crop";
            btnHelp.Text = "Help";


            //This code will create event handlers for buttons.
            btnCamera.Click += new EventHandler(btnCamera_Click);
            btnCrop.Click += new EventHandler(btnCrop_Click);
            btnHelp.Click += new EventHandler(btnHelp_Click);

            //This code adds buttons to application bar.
            ApplicationBar.Buttons.Add(btnCamera);
            ApplicationBar.Buttons.Add(btnCrop);
            ApplicationBar.Buttons.Add(btnHelp);

            //Disable crop button until photo is taken.
            btnCrop.IsEnabled = false;

            textStatus.Text = "Tap the camera button to take a picture.";

            //Create new instance of CameraCaptureClass
            ctask = new CameraCaptureTask();

            //Create new event handler for capturing a photo
            ctask.Completed += new EventHandler<PhotoResult>(ctask_Completed);
        }

        /// <summary>
        /// Click event handler for the help button.
        ///This will create a popup/message box for help and add content to the
popup.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        void btnHelp_Click(object sender, EventArgs e)
        {
            //Stack panel with a black background.
            StackPanel panelHelp = new StackPanel();
            panelHelp.Background = new SolidColorBrush(Colors.Black);
            panelHelp.Width = 300;
            panelHelp.Height = 400;

            //Create a white border.
            Border border = new Border();
            border.BorderBrush = new SolidColorBrush(Colors.White);
            border.BorderThickness = new Thickness(7.0);
```

```
            //Create a close button to exit popup.
            Button close = new Button();
            close.Content = "Close";
            close.Margin = new Thickness(5.0);
            close.Click += new RoutedEventHandler(close_Click);

            //Create helper text
            TextBlock textblockHelp = new TextBlock();
            textblockHelp.FontSize = 24;
            textblockHelp.Foreground = new SolidColorBrush(Colors.White);
            textblockHelp.TextWrapping = TextWrapping.Wrap;
            textblockHelp.Text = "Tap the camera button image on the application
bar to take a photo." + " Once the photo is taken and returned to this page, tap
the crop button on the application bar to crop the image.";
            textblockHelp.Margin = new Thickness(5.0);

            //Add controls to stack panel
            panelHelp.Children.Add(textblockHelp);
            panelHelp.Children.Add(close);
            border.Child = panelHelp;

            // Set the Child property of Popup to the border
            // that contains a stackpanel, textblock and button.
            help.Child = border;

            // Set where the popup will show up on the screen.
            help.VerticalOffset = 200;
            help.HorizontalOffset = 85;

            // Open the popup.
            help.IsOpen = true;

        }

        /// <summary>
        /// Click event handler for the close button on the help popup.
        /// Closes the poupup.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        void close_Click(object sender, RoutedEventArgs e)
        {
            help.IsOpen = false;
        }
```