UNIVERSIDAD POLITECNICA DE VALENCIA

ESCUELA POLITECNICA SUPERIOR DE GANDIA

Grado en Ing. Sist. de Telecom., Sonido e Imagen

UNIVERSIDAD
POLITECNICA
DE VALENCIA

ESCUELA POLITECNICA
SUPERIOR DE GANDIA

“Estudio de la herramienta 'RAD Studio' para el desarrollo de aplicaciones multiplataforma en Android, iOS, Mac y Windows”

TRABAJO FINAL DE GRADO

Autor/a:
Luisa María Leboso Ruiz

Tutor/a:
Jesús Tomás Gironés

GANDIA, 2016

Índice

Resumen del Trabajo final de grado.

La convivencia de diferentes tipos de **plataformas informáticas**, hace que el desarrollo de una **aplicación** que se desea ejecutar en varias de estas plataformas sea una tarea que requiere de diferentes equipos de trabajo especializados cada uno de ellos en cada una de estas plataformas. Para facilitar esta tarea los fabricantes de herramientas de desarrollo software ofrecen herramientas de desarrollo **multiplataforma**.

RAD Studio es un **IDE (Entorno de desarrollo integrado)** multiplataforma fabricado por Embarcadero, que ofrece la posibilidad de desarrollar aplicaciones para plataformas móviles (iOS y Android) o de ordenadores personales (Windows y Mac).

En el presente trabajo final de grado se realizará una presentación de las principales características y herramientas de este IDE, además se realizará una introducción sobre cómo trabajar con él y se realizará una aplicación de ejemplo que se ejecutará en diferentes plataformas.

The coexistence of different kinds of **computing platforms**, makes it difficult to develop a cross-platform **application** because it needs several specialist teams each working on a different platform. To facilitate this, the makers of software development tools offer **cross-platform** development tools.

RAD Studio is an **IDE (integrated development environment)** for cross-platform applications manufactured by Embarcadero, that offers the possibility to develop applications for mobile platforms (iOS and Android) or PC platforms (Windows and Mac).

This final degree makes a presentation of the main features and tools of this IDE, and also, an overview of how to work with it to make an example application that will run on different platforms.

1. Introducción y objetivos.

La situación actual del mercado tecnológico presenta la convivencia de diferentes plataformas informáticas, cada una de ellas con sus características y lenguajes de desarrollo diferentes. Esta situación supone un problema para los desarrolladores de aplicaciones software, ya que deben desarrollar diferentes versiones de una misma aplicación si quieren llevarla al máximo número de usuarios posible. Esto les supone un incremento del esfuerzo, tanto en el desarrollo, como en el mantenimiento y la evolución de sus aplicaciones.

Dada esta situación, los fabricantes de herramientas de desarrollo software están desarrollando herramientas que permitan el desarrollo de una aplicación en diferentes plataformas de la manera más directa posible, reduciendo los esfuerzos de los desarrolladores de aplicaciones.

El objetivo de este trabajo fin de grado es el estudio de RAD Studio, uno de los entornos de desarrollo integrado (IDE) multiplataforma que actualmente podemos encontrar en el mercado. Para probar las herramientas de trabajo que ofrece este IDE se ha desarrollado una aplicación multiplataforma y se ha probado su ejecución en diferentes plataformas.

La complejidad de esta aplicación está limitada por la compatibilidad entre plataformas y de las herramientas que ofrece RAD Studio para aumentar esta compatibilidad. Por tanto, el objetivo de este TFG es conseguir una aplicación que funcione de la misma manera en las diferentes plataformas, y no el desarrollo de una aplicación más o menos compleja.

2. Introducción a las plataformas informáticas.

Algunos usuarios de tecnologías informáticas confunden plataforma informática con sistema operativo o con arquitectura hardware. No están muy desencaminados, pero su concepto de plataforma está incompleto. Pues, en informática, ***plataforma*** es un término global que hace referencia a todo el software y hardware con el cual una aplicación informática o un componente informático es compatible.

La plataforma está definida por un estándar alrededor del cual se determina una arquitectura de hardware y una plataforma de software. Al definir una plataforma, se debe establecer el tipo de arquitectura, sistema operativo (en adelante SO), lenguaje de programación o interfaz de usuario compatibles con ella.

Ejemplos de plataformas informáticas para PC, son *IBM-PC*, que incluye las arquitecturas I386(x86), IA64 o AMD64 (x86-64), sobre la que trabajaría alguno de los SO de Windows o Linux; y *Macintosh*, que incluye la arquitectura Gecko y PowerPC, sobre la que trabajaría alguno de los SO de Mac o Linux.

2.1. Windows.

La historia de Windows comienza el año 1975, cuando Gates y Allen crearon la compañía Microsoft. En 1981 empezaron a comercializar equipos IBM que ejecutaban el sistema MS-DOS. MS-DOS es un sistema operativo muy robusto y eficaz, pero de difícil comprensión para los usuarios recién iniciados a la informática que requieren de mucho tiempo de aprendizaje para dominarlo. Así que vieron necesaria la creación de un sistema operativo más sencillo y con un tiempo de aprendizaje más rápido.

En 1982, comenzaron el desarrollo de su nuevo sistema operativo llamado *Interface Manager*. En él eliminaron los comandos de MS-DOS e introdujeron el uso del "ratón" con el que el usuario se desplaza a través de menús desplegables, iconos, cuadros de diálogo y "ventanas" informáticas. Fueron estas ventanas las que cambiaron el nombre original del sistema operativo, que salió finalmente a la venta en 1985 con el nombre de ***Windows 1.0.***

Durante la segunda mitad de la década de los 80, los equipos informáticos comienzan a ser una herramienta fundamental en muchas empresas aumentando el número de usuarios de computadoras en el mundo. Además la investigación y desarrollo hardware mejoran la calidad y rapidez de los procesadores y la capacidad de las memorias virtuales.

Así en las versiones **2** (1988) y **3** (1990) de Windows, aprovechando las mejoras técnicas, se mejoraron los gráficos (16 colores) y aumentó la velocidad de ejecución de los programas; además se introdujeron nuevas herramientas que facilitaban el aprendizaje de los nuevos usuarios, como el Administrador de programas, el Administrador de archivos y el Administrador de impresión.

A principios de los 90, en el ámbito profesional y científico, aparecen nuevas necesidades. Como los grupos de trabajo punto a punto y redes de dominio (que

permitía la versión ***Windows 3.11,*** por primera vez los equipos con Windows entran a formar parte del entorno cliente/servidor). O como el aumento de la información a procesar por los programas, en 1993 se lanzó ***Windows NT*** de 32 bits, compatible con programas científicos y de ingeniería de última generación.

También durante los 90, los ordenadores empiezan a introducirse en los hogares como un aparato electrónico básico más. Sobre todo como herramienta para el estudio y para el ocio (videojuegos). Así que el número de usuarios sin formación informática siguió en aumento. Además, en la segunda mitad de los 90, las conexiones a Internet empiezan a popularizarse tanto en el ámbito profesional como en el doméstico. Así, para hacer frente a las necesidades de los usuario, en las versiones ***Windows 95*** (agosto de 1995) y ***Windows 98*** (1998) se fueron añadiendo mejoras:

- en la navegabilidad (menú Inicio, barra de tareas, botones minimizar/cerrar ventana),
- en las conexiones del equipo con redes o periféricos (compatibilidad integrada con Internet, conexión de red por acceso telefónico, funciones Plug and Play, compatibilidad con discos DVD y dispositivos USB...),
- en ayudas al usuario (asistentes para la búsqueda de información en los equipos y en Internet, traducciones a 25 idiomas...)
- y en las nuevas tecnologías orientadas al ocio multimedia (reproductor de Windows Media, funciones para fotografía digital, ...).

En la primera década del siglo XXI, los ordenadores ya están presentes en todos los ámbitos, lo que supone un problema en la creación de una versión única de un sistema operativo. Ya que un sistema operativo que cumple con las necesidades del ámbito científico puede ser demasiado complejo para su uso en el ámbito doméstico. Así pues, para la versión de **Windows XP** (2001) se desarrollaron diferentes ediciones según su destinatario:

- Edición Professional: con herramientas para el trabajo a distancia (Escritorio remoto), sistema de archivos cifrado, compatibilidad con redes inalámbricas 802.1x, Windows Messenger y Asistencia remota.
- Edición Professional de 64 bits (2001), para trabajar con grandes volúmenes de memoria y proyectos, como programas de efectos especiales de películas, de animaciones en 3D, de ingeniería y científicos.
- Windows XP Media Center Edition (2002), para entretenimiento doméstico.
- Windows XP Tablet PC Edition (2002). Los equipos Tablet PC incluyen un lápiz digital para el reconocimiento de la escritura manual.

La realización de diferentes ediciones de una misma versión del SO se mantuvo en las siguientes versiones de Windows:

- ***Windows Vista*** (2006): aumentó los idiomas de presentación hasta los 35, incidió en el uso del PC como centro multimedia y modernizó el diseño de las ventanas.
- ***Windows 7*** (2009) pensado para las tecnologías inalámbricas y las nuevas pantallas táctiles.

En la segunda década del siglo XXI, los dispositivos móviles han ganado terreno a los ordenadores y Windows decide adaptar su SO a estos nuevos dispositivos. Una de las medidas tomadas con este fin, fue el desarrollo de apps de Windows que se consiguen exclusivamente desde la Tienda Windows.

Así en 2012 se lanzó ***Windows 8***, con una interfaz completamente nueva que actúa como una tableta (compatible con las apps de Windows) para el ocio y como un equipo con el Escritorio Windows clásico para el trabajo. En paralelo, se lanzó ***Windows RT***, para tabletas y PCs ligeros, diseñado para aumentar la duración de la batería, compatible exclusivamente con apps de la Tienda Windows.

En 2013 se presentó ***Windows 8.1***, que permite más opciones de personalización de la pantalla Inicio y la sincronización entre dispositivos. Además del uso de varias pantallas (permite tener una app diferente en hasta cuatro monitores conectados al equipo). Además mejora la conexión a los recursos corporativos con la Conexión a área de trabajo y Carpetas de trabajo.

A principios de 2015, Microsoft cambia la estrategia de desarrollo de su SO, iniciando el programa Windows Insider, que consiste en el lanzamiento gratuito de la primera versión ***Windows 10*** para su "testeo" por parte de los clientes inscritos en el programa, quienes lo evaluarán y contribuirán en el desarrollo con sus sugerencias. Esta nueva estrategia se basa en el desarrollo de evoluciones y actualizaciones automáticas frecuentes en vez de en grandes versiones con plazos mayores. Entre las novedades de Windows 10 destacan su nueva interfaz (centrada en el menú Inicio), "*Cortana*" (en 6 idiomas), la primera asistente digital de Microsoft (incluida también en los teléfonos con Windows 8.1), y sus 9 ediciones (Home, Pro, Enterprise, Enterprise LTSB, Education, Mobile, Mobile Enterprise, IoT Edition y N y KN Edition)

2.2. Mac.

La historia de Mac se desarrolla en paralelo a la de Windows, por lo que a continuación se detallarán las evoluciones más destacadas de Mac sin explicar de nuevo la evolución del mercado a lo largo de los años.

El proyecto Macintosh comenzó en 1979, de manos de Jef Rasking, con la finalidad de buscar una computadora de bajo precio y de fácil uso para el cliente promedio. El proyecto pasó a manos de Steve Jobs en 1981.

La primera versión del Mac OS, llamada ***System***, fue lanzada en 1984, en ese momento era el único SO que usaba una interfaz gráfica de usuario en vez de la línea de comandos. System incluía *Finder*, una aplicación usada para la administración de archivos y para mostrar el escritorio. Los dos archivos estaban contenidos en una carpeta etiquetada como "System Folder" (carpeta del sistema), la cual contenía otros archivos necesarios para interactuar con System, como el controlador de la impresora.

En las ediciones ***System 1, 2*** (1985)***, 3*** (1986) y ***4*** (marzo 1987), el SO sólo podía ejecutar una aplicación. Para ejecutar varias aplicaciones a la vez, hacía falta el uso de aplicaciones adicionales desarrolladas por terceros como *Servant*, *MultiMac* o

Switcher. Además las versiones 1.0, 1.1 y 2.0 usaban un sistema de archivos con un sólo nivel de directorios, el *Macintosh File System* (MFS), por tanto su soporte para carpetas (subdirectorios) estaba incompleto. Todo esto complicaba la personalización y optimización del sistema a los usuarios con menor experiencia.

Para la versión ***System Software 5*** (octubre 1987)***,*** se desarrolló *MultiFinder*, una extensión opcional que permitía al sistema ejecutar varios programas al mismo tiempo con la multitarea cooperativa (se cede tiempo a las aplicaciones en segundo plano sólo cuando la aplicación que se ejecuta en primer plano cede el control). Esta versión estuvo disponible poco tiempo y sólo en algunos países.

System Software 6 (1987), fue una versión más consolidada del Mac OS produciendo un SO completo, estable y de larga duración (se mantuvo hasta principios de 1992). A partir de esta versión *System* y *Finder* unificaron su número de versión. A lo largo de su vida se crearon 9 evoluciones, en la evolución 6.0.4 se presentó ***Macintosh Portable*** para ordenadores portátiles.

En mayo de 1991 salió ***System 7.*** Con una interfaz gráfica de usuario renovada, con mejoras en la estabilidad y nuevas características como:

- el soporte para intercambio de página, *
- el administrador de extensiones "*Extension Manager"* (para la versión 7.5) ,*
- la multitarea cooperativa, ahora ya sí, integrada en el sistema (evitando tener que instalar por separado la extensión *MultiFinder*),
- la introducción de los alias,
- mejoras en el sistema de organización de archivo: ahora las extensiones se trasladan cada una a su propia subcarpeta.

*Anteriormente sólo posible con extensiones de terceros.

En la versión ***Mac OS 7.6***. se abandona el nombre System, por estrategia de mercado (querían dar licencias para clones de Macintosh de otros fabricantes) y se adopta el de MAC.

Mac OS 8, fue lanzado en julio de 1997, poco después del regreso de Steve Jobs a la compañía. Era un momento difícil para Apple y decidieron sacar una versión nueva del SO y no una evolución de la versión 7, a pesar de no presentar cambios importantes.

Mac OS 9 fue lanzado el 23 de octubre de 1999. Incluía un soporte mejorado para la red inalámbrica *AirPort*, y una implementación y administración de memoria mejorada. Se introdujo una primera implementación de soporte multi-usuario y la máquina de búsqueda *Sherlock*. *AppleScript* se amplió para controlar redes y conexiones TCP/IP. Además se introdujo el uso de *Apple Software Update* (Actualización de Software Apple) para encontrar e instalar actualizaciones del SO y del hardware.

En Mac OS 9 también se introdujeron algunas tecnologías de transición para ayudar a los desarrolladores de aplicaciones a adaptar algunas características del futuro Mac OS X antes de introducir este nuevo SO al público, para facilitar la transición.

Mac OS X es la versión más reciente del sistema operativo de Apple. Tiene un código fuente, un sistema de archivos, un diseño y soporte hardware completamente distinto a las versiones anteriores. Mac OS X es un sistema operativo tipo UNIX. En 1999 fue presentada la versión para servidores, *Mac OS X Server 1.0*, y más tarde, en marzo de 2001, la versión para escritorio, *Mac OS X v10.0*.

Se han lanzado diez versiones del Mac OS X. a las que se les ha dado sobrenombres de felinos, hasta la versión OS X 10.9, cuando comenzaron a usarse sobrenombres de lugares de California.

- Mac OS X 10.0 (Cheetah)
- Mac OS X 10.1 (Puma)
- Mac OS X 10.2 (Jaguar)
- Mac OS X 10.3 (Panther)
- Mac OS X 10.4 (Tiger)
- Mac OS X 10.5 (Leopard)
- Mac OS X 10.6 (Snow Leopard)
- Mac OS X 10.7 (Lion)
- Mac OS X 10.8 (Mountain Lion)
- Mac OS X 10.9 (Mavericks)
- Mac OS X 10.10 (Yosemite)
- Mac OS X 10.11 (El Capitán)

2.3. Android.

Android es un SO basado en núcleo Linux. En sus inicios fue diseñado para dispositivos móviles con pantalla táctil y gradualmente se ha ido incorporando a otros dispositivos (televisores, automóviles, relojes inteligentes...).

El proyecto Android, nace en 2003 en Palo Alto, cuando Andy Rubin, Rich Miner, Chris White y Nick Sears fundaron Android Inc., empresa con la que querían desarrollar un sistema operativo para móviles, para ello contaban con el respaldo económico de Google.

En 2005 Google compró Android Inc. y el proyecto avanzó hasta que en noviembre de 2007 se creó la *Open Handset Alliance* (un grupo de fabricantes y desarrolladores de hardware, software y operadores de servicio) y se anunció la primera versión del sistema operativo: ***Android 1.0 Apple Pie***. El primer terminal con Android (el *HTC Dream*) no estuvo disponible hasta octubre de 2008.

2.3.1. Características de Android.

Las principales características de Android y que lo diferencia de otras plataformas móviles son:

- **Plataforma abierta**. Es una plataforma de desarrollo libre basada en Linux y de código abierto. Pudiendo usar y personalizar el sistema sin pagar derechos.
- **Adaptable a cualquier tipo de hardware**. Como ya se ha mencionado, está diseñado para su ejecución en otros dispositivos a parte de en teléfonos móviles o tabletas. Esto supone un esfuerzo mayor en el desarrollo de aplicaciones, ya que deberán funcionar correctamente en dispositivos con diferentes tipos de entrada, pantalla, memoria, etc. En comparación con iOS, donde hay que desarrollar la aplicación en versión para iPhone (teléfono móvil) y versión para iPad (tableta).
- **Portabilidad**. Las aplicaciones son desarrolladas en Java y por tanto ejecutadas en una máquina virtual de Java. Esto asegura la compatibilidad en cualquier tipo de CPU.
- **Arquitectura basada en componentes inspirados en Internet**. El diseño de la interfaz de usuario se hace en xml, permitiendo que una misma aplicación se ejecute en la pantalla reducida (móvil) o en otra de mayores dimensiones (TV).
- **Filosofía de "dispositivo siempre conectado a Internet"**.
- **Gran cantidad de servicios incorporados:** localización geográfica (basada tanto en GPS como en redes), bases de datos con SQL, reconocimiento y síntesis de voz, navegador, multimedia...
- **Buen nivel de seguridad**. Los programas se encuentran aislados unos de otros (concepto Linux de "ejecución dentro de una caja"), disponiendo de una serie de permisos que limitan su rango de actuación (localización, acceso a Internet, etc.)
- **Optimizado para baja potencia y poca memoria**. Android utiliza la Máquina Virtual Dalvik (una implementación de Google de la máquina virtual de Java optimizada para dispositivos móviles).
- **Alta calidad de gráficos y sonido**. Incorpora los códecs estándar más comunes de audio y vídeo (AVC, MP3, AAC, etc). Además reproduce gráficos vectoriales suavizados, animaciones inspiradas en Flash, gráficos en 3 dimensiones basados en OpenGL.

2.3.2. Arquitectura de Android

La arquitectura de Android está formada por cuatro capas, basadas en *software* libre.

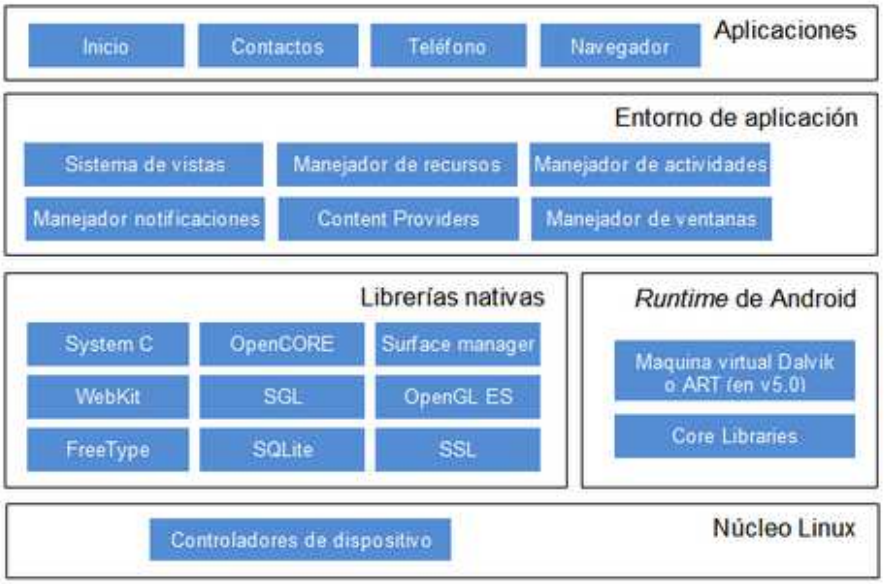


Figura 1. Arquitectura de Android

a. *El núcleo Linux*

Android utiliza el **núcleo de Linux 2.6** como una capa de abstracción entre el hardware y el resto de la pila. Es la única capa dependiente del *hardware* y contiene los drivers necesarios para el uso de sus componentes. Además proporciona los servicios de seguridad, manejo de la memoria, multiproceso y pila de protocolos.

b. *Runtime de Android y Bibliotecas nativas*

Al mismo nivel encontramos las bibliotecas nativas de Android y el entorno de ejecución.

- *Runtime de Android*

El entorno de ejecución está compuesto por las *Core Libraries* (las principales bibliotecas de clases Java), y la máquina *virtual Dalvik*.

Los dispositivos con Android tenían poca memoria y un procesador limitado, por lo que no podían usar una máquina virtual Java estándar, por eso Google creó la máquina virtual Dalvik, que fue reemplazada por *ART* a partir de Android 5.0.

- *Bibliotecas nativas*

Las bibliotecas nativas de Android proporcionan la mayor parte de sus utilidades. Muchas de éstas utilizan proyectos de código abierto. Destacan:

- **System C library**: con las cabeceras y funciones del estándar de lenguaje C. Todas las demás librerías están escritas en este lenguaje.
- **Media Framework**: biblioteca con los *codecs* necesarios para reproducir y grabar el contenido multimedia: MPEG4, H.264, MP3, AAC, AMR, JPG y PNG.

- **Surface Manager**: con los elementos de navegación de pantalla y de gestión de las ventanas de las aplicaciones activas. Además maneja el acceso al subsistema de representación gráfica en 2D y 3D.
- **WebKit**: base del navegador incluido por defecto en Android. Es la misma biblioteca utilizada en Google Chrome y Safari de Apple.
- **SGL**: es la biblioteca gráfica de Android que proporciona los gráficos en 2D, por tanto es la biblioteca más utilizada por las aplicaciones.
- **Librerías 3D**: la biblioteca OpenGL/SL maneja gráficos en 3D y permite utilizar, si el dispositivo dispone de él, el hardware que proporciona gráficos 3D.
- **SQLite**: motor ligero de bases de datos relacionales.
- **SSL**: proporciona servicios de encriptación *Secure Socket Layer* (capa de conexión segura).

c. Entorno de aplicación

Esta capa proporciona una plataforma de desarrollo de aplicaciones libre. Está diseñada para simplificar la reutilización de componentes. Las aplicaciones pueden publicar sus capacidades para que otras hagan uso de ellas (sujetas a las restricciones de seguridad).

Algunos de los servicios que incluye son:

- **Views (Manejador de vistas):** colección de vistas (parte visual de los componentes) para el diseño de las interfaces gráficas de usuario (GUI).
- **Activity Manager (Manejador de actividades):** Conjunto de API para el manejo del ciclo de vida de las aplicaciones, además proporciona un sistema de navegación entre ellas.
- **Notification Manager (Manejador de notificaciones):** para la gestión de las alertas lanzadas por las aplicaciones, estas notificaciones se muestran en la barra de estado sujetas a un mismo formato.
- **Content Providers:** para gestionar el intercambio de datos entre aplicaciones (como los contactos).

d. Aplicaciones

Este nivel está formado por todas las aplicaciones instaladas en una máquina Android, tanto las incluidas por defecto en la versión del SO como las instaladas por el usuario. Todas las aplicaciones se ejecutan sobre la máquina virtual Dalvik (o ART) y utilizan los servicios, librerías, etc. proporcionadas en las capas inferiores.

2.3.3. Versiones de Android y niveles de API

Las plataformas Android se identifican de tres formas alternativas: nombre comercial, versión y nivel de API. El nivel de API corresponde a números enteros comenzando desde 1. Para los nombres comerciales se han elegido nombres de dulces en orden alfabético: Apple Pie (v1.0), Banana Bread (v1.1), Cupcake (v1.5), Donut (v1.6), ...

Para programar una aplicación Android lo primero que se debe elegir es la versión mínima sobre la cual se ejecutará, ya que hay clases y métodos que están disponibles sólo a partir de una versión concreta. Todas la nuevas versiones son compatibles con las anteriores, solamente añaden funcionalidades nuevas; y en el caso de que se modifique alguna función, la antigua se clasifica como obsoleta pero se puede seguir usando.

La evolución de las versiones de Android es la siguiente:

Apple Pie _ ***Android 1.0 _ API 1*** (09/2008). Primera versión, incluía: Navegador, Google Maps, Google Mail, Calendar, reproductor de YouTube, la tienda de aplicaciones Android Market, entre otras herramientas.

Banana Bread _ ***Android 1.1 _ API 2*** (02/2009). Primera actualización, dedicada a la corrección de errores y no en la introducción de novedades.

Cupcake  _ ***Android 1.5 _ API 3*** (04/2009). Algunas novedades: teclado en pantalla con predicción de texto, grabación avanzada de audio y vídeo, widgets de escritorio, transiciones entre ventanas mediante animaciones...

Donut  _ ***Android 1.6 _ API 4*** (09/2009). Algunas novedades: búsqueda avanzada, síntesis de texto por voz, soporte para resolución de pantallas WVGA, nuevo atributo XML, mejora en la integración de la Galería, la Cámara y la Videocámara...

Éclair  _ ***Android 2.0 _ API 5*** (10/2009): sin apenas usuarios porque la mayoría de fabricantes pasaron directamente a la versión 2.1.

Android 2.1 _ API 7 (01/2010). Novedades: reconocimiento de voz, fondos de pantalla animados, información sobre la señal de red...

Froyo  _ ***Android 2.2 _ API 8*** (05/2010). Algunas novedades: modos de interfaz de usuario (“automóvil” y “noche”), opción de instalar apps en el almacenamiento externo (tarjeta SD), actualización automática de las apps, acceso a Internet a otros dispositivos (tethering) por USB o Wi-Fi....

Gingerbread  _ ***Android 2.3 _ API 9*** (12/2010). Algunas novedades: mayores tamaños de pantalla y resoluciones (WXGA y superiores), soporte para cámara delantera (pensado para videoconferencias), nuevo recolector de basura de la máquina virtual de Dalvik...

Honeycomb  _ ***Android 3.0 _ API 11*** (02/2011). La versión Honeycomb se destinó en exclusiva al uso en tabletas. Algunas novedades: optimización para pantallas grandes, aparecen los fragments, compatibilidad con procesadores multi-núcleo (multi-procesado con máquina Dalvik)...

Android 3.1 _ API 12 (05/2011). Algunas novedades: manejo de dispositivos conectados por USB, protocolo de transferencia de fotos y vídeo (PTP/MTP) y de tiempo real (RTP).

Android 3.2 _ API 13 (07/2011). Novedades: Optimizaciones para distintos tipos de tableta, sincronización multimedia desde SD...

Ice Cream Sandwich  ***_ Android 4.0 _ API 14*** (10/2011): Se unifican las dos versiones anteriores (2.x para teléfonos y 3.x para tabletas) en una sola compatible con cualquier tipo de dispositivo, nueva interfaz de usuario, nueva API de reconocimiento facial, gestor de tráfico de datos por Internet (permite definir el consumo límite para evitar cargos extras de la operadora)...

Android 4.0.3 _ API 15 (12/2011): Mejoras menores.

JellyBean  ***_ Android 4.1 _ API 16*** (07/2012): mejoras en la fluidez del interfaz de usuario y en las notificaciones...

Android 4.2 _ API 17 (11/2012) : posibilidad de tener varias cuentas de usuario en el mismo dispositivo (para tabletas), nuevo teclado predictivo deslizante, conexión con TVHD mediante wifi (Miracast), aplicación con funcionalidad PhotoSphere para hacer fotos panorámicas inmersivas (en 360º).

Android 4.3 _ API 18 (07/2013): perfiles restringidos para controlar los derechos de los usuarios (sólo en tabletas), mejoras en el asistente por voz de Google (arranque por voz del asistente con la orden "OK Google"), búsqueda de información sobre el número en una llamada entrante con la lista local de Google Maps si el número no está incluido en Contactos...

KitKat  ***_ Android 4.4 _ API 19*** (10/2013): para aumentar el número de dispositivos compatibles incluyendo a los de menor RAM (512 MB), se recortaron los componentes principales de Android y se creó una API para adaptar el comportamiento de las apps en los dispositivos con menor RAM. Nueva máquina virtual ART (en pruebas).

Lollipop  ***_ Android 5.0 _ API 21*** (11/2014): extensión a nuevas plataformas (Google Wear, Google TV y Google Card), sustitución de la máquina virtual Dalvik por la ART, compatibilidad con dispositivos de 64 bits en procesadores ARM, x86, y MIPS.

Android 5.1 _ API 22 (03/ 2015): soporte para múltiples tarjetas SIM, mejoras en la seguridad, velocidad y estabilidad...

Marshmallow  ***_ Android 6.0 _ API 23*** (10/ 2015): Soporte para lectura de huellas dactilares, compatibilidad con lápices Bluetooth, USB 3.1., compatibilidad con pantallas 4k...

Nougat _ ***Android 7.0*** _ ***API 24*** (06/ 2016). Última versión de Android. Incluye la función "Doze" para la optimización de la batería (detección de que el móvil está guardado en un bolsillo o un bolso y puesta automática de las aplicaciones en modo ahorro), modo multitarea para abrir dos aplicaciones simultáneamente, respuesta directa a las notificaciones...

2.4. iOS.

Es el sistema operativo móvil desarrollado por Apple Inc. En un principio se desarrolló únicamente para su uso en el iPhone, pero más tarde fue adaptado para usarse en el resto de sus dispositivos móviles (iPad y iPod touch). Actualmente más de la mitad de los usuarios de dispositivos móviles de Apple tienen instalada la versión **iOS 8**.

2.4.1. Características de iOS.

La principal característica del sistema operativo iOS y que lo diferencia de Android, es que no está permitida su instalación en hardware de terceros. Para poder usar este sistema operativo es necesario disponer de un dispositivo de Apple, los únicos autorizados para usarlo.

iOS incluye toda la tecnología necesaria para implementar las aplicaciones nativas, y las aplicaciones básicas para aprovechar el dispositivo móvil: teléfono, Mail, Safari… Su funcionamiento está basado en el concepto de “manipulación directa”: el usuario interactúa con la pantalla del dispositivo por medio de gestos táctiles como toques, pellizcos y deslizamientos.

Pantalla principal (SpringBoard)

En ella están los iconos de las aplicaciones instaladas. Además en su parte inferior está el “Dock” dónde se pueden anclar las aplicaciones más usadas. En la parte superior está la “Barra de estado” con información como: la hora, el nivel de batería y la intensidad de la señal.

Carpetas

Desde la versión iOS4 existe un sistema simple de carpetas. Moviendo una aplicación sobre otra se crea una carpeta a la que se pueden añadir más aplicaciones con el mismo proceso. El nombre de la carpeta se asigna automáticamente según el tipo de aplicaciones que contenga, aunque también puede ser personalizado por el usuario.

Seguridad

Debido a la gran cantidad de robos de iPhones, el gobierno de EEUUpidió a Apple que diseñara un sistema de seguridad para sus equipos. Este sistema se incluyó en la versión iOS7: el sistemaiCloudpide los datos de acceso de la cuenta del usuario permitiendo bloquear e inutilizar un equipo perdido o robado. Además da información de la ubicación del mismo vía GPS.

Centro de notificaciones

Desde la versión iOS5, las notificaciones son mostradas en un área accesible haciendo un deslizamiento hacia abajo desde la barra de estado. Dando un toque sobre una notificación el SO abre la aplicación afectada.

Multitarea Opcional

La multitarea estaba reservada para las aplicaciones del sistema, ya que Apple quería evitar problemas de batería y rendimiento por la ejecución de varias aplicaciones de terceros al mismo tiempo. A partir de iOS 4, los dispositivos de tercera generación y posteriores permiten el uso de multitarea también para:

1. Audio en segundo plano
2. Voz IP
3. Localización en segundo plano
4. Notificaciones push
5. Notificaciones locales
6. Completado de tareas
7. Cambio rápido de aplicaciones

Tecnologías no admitidas

iOS no permitía el uso de la Plataforma Java y Adobe Flash, debido a las críticas recibidas, desde iOS 8, ya se permite el uso de Adobe Flash.

2.4.2. Arquitectura de iOS.

La Arquitectura iOS presenta una arquitectura en 4 capas, al igual que Android. Las capas inferiores contienen servicios y tecnologías fundamentales. Las capas de nivel superior se basan en las capas inferiores y proporcionar servicios y tecnologías más sofisticadas.

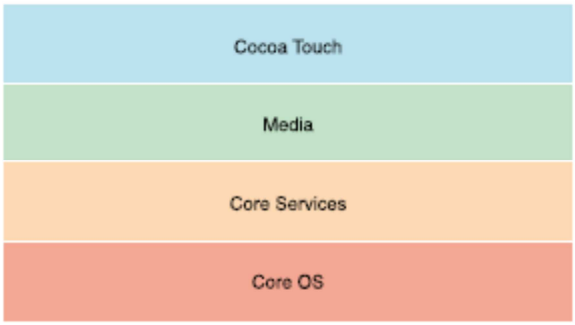


Figura 2. Arquitectura de iOS

CocoaTouchLayer

La capa *CocoaTouch* contiene los *frameworks*basepara la construcción de aplicaciones de iOS que definen el aspecto de éstas. También proporciona la infraestructura básica y apoyo para implementar otras tecnologías como la multitarea, la entrada táctil, las notificaciones, y otros servicios de alto nivel del sistema. Al diseñar aplicaciones para iOS, hay que comprobar que las tecnologías de esta capa cumplen con nuestras necesidades.

Media Layer

La capa “Media” contiene gráficos, audio y tecnologías de vídeo usados para implementar reproducción multimedia en lasapps.

Core ServicesLayer

La capa *Core Services* contiene los servicios que el sistema provee a las aplicaciones. Los servicios más importantes son: el *Core Foundation* y los *frameworksFoundation*. Esta capa además contiene las tecnologías necesarias para implementar características como la ubicación, iCloud, medios de comunicación social y la creación de redes.

Core OS Layer

La capa Core OS es la de más bajo nivel, sobre la que se construyen el resto de tecnologías. El desarrollador no programa directamente sobre ella, sino que los *frameworks* hacen uso de ella.

2.5. Cuadro comparativo de los SO.

	Windows	Macintosh	Android	iOs
Empresa de desarrollo	Microsoft Corporation	Apple Inc.	Google	Apple Inc.
Fecha primera versión	1985	1984	2008	2007
Versión actual	Windows 10	Mac X	Android 7.0 Nougat, API 24	iOS 8
Dispositivos compatibles	PCs y portátiles. Smartphones Tabletas	PCs y Portátiles exclusivamente de la marca Apple	Multi-dispositivo: móviles, tabletas, relojes, frigoríficos,…	Móviles y Tabletas exclusivamente de la marca Apple
Lenguaje de desarrollo aplicaciones	.Net C# C++ Visual Basic	Objective C	Java	Objective C
Entorno de desarrollo estándar	Visual Studio Community	Herramientas del Mac Dev Center	Android Studio	Herramientas del Mac Dev Center

2.6. El problema de la coexistencia de distintas plataformas.

De la introducción a los SO operativos anterior, no es difícil deducir, que la cantidad de horas de estudio y experiencia que requiere un desarrollador software para poder conocer todas las características y herramientas de un SO es muy elevada. Además este conocimiento tiene que ser revisado en cada uno de los lanzamientos de actualizaciones importantes y/o de versiones nuevas.

Por tanto, cuando lo que se desea es el desarrollo de una aplicación que será ejecutada en diferentes plataformas. La tarea que de por sí ya puede ser compleja en una única plataforma, necesitará de varios grupos de desarrolladores especialistas cada uno en una plataforma diferente, lo que incrementa los costes de producción de la aplicación.

Para resolver este inconveniente, los fabricantes de herramientas de desarrollo software, han estado desarrollando herramientas que permitan la programación de aplicaciones sobre diferentes plataformas sin la necesidad de que el desarrollador software conozca en profundidad todas estas plataformas, sus herramientas y lenguajes de desarrollo.

3. ¿Qué es RAD Estudio y cómo trabajamos con él?

3.1. Embarcadero.

Embarcadero desarrolla herramientas para mejorar la eficiencia en los trabajos sobre las principales plataformas de bases de datos, sistemas operativos, *frameworks* y lenguajes de programación. Con las herramientas multiplataforma, Embarcadero busca facilitar a los desarrolladores de software el diseño, la construcción y la administración de bases de datos y aplicaciones en los entornos que deseen, reduciendo las limitaciones, los costes y las curvas de aprendizaje que conlleva el uso de múltiples herramientas específicas para cada plataforma.

3.2. Características de RAD Estudio.

RAD Studio es un **IDE (Entorno de desarrollo integrado)** multiplataforma fabricado por Embarcadero, que ofrece la posibilidad de desarrollar aplicaciones para plataformas móviles (iOS y Android) o de ordenadores personales (Windows y Mac).

RAD Studio puede contener dos productos, Delphi y C++ Builder, que permiten desarrollar las aplicaciones con dos lenguajes de programación diferentes:

- **Delphi** utiliza Object Pascal como lenguaje de programación.
- **C++ Builder** utiliza C++ como lenguaje de programación.

Con el IDE, se pueden desarrollar aplicaciones en ambos lenguajes de programación (Delphi y C++), dependiendo de los productos comprados. Estos lenguajes de programación son también conocidos como ***"personalidades"*** en RAD Studio y su documentación.

Las herramientas disponibles en el IDE dependen de la versión en la que nos encontremos. Las más comunes están descritas en el ANEXO A. Introducción a RAD Studio.

3.2.1. Frameworks y Bibliotecas admitidas.

RAD Studio contiene las siguientes 3 bibliotecas/frameworks:

- **FMX :** Disponible desde la versión XE2. *FireMonkey* es un framework de desarrollo de Interfaces de usuario para aplicaciones multi-plataforma que permite desarrollar dos categorías de aplicaciones e interfaces: HD (interfaces clásicas en 2D) y 3D (que permiten la visualización de escenas en 3D). Ambas categorías pueden ser combinadas. FMX también incorpora efectos visuales (desenfoques, brillos...) y animaciones. Además, aunque no suele usarse con los "temas" nativos de las distintas plataformas, sí es compatible a través de bibliotecas de terceros.
- **VCL** (Visual Component Library) es la biblioteca de componentes Visual, conjunto de componentes visuales para el desarrollo de aplicaciones de Windows en lenguaje Delphi. Contiene clases visuales, no visuales, y de servicios para la construcción de aplicaciones de Windows, aplicaciones web, aplicaciones de bases de datos y aplicaciones de consola.

- **RTL** (run-time library). En programación de computadoras, una run-time library (biblioteca en tiempo de ejecución), es un conjunto de rutinas de bajo nivel que usa el compilador para hacer funcionar el entorno de ejecución mediante llamadas al ejecutable binario compilado. RTL se compone de unidades básicas que proporcionan el soporte para la mayoría de componentes de VCL y FMX.

  La siguiente tabla muestra la compatibilidad entre las tres principales bibliotecas/frameworks disponibles en Rad Studio y las plataformas de desarrollo.

Library\Platform	Win32	Win64 (Delphi and C++)	Mac OS X	iOS (Simulator and Device)	Android
FMX	Yes	Yes	Yes	Yes	Yes
RTL	Yes	Yes	Yes	Yes	Yes
VCL	Yes	Yes	Not supported	Not supported	Not supported

Figura 3. Compatibilidad entre Bibliotecas/ Framewokrs y SSOO.

Actualmente está en activo la primera revisión de la décima versión, llamada ***RAD Studio 10.1 Berlín***. Algunas de las características de ésta y anteriores versiones son:

- Rad Studio 2009 y 2010:
  - Soporte para desarrollo en Windows 7.
  - Desarrollo de aplicaciones compatibles con pantalla táctil.
  - En lenguaje Delphi, sistema de RTTI para acceder a la información de cualquier objeto, independientemente de su clase.

- Rad Studio XE
  - Ampliación del desarrollo multi-capa con DataSnap.
  - Cloud computing soportado por Azure y Amazon.
  - Integración de Subversion en el IDE.

- Rad Studio XE2
  - Compilador Delphi 64-bit para Windows
  - Plataforma de aplicaciones FireMonkey, creación de aplicaciones multi-plataforma y/o multi-dispositivo desde un único código fuente.
  - Creación de aplicaciones de escritorio de Windows y OS X con FireMonkey.
  - Conectores móviles DataSnap.

- Rad Studio XE3
  - Compilador C++ 64-bit para Windows

  - Soporte para Windows 8
  - Incorporación de los diseños modernos de Windows a las aplicaciones con Metropolis UI.
  - Soporte para OS X Mountain Lion y Retina Display
  - Mejoras en estilos FireMonkey, audio/video y 3D
  - HTML5 Builder para crear aplicaciones para webs y webs móviles.

- Rad Studio XE4
  - Desarrollo de aplicaciones iOS con el compilador Delphi iOS ARM.
  - Base de datos IB Lite para iOS.
  - FireDAC, para acceso a datos.
  - Sensores FireMonkey para la localización, orientación y movimiento.
  - Modo pantalla completa OS X.

- Rad Studio XE5
  - Desarrollo de aplicaciones Android con Delphi y de aplicaciones iOS con C++ .
  - Nuevos componentes móviles para Android e iOS.
  - Soporte REST Client para soporte al acceso a la nube basado en REST ful web services.
  - Base de datos IB Lite para Android.
  - Desarrollo de aplicaciones iOS 7.

- Rad Studio XE6
  - Componentes Tethering, para el uso de un dispositivo móvil con conexión a Internet como pasarela para ofrecer acceso a la red a otros dispositivos
  - Componente Taskbar (barra de tareas)
  - Servicio en la nube (Cloud Service - BaaS) integrado con componentes Kinvey y Parse
  - Aplicaciones de compras y publicidad.
  - Creación de aplicaciones para Google Glass.
  - Construcción de aplicaciones Android con C++.
  - Paquete de soporte C++ para Windows 64-bit
  - Mejoras FireDAC y actualización del driver Informix.

- Rad Studio XE7
  - Mejoras en FireMonkey, como la ubicación de elementos de IU: los controles se ajustan automáticamente según la plataforma destino de la aplicación (como las pestañas).
  - El componente MultiView, muestra dinámicamente cómo se verá la aplicación en diferentes plataformas y dispositivos.
  - Acceso a bases de datos profesionales de Oracle, DB2, Microsoft SQL Server Informix, SQL Server y otras, desde una aplicación móvil a

través de *middleware* (lógica de intercambio de información entre aplicaciones).
    - Servicio de almacenamiento seguro de datos tanto en servidores como dispositivos móviles.
    - Conectividad de las aplicaciones con App Tethering, REST y Bluetooth
    - Nuevo diseñador multi-dispositivo integrado.
    - Soporte IDE para agregar clases de Java a una apk Android
    - Tour guiado por el IDE para aprender las características de RAD Studio.

- Rad Studio XE8
    - Disponibilidad individual de Delphi 7 y C++ Builder. Podemos instalarnos sólo el módulo que deseemos. (Por ejemplo, para la realización de este proyecto, sólo se ha requerido de la instalación de C++ Builder).
    - Base de datos integrada. InterBase XE7 Developer Edition, con hasta 20 usuarios y 80 conexiones lógicas.
    - Biblioteca FIREDAC de acceso a datos multi-dispositivo.
        - Soporte para bases de datos para móviles iOs y Android, incluyendo SQLite, InterBase ToGo y IB Lite.
        - Herramientas y scripts de ayuda a la migración de código dbExpress a FireDAC.
    - Unidad de Testing integrada., con el framework de testing DUnitx.
    - Integración del controlador de versiones GIT, incluyendo autentificación y cambios Push and Pull de y hacia repositorios remotos.
    - Controlador de versiones Mercurial, con soporte para la clonación del repositorio remoto y confirmación de cambios a nivel local.
    - Mejoras en el Project Manager, para añadir archivos de base de datos, simplificando su despliegue.
    - Nueva unidad de System.Hash RTL, con funciones hash para soportar el nuevo framework HTTP.
    - Nuevos componentes NetHTTPClient y NetHTTPRequest, para un acceso fácil al framework cliente HTTP.

- Rad Studio 10 Seattle
    - Mejoras en VCL para construir aplicaciones de 32 y 64 bits para **Windows 10**, 8.x Windows y Windows 7; Windows Server 2008 y 2012.
    - Incluye la SDK de Microsoft Windows para desarrollo de aplicaciones para las APIs Windows 7, Windows 8 y Windows 10, tanto para las APIs clásicas Win32 / Win64 como para la API WinRT.
    - Plataforma FMX para crear aplicaciones nativas Android ARMv7 para Ice Cream Sandwich (4.0.3, 4.0.4), JellyBean (4.1, 4.2, 4.3), Kit Kat (4.4) y Lollipop (5.x).
    - Plataforma FMX para crear aplicaciones nativas iOS para iOS 7.x e iOS 8.x, para ambos ARMv7 y ARM 64-bits.

  - Sugerencias posicionando el ratón sobre los controles visuales de FireMonkey.
  - Mejoras de FireMonkey para dar soporte al desarrollo en Windows 10.
  - Mejoras en el framework Bluetooth de FMX y un nuevo componente para conexiones Bluetooth clásicas.
  - Ampliación del framework VLC: Activity Indicator, SearchBox, RelativePanel, ToggleSwitch, SplitView,

- Rad Studio 10.1 Berlin
  - *Diseñador Multi-dispositivo FireUI*, para la creación de aplicaciones que funcionen con múltiples formatos de teléfonos móviles, tablets y equipos de escritorio a partir de un código fuente en común y compartido.
  - *FireUI App Previews*, para la visualización en tiempo real de las interfaces de cada plataforma.
  - Revisión del compilador C++ para optimizarlo en el desarrollo sobre Windows 10 y móviles.
  - Soporte extendido para múltiples monitores
  - Inspector de objetos mejorado

Tras este breve repaso a la evolución del IDE y sus herramientas, quisiera destacar, que como bien se indica, algunas de éstas herramientas no son compatibles con todas las plataformas de desarrollo, ni con las dos personalidades que Rad Studio ofrece (Delphi y C++). Esto será un factor a tener en cuenta a la hora de pensar qué aplicación se va a desarrollar, ya que las herramientas empleadas deberán ser compatibles en todas las plataformas en las que se ejecutará la aplicación y en la personalidad Rad que empleemos (en este proyecto se ha usado C++ Builder).

3.3. Desarrollo de aplicaciones multi-dispositivo en RAD Studio

Podemos usar RAD Studio para desarrollar (en un entorno Win32) aplicaciones multiplataforma que se ejecutarán en las siguientes plataformas.

- 32-bit Windows
- 64-bit Windows
- Mac OS X
- 32-bit iOS Device
- 64-bit iOS Device
- iOS Simulator
- Android

Actualmente, iOS Simulator sólo es compatible con Delphi. El resto de plataformas admiten tanto Delphi como C++ Builder.

Ejecutar y depurar una aplicación multi-dispositivo para Mac OS X, iOS o Windows 64-bits (si estamos desarrollando en un entorno Win32) requiere que nuestro entorno de desarrollo esté conectado a la plataforma de destino (o a una plataforma

intermedia que lo soporte), donde el *Platform Assistant server* (el servidor remoto de aplicaciones) se ejecuta en modo de escucha.

Teniendo en cuenta que RAD Studio es un IDE desarrollado en Windows 32-bits, podemos encontrarnos con 3 casos posibles:

- **Plataforma destino Windows 64-bits.**
  - En el caso de estar desarrollando sobre un SO Windows de 64-bits, no será necesaria una configuración especial.
  - Si desarrollamos sobre un entorno diferente habrá que seguir los siguientes pasos:
    1. Conectar tu entorno a un PC de 64 bit, Embarcadero nos proporciona un tutorial sobre cómo hacerlo en el link.
    2. Instalar y ejecutar ***PAServer***, servidor multiplataforma, en un PC de 64 bits.
    3. Crear un perfil de conexión que describa la conexión a la PAServer en el PC de 64 bits.

- **Plataforma destino Mac OS X o iOS.**

  1. Conectar el entorno de desarrollo con un dispositivo Mac.
  2. Instalar y ejecutar *PAServer*, en el dispositivo Mac.
  3. Crear un perfil de conexión que describa la conexión a la PAServer en el MAC.
  4. Añadir un SDK para construir la aplicación.
  5. Adquirir los ID de desarrollador y certificados necesarios de Apple.
     - Mac: Mac OS X ApplicationDevelopment
     - iOS: Joining the Apple Developer Program,
  6. Configurar en RAD Studio los certificados y perfiles Apple adquiridos.

- **Plataforma destino Android.**
  1. Instalar las herramientas de desarrollo Android (SDK y NDK), si no se ha elegido esa opción durante la instalación de RAD Studio.
  2. Añadir un SDK Android para construir la aplicación (sólo si se han instalado las herramientas de desarrollo Android manualmente o deseas instalar alguna en concreto).
  3. Configurar el dispositivo de pruebas:
     - Si se usa un dispositivo Android
       a. Habilitar la depuración por USB en el dispositivo Android.
       b. Configurar el sistema de desarrollo para que detecte el dispositivo Android (si no se usa un simulador).

     - Si se usa un simulador Android
       a) Si se usa un simulador de Android, RAD Studio lo detecta automáticamente cuando el dispositivo virtual Android es arrancado.

Para la realización de este proyecto se dispuso de un dispositivo con Windows 64-bits y otro con Android (ningún dispositivo de Apple), por lo que sólo fue necesaria realizar la configuración de la plataforma Android.

Para la configuración de la plataforma Android, se procedió a la descarga de la API 17 de Android, ya que la instalada por defecto con RAD Studio es la API 22, y no es conveniente desarrollar las aplicaciones Android en un nivel de API demasiado nuevo, para evitar incompatibilidades con los dispositivos más antiguos.

Además las pruebas y depuración de la aplicación para Android, se realizó tanto con un simulador de Android (Genymotion) como con un dispositivo real Nexus 4 LG con Android versión 5.1.1.

3.4. Genymotion. (link)

Genymotion es un emulador de Android disponible para Linux, Windows y OS X, con el que se pueden descargar y ejecutar imágenes de Android de diferentes versiones y dispositivos, del que se distribuyen versiones gratuitas.

Generar una máquina virtual Android con Genymotion es muy sencillo. Solo hay que seleccionar un *smartphone* o tablet de la lista, revisar la información del dispositivo y darle un nombre a la máquina virtual, Genymotion realizará la descarga de la imagen.

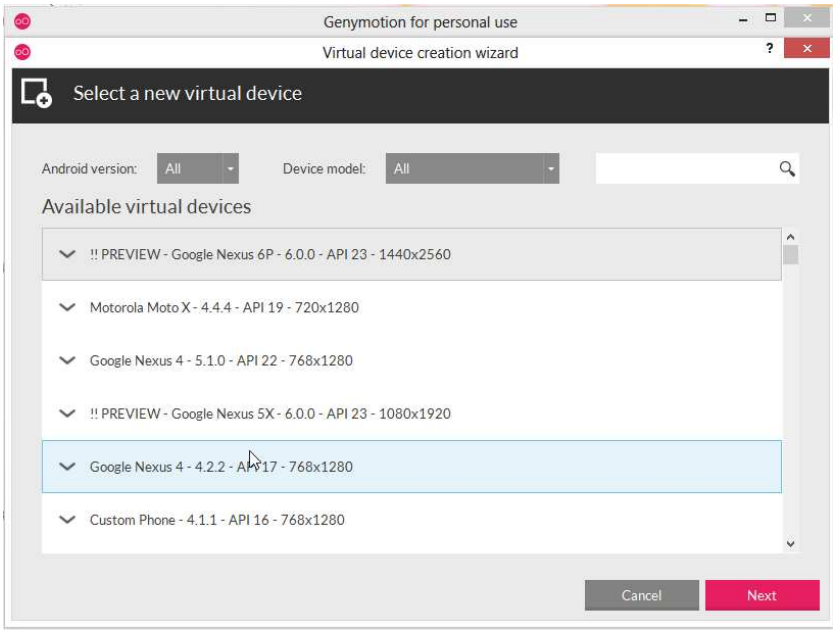


Figura 4. Pantalla inicio Genymotion

Para este proyecto creé dos imágenes, correspondientes a un Google Nexus 4 y un Motorola Moto X, con las API 17 y 19, respectivamente, suficientes para ejecutar la aplicación desarrollada en API 17.

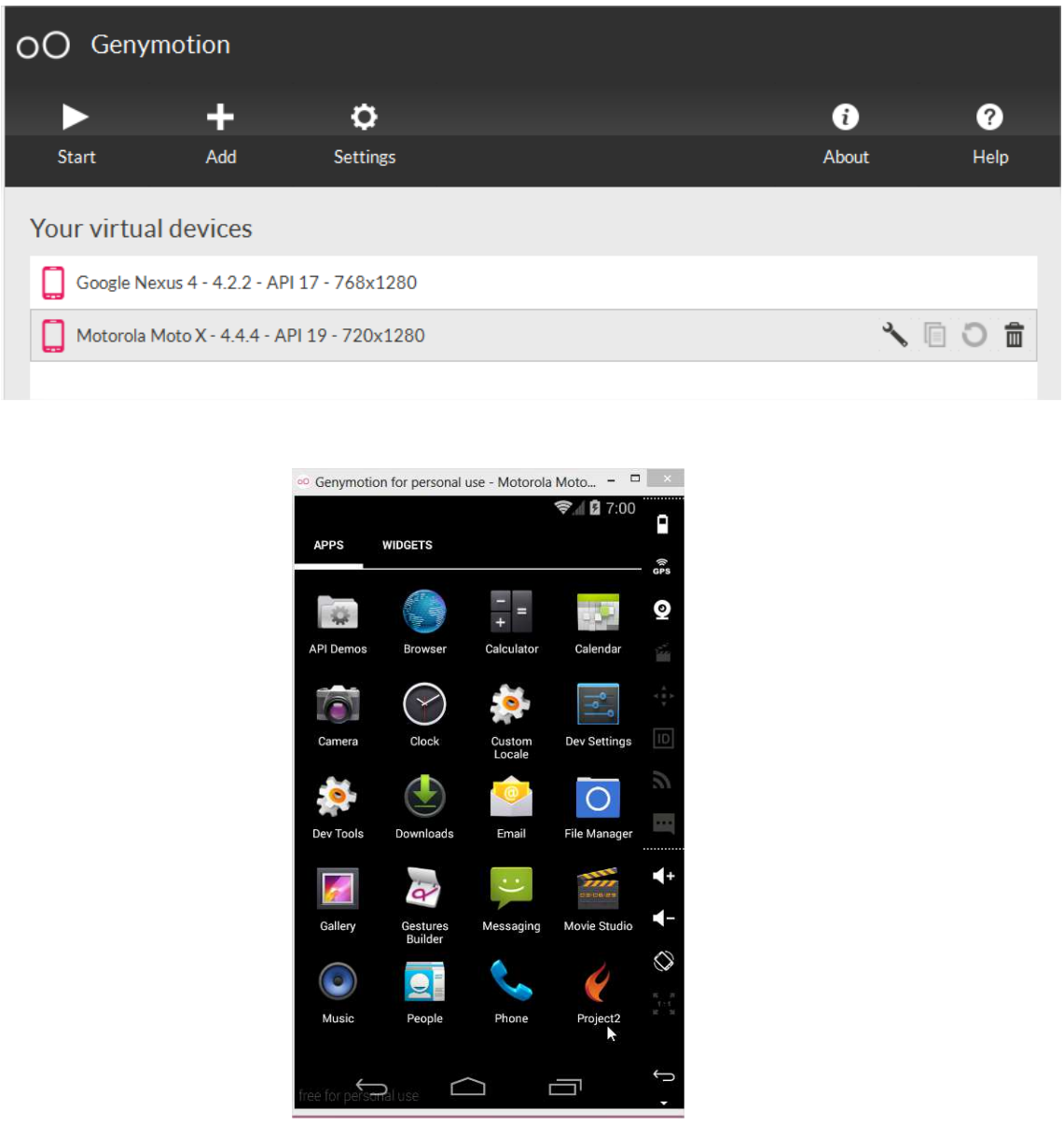


Figura 5. Dispositivo virtual Genymotion con aplicación Project2 (desarrollada con RadStudio) instalada.

4. La aplicación creada con RAD Estudio.

Antes de empezar a programar la primera aplicación que me viniese a la mente, debía pensar bien las limitaciones técnicas del IDE (que en uno de los SO operativos ya sea posible realizar alguna función nueva, no quiere decir que el IDE ya se haya actualizado para poder implementar esas nueva función), la compatibilidad entre plataformas (no todos los dispositivos disponen del mismo tipo de sensores, ni compatibilidad con controles especiales, etc.) y mis conocimientos en lenguaje C++.

Dentro del tiempo destinado a la realización de este proyecto, tuve que dedicar bastantes horas al aprendizaje del lenguaje C++, ya que mi experiencia con la programación hasta hoy había sido con otros lenguajes (Java, Visual Basic, JavaScript). Al ser mi primera aplicación en lenguaje C++ decidí que no intentaría usar clases C++ muy avanzadas y que en caso de usar alguna clase avanzada lo haría mediante alguna de las herramientas de Rad Studio.

Para conseguir la mayor compatibilidad entre plataformas opté por hacer una aplicación compuesta por controles tipo Button, Label y EditText, ya que son componentes básicos existentes en todas los SO.

La aplicación que realicé es una copia de la calculadora de Windows 8. Además, para aumentar su complejidad y funcionalidad, decidí ampliarla con un conversor de divisas.

Para realizar la copia de la calculadora, no sólo me fijé en su aspecto, sino que se estudié como es el funcionamiento de cada una de sus teclas, para redactar las siguientes especificaciones funcionales.

Figura 6. Calculadora de Windows.

4.1 . Especificaciones funcionales:

MC **MC (Memory Clear):** Elimina cualquier número almacenado en memoria.

MR **MR (Memory Recall):** Recupera el número almacenado en memoria. El número permanece en memoria

MS **MS (Memory Storage):** Almacena en memoria el número mostrado en ese momento por pantalla.

M+ **M+:** Suma el número mostrado a otro número que se encuentre en memoria pero no muestra la suma de estos números.

M- **M-:** Resta el número mostrado a otro número que se encuentre en memoria pero no muestra la resta de estos números.

CE **CE (Clear error):** Elimina el número mostrado. Se utiliza para cuando se comete un error en el ingreso de datos pero sin eliminar todo el cálculo que se está realizando.

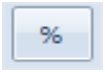
C **C (Clear):** Elimina todo el cálculo actual.

← **Delete**: Si se está introduciendo un operando elimina el último carácter introducido. Si se está mostrando un resultado, no hace nada.

Teclado numérico: Concatena el valor pulsado a la entrada de datos. Si ya hay una coma escrita y se pulsa otra vez el botón "," no se volverá a concatenar. Si la pantalla está mostrando un resultado, no se concatenan los valores, se borra el resultado y se muestra el número pulsado o "0," si se ha pulsado ",".

/ * - + **Operaciones con dos operandos:** Si hay otra operación en curso, realizan la operación mostrando el resultado por pantalla y se guarda la nueva operación que se enlazará.

Operaciones con un operando que no enlazan con otras operaciones: realiza la operación correspondiente (inversa o raíz cuadrada) sobre el número mostrado en pantalla, o si hay una operación en curso sobre el resultado de ésta, y muestra el resultado por pantalla.



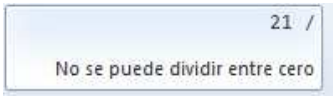
Operaciones con un operando que enlazan con otras operaciones - Porcentaje: Realiza el X% del operando Y introducido previamente, para usar ese porcentaje como segundo operando en una operación enlazada. Pulsando "=" u otra operación, se muestra el resultado de esa operación enlazada. Por ejemplo, la secuencia : "1000" -> "/" -> "10" -> "%" -> "=", realiza la operación 1000/(el 10% de 1000) = 1000/100 = 10. En el caso de pulsar otro número antes de pulsar =, se pierde el cálculo del % y el numero pulsado será el que enlace con la operación pendiente. Por ejemplo, la secuencia : "1000" -> "/" -> "10" -> "%" -> "50"->"=", realiza la operación 1000/50 = 20.

Igual: realiza la última operación tomando como segundo operador, el que esté mostrado en pantalla, y muestra el resultado, limpiando el historial de operaciones. Si se pulsa varias veces seguidas sin haber intercalado otra operación (+,-,x,/) entre medias, repite la última operación realizada la última vez que se pulsó =.

Pantalla de entrada/salida de datos: en la pantalla se muestra o el operando que se está introduciendo o el último resultado obtenido. Además si hay algún número guardado en memoria, muestra una M de aviso.

Excepciones:

Cuando se produce un excepción se bloquean todos los botones excepto C y CE y aparecen los mensajes correspondientes a la excepción.



División por cero: al intentar dividir por 0 (con la operación inversa o división), aparece un mensaje de error.

División 0/0: aparece un mensaje de resultado indefinido. Se bloquean todos los botones excepto C y CE.

Inversa de 0: aparece un mensaje de resultado indefinido. Se bloquean todos los botones excepto C y CE.

Raíz cuadrada de un negativo: mensaje de entrada no válida. Se bloquean todos los botones excepto C y CE.

Funcionalidad añadida: Conversor de divisas.

De € a otra divisa: toma el valor en euros del número representado en la pantalla y devuelve el cambio a la divisa correspondiente. El valor devuelto debe poder enlazarse con otras operaciones de la calculadora.

De otra divisa a €: toma el valor en la divisa correspondiente del número representado en la pantalla y devuelve el cambio a euros. El valor devuelto debe poder enlazarse con otras operaciones de la calculadora.

4.2 . Estructura de la aplicación.

Una vez redactadas las especificaciones funcionales de la aplicación, lo siguiente es pensar de qué manera se va a implementar la aplicación. Como Embarcadero dice que con Rad Studio es posible reutilizar el código, pensé que la mejor manera de aprovechar esta característica del IDE, es utilizar algún patrón de diseño que también favorezca la reutilización de código.

Actualmente en el diseño de Software se suelen usar las arquitecturas multinivel o por capas. En estas arquitecturas cada nivel tiene una misión simple, esto permite que las aplicaciones sean escalables (pueden ampliarse o adaptarse en caso de que las necesidades cambien).

Para seguir al máximo posible las buenas prácticas en desarrollo software, elegí la arquitectura en 3 capas , la cual sigue el siguiente esquema:

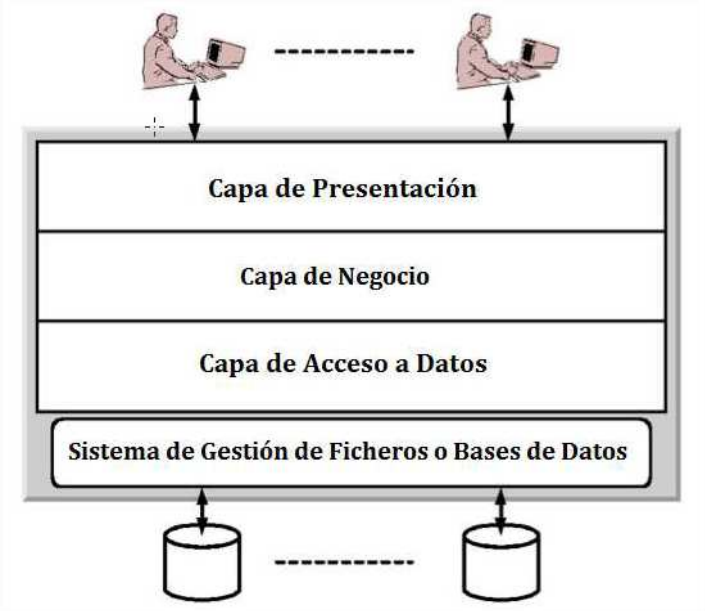


Figura 7. Arquitectura en 3 capas.

La capa de presentación (Vista): es la que ve el usuario (también llamada "capa de usuario"). Presenta el sistema al usuario, comunicando y capturando la información que éste maneja. También es conocida como "interfaz gráfica" puesto que suele basarse en imágenes entendibles para el usuario (pocas aplicaciones siguen teniendo una consola como interfaz de usuario) .

Las funciones de la capa de presentación son, por tanto:

- Recibir las peticiones del usuario.
- Ordenar la ejecución de acciones.
- Comunicar los resultados al usuario.
- Tratar las interfaces gráficas, tales como botones, diálogos, menús o listados.

La capa de negocio: es la que aloja los programas que se ejecutan. Recibe las peticiones del usuario, las trata y devuelve los resultados a la Vista. Se llama capa o lógica de Negocio porque es aquí donde se establecen todas las reglas que deben cumplirse. Básicamente realiza todas las operaciones tomando y devolviendo datos, pero ignorando cómo se guardan los datos y cómo son presentados al usuario. Por tanto, se ocupa de:

- Conocer los eventos.
- Controlar la validez de los datos.
- Cambiar el estado del dominio.
- Ejecutar las acciones encomendadas.
- Conocer las consultas.
- Comunicar las respuestas.

La capa de datos: es donde residen los datos y es la capa encargada de acceder a ellos para consultarlos, modificarlos o guardarlos. Está formada por las clases que gestionan la comunicación con el sistema encargado de la persistencia de datos (una base de datos, un fichero binario o XML, ...). Se ocupa de:

- Permitir a la capa de negocio ignorar donde se encuentran los datos con los que trabaja.
- Permitir que determinados objetos del dominio sean persistentes.

La ventaja principal que vi en desarrollar la aplicación en esta arquitectura, es que al programar una aplicación multi-dispositivo, si deseamos optimizar la vista mostrada según las diferentes pantallas que podemos encontrar en cada plataforma, sólo deberemos modificar la parte correspondiente a la vista sin tener que tocar el resto. Del mismo modo, si cada plataforma, para la que estamos desarrollando, es compatible con diferentes tipos de almacenamiento de datos, la aplicación sólo necesitará ser modificada en la parte correspondiente a la capa de Datos.

Pensé que la mejor manera de que la calculadora tuviese el valor de las divisas actualizado en tiempo real (o lo más real posible), era usar un Servicio Web que me proporcione esos datos, en vez de instalar una base de datos que además de ser más pesado necesita que actualice los datos cada día. Además, hoy día, tanto los dispositivos Android como Windows los usamos conectados a Internet todo el tiempo, así que la necesidad de conexión a Internet para el uso del conversor no supone un gran inconveniente. Aunque sí presenta el inconveniente de que el Servicio Web, al ser gratuito, deje de dar servicio en algún momento.

Busqué un Servicio Web que proporciona el valor del cambio de divisas a través del WSDL : *http://www.webservicex.com/CurrencyConvertor.asmx?wsdl.* y aproveché la herramienta de Rad Studio *WSDL Wizard* que escribe automáticamente una clase avanzada con los métodos necesarios para la solicitud de los datos al Servidor Web.

Al hacer pruebas de esta clase sobre las diferentes plataformas, observé que en el caso de Android, la clase implementada por *WSDL Wizard* no funciona correctamente, por lo que tuve que modificar la capa de datos de la aplicación Android. Haber diseñado la aplicación con el modelo de 3 capas, me facilitó el trabajo.

Así pues, en las dos versiones realizadas (Android y Windows) los valores del cambio de divisas se encuentran almacenados de dos modos diferente:

1. Android: los valores están contenidos en unas variables pertenecientes a la Clase Divisa (no se realiza ningún acceso a datos).
2. Windows: los valores del cambio de divisas son proporcionados por el método ConversionRate de la clase CurrencyConvertor, la cual es la clase avanzada escrita automáticamente por C++ Builder con su herramienta *WSDL Wizard*.

Así, la estructura de la *aplicación Calculadora Android* queda representada en el siguiente diagrama:

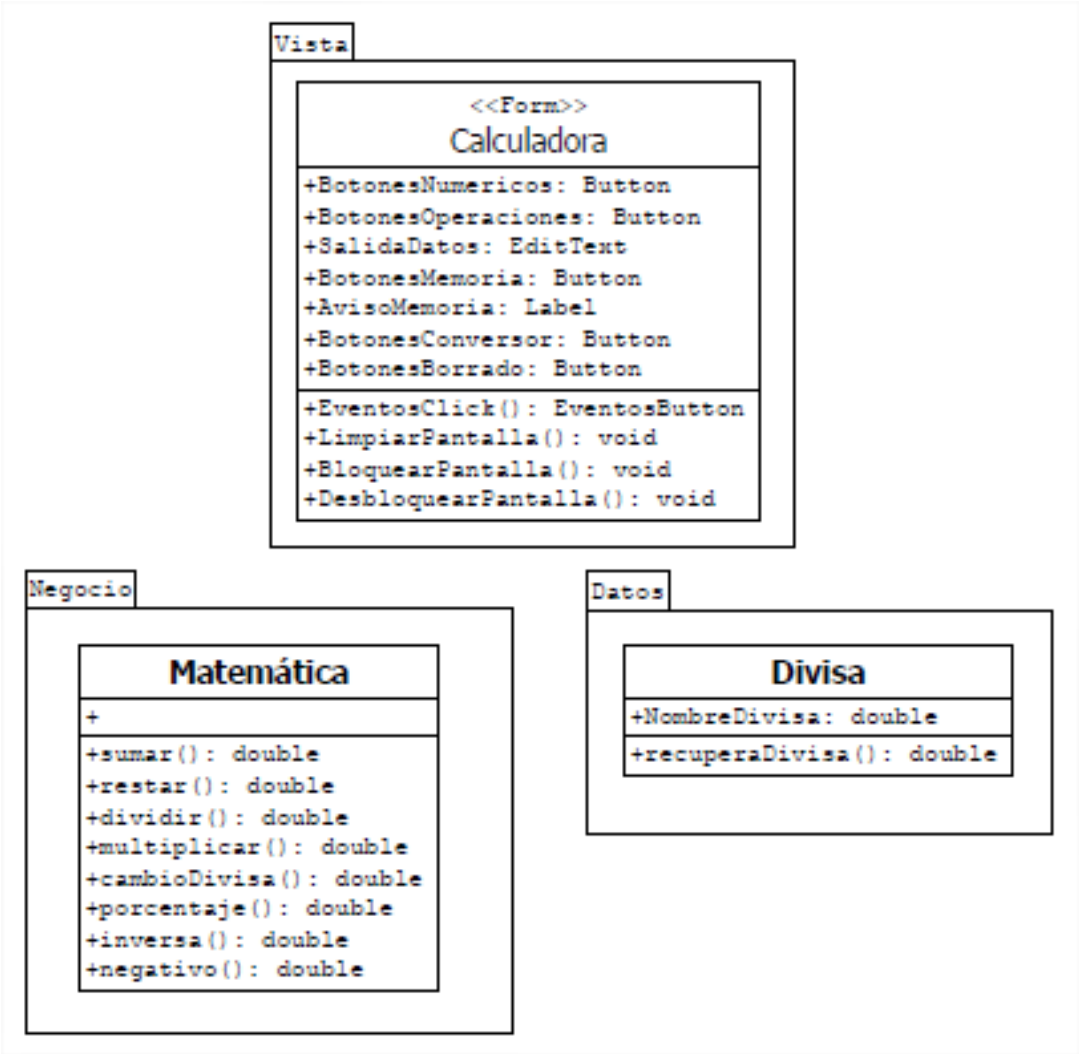


Figura 8. Diagrama UML Calculadora Android.

Y el diagrama de la estructura de la *aplicación Calculadora Windows*:

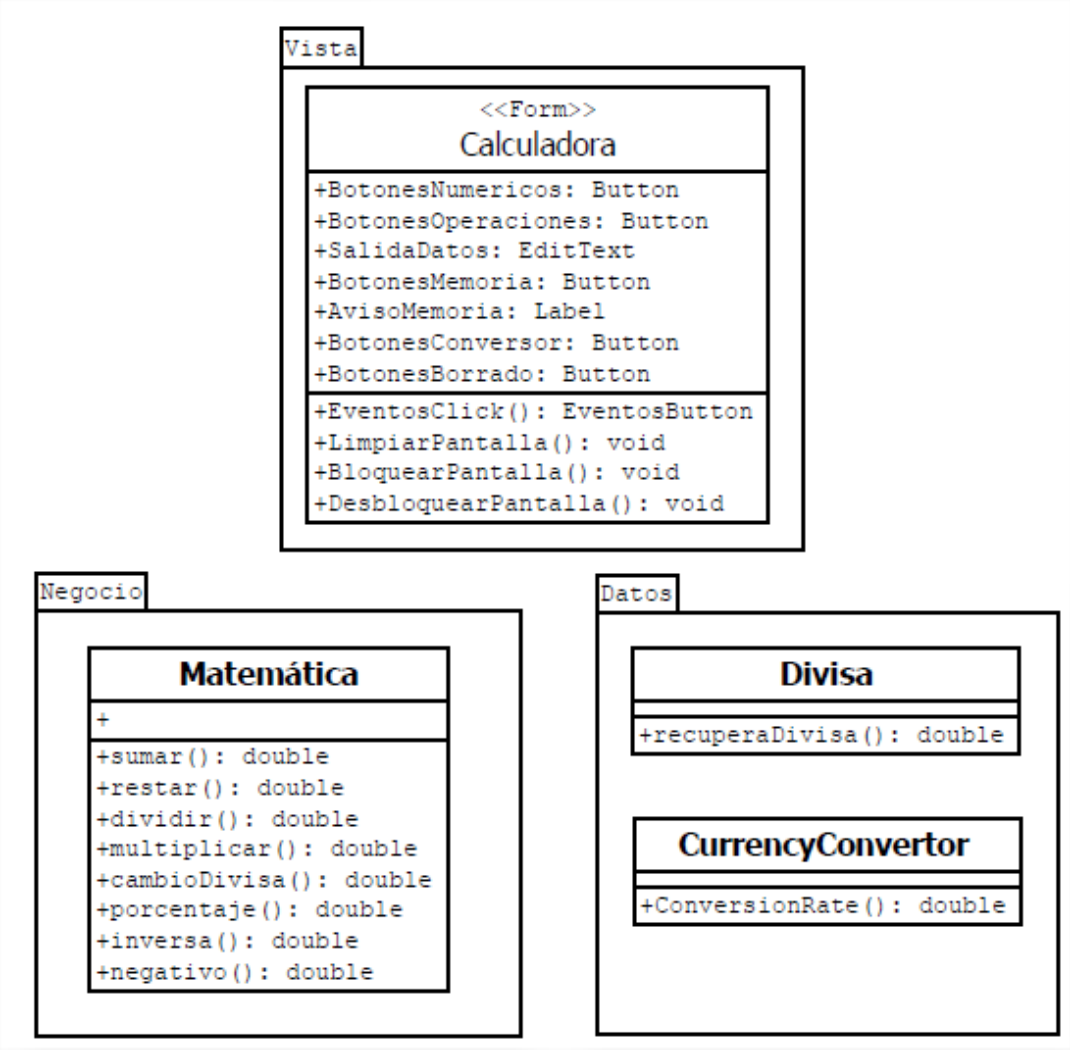


Figura 9. Diagrama UML Calculadora Windows.

En ambas aplicaciones, la Vista queda integrada por una única clase Calculadora, que es un Formulario que contiene todos los controles (botones, etiquetas y EditText) que componen la interfaz gráfica (pantalla/ventana única) cuyos métodos manejan los eventos OnClick de los botones (En el siguiente punto se explica cómo se realiza el diseño de formularios en Rad Studio).

Y la capa Negocio queda integrada por la clase Matemática, encargada de hacer todas las operaciones de la calculadora.

En la app Android, la capa de Datos queda integrada por la clase Divisa, que es la encargada de pasar a la clase Matemática el valor del cambio de la divisa deseada que está guardada en una constante atributo de la clase.

En la app Windows, la capa de Datos queda integrada por la clase Divisa, que es la encargada de pasar a la clase Matemática el valor del cambio de la divisa deseada, solicitando el valor de ésta a la clase CurrencyConvertor; y por la clase CurrencyConvertor, que es la encargada de solicitar este valor al Servicio Web.

4.3 . Diseño de la Vista

RAD Studio dispone de un Diseñador de Formularios.

Con él podemos diseñar visualmente un *formulario Master* (View: Master) viendo en tiempo real su aspecto final. Simplemente hay que arrastrar y posicionar los componentes que compondrán la Vista sobre el formulario, e internamente el IDE creará automáticamente el código del formulario en el archivo Vista.fmx.

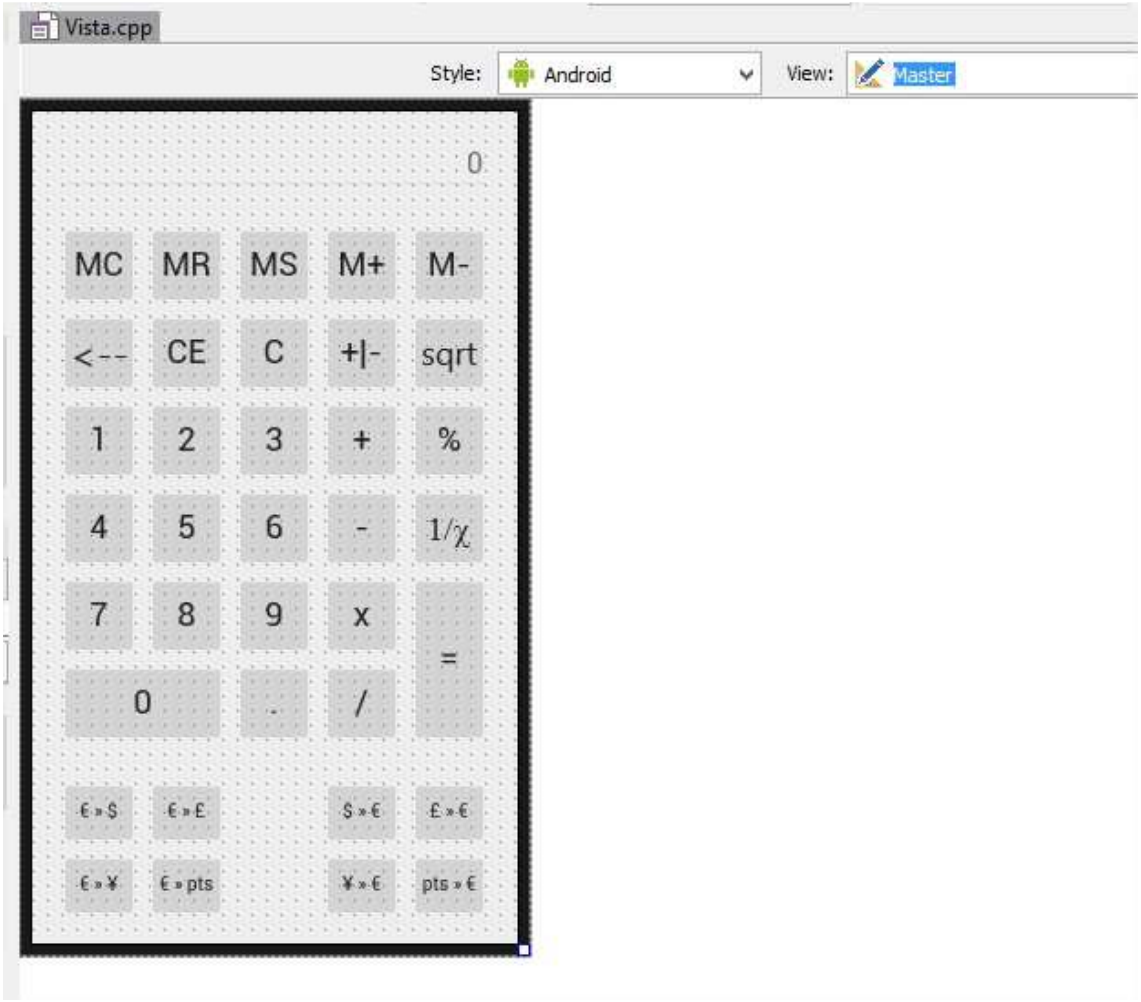


Figura 10. Diseño de la vista en modo visual.

El IDE también nos permite acceder al código del formulario para poder modificarlo. Para ello hay que pulsar el botón derecho del ratón sobre la vista y seleccionar "View as Text". Para volver al modo visual, hay que hacer la misma operación pero seleccionando "View as Form".

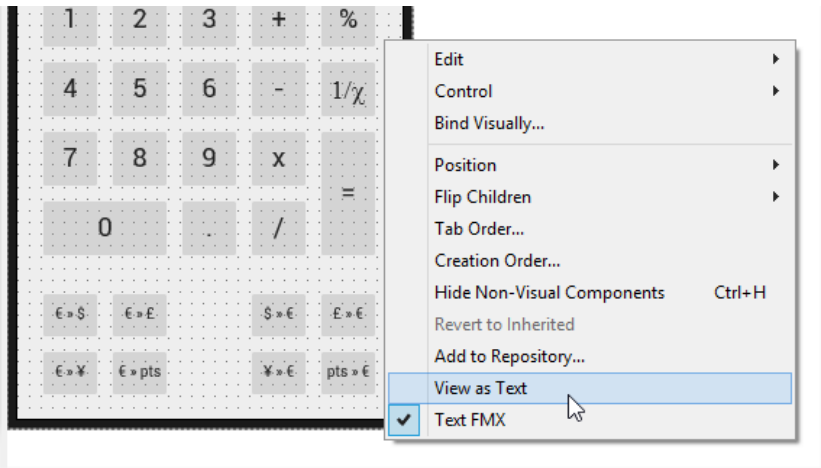


Figura 11. Diseñador de formularios. Cambio a modo Texto

Las modificaciones que hagamos sobre el archivo fmx, también modificarán el aspecto de nuestro Formulario.

.

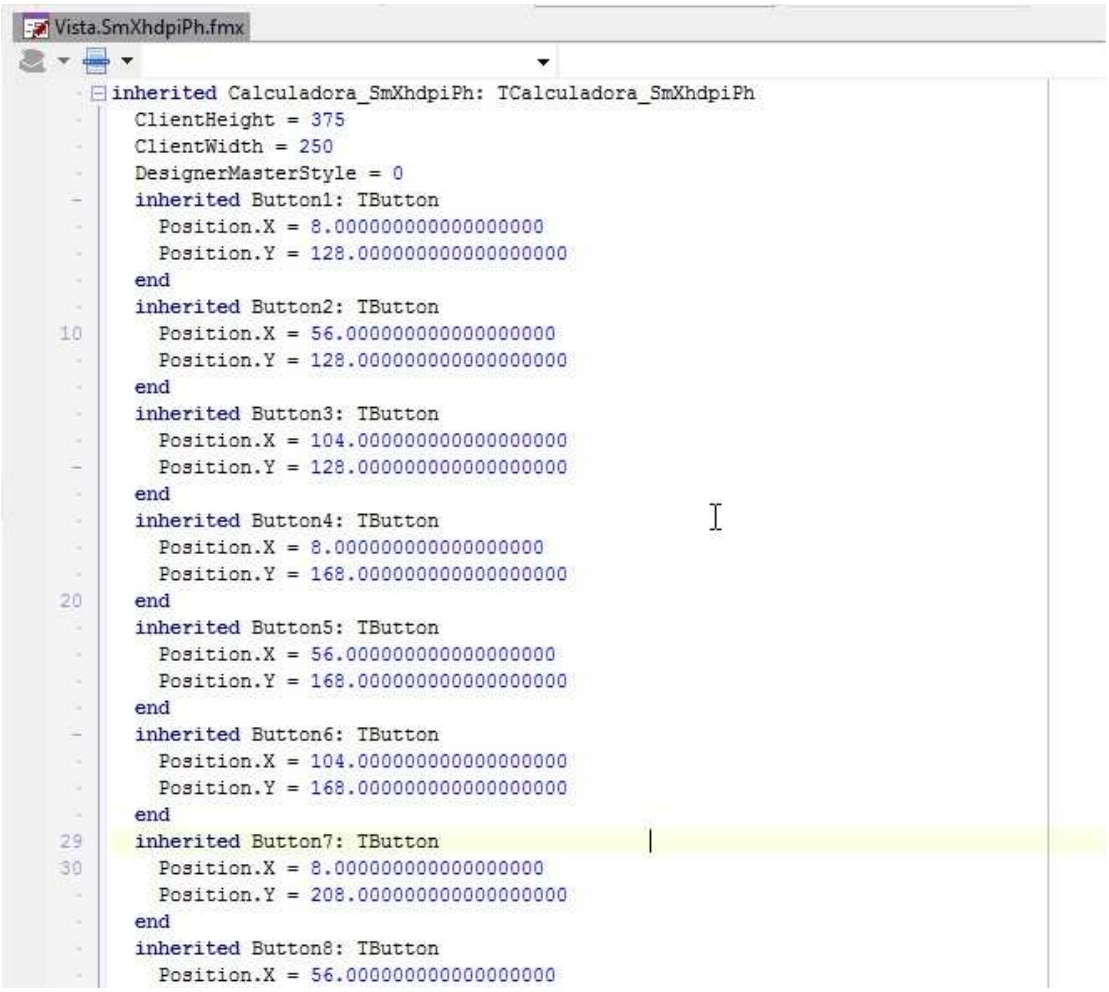


Figura 12. Diseñador de formularios. Archivo Vista.fmx

Una vez creado el diseño del formulario Master, hay que pensar en qué dispositivo va a correr la aplicación y modificar, si hiciese falta, el formulario correspondiente a ese dispositivo.

En el caso de la plataforma Windows, el diseño del formulario Master, se corresponde casi completamente con el diseño del formulario para esta plataforma (respeta al 100% los tamaños y las posiciones de los componentes), aunque los componentes pueden tener ciertas características diferentes. Por ejemplo, los componentes Windows son compatibles con las fuentes de texto "Symbol" y "Wingdings3" por lo que en los botones de Borrado y Raíz cuadrada, he podido cambiar el texto por los símbolos de una flecha y de la raíz.

Las modificaciones que hacemos sobre el formulario "Windows Desktop" no afectan al formulario Master.

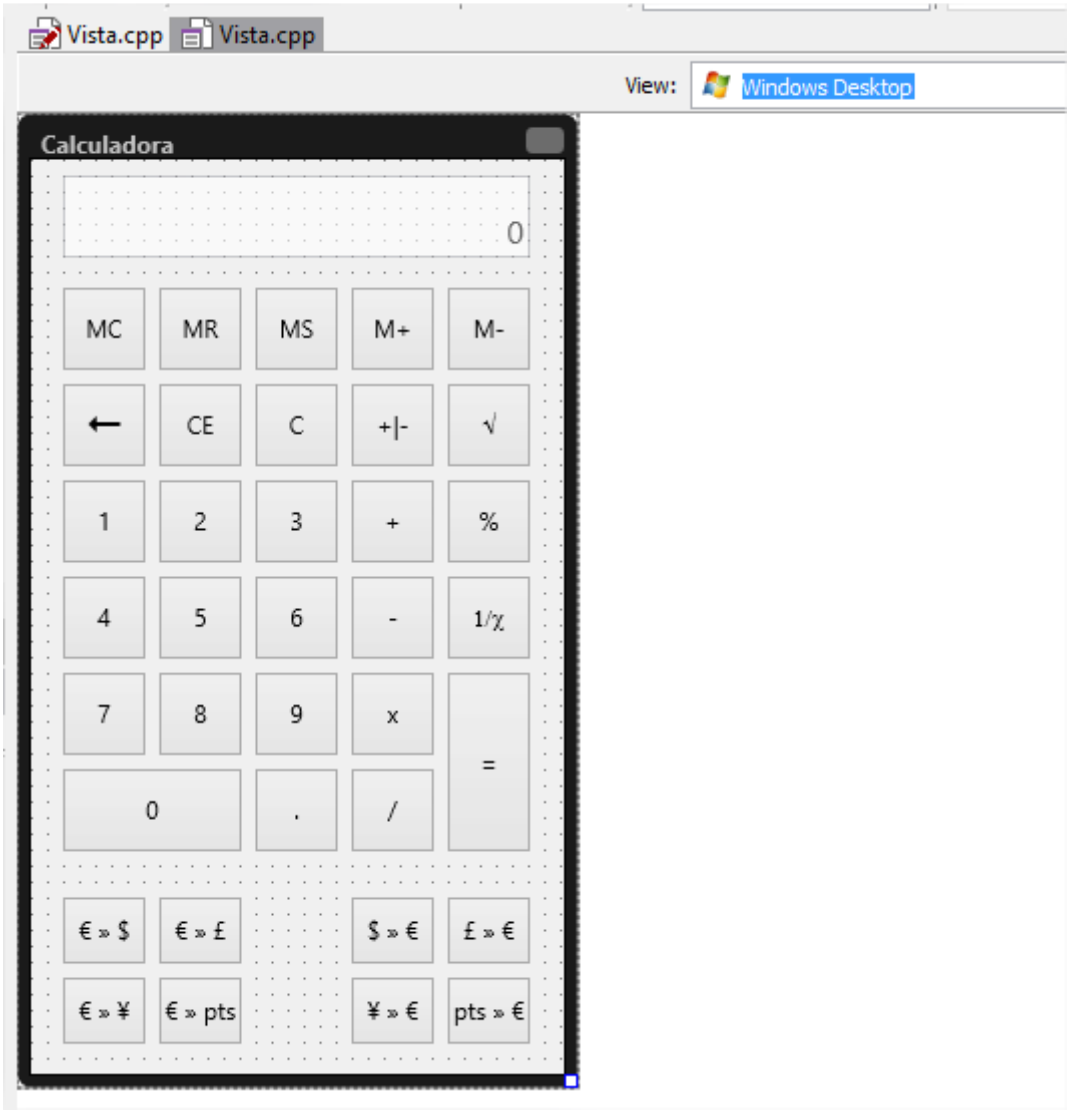


Figura 13. Formulario Windows. Modo visual

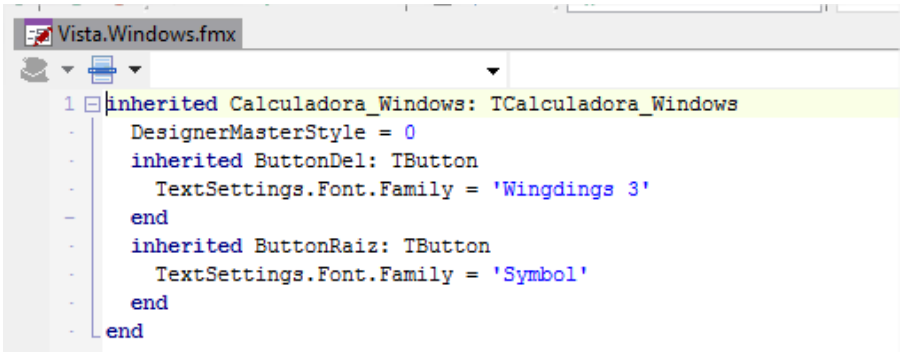


Figura 14. Formulario Windows. Modo texto.

El diseño de los formularios, es especialmente delicado en el caso de la programación de aplicaciones Android, ya que al ser multidispositivo, RAD Studio crea automáticamente los formularios para los dispositivos siguientes: Phones de 3.5", 4" y 5"; Tablet de 7" y 10"; Google Glass, Motorola Moto 360 y Sony SmartWatch 3.

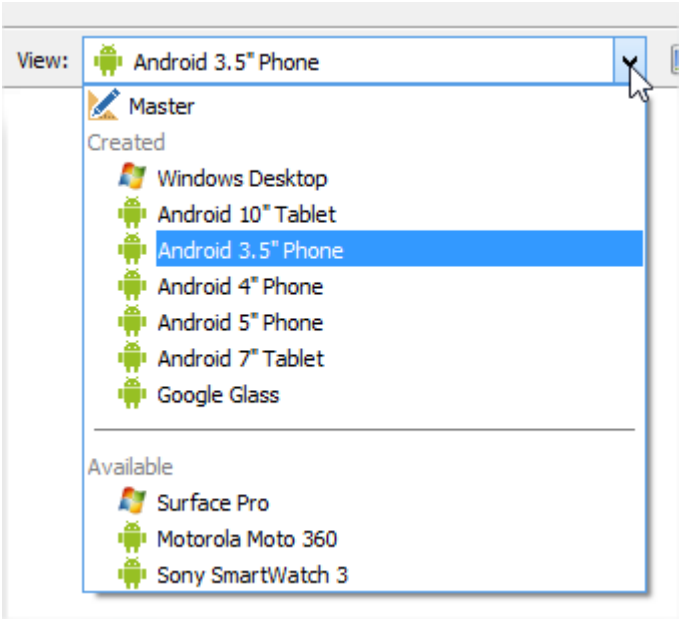


Figura 15. Formularios según dispositivo

Como la creación de estos formularios es automática y se basa en el tamaño y posición de los componentes del formulario Master, el tamaño y la posición de los componentes no será la adecuada a los diferentes tamaños de pantalla, ya que Rad Studio, no realiza ningún tipo de autoajuste.

En las imágenes siguientes, se ve el aspecto de las vistas creadas automáticamente a partir del formulario Master en dos dispositivos distintos: en el Phone 3,5" se observa que no se visualizan en su totalidad los componente de la derecha porque se salen de los márgenes de la pantalla, y en el Phone 5" que quedan muchos huecos anti-estéticos en la pantalla ya que los componentes son demasiado pequeños.

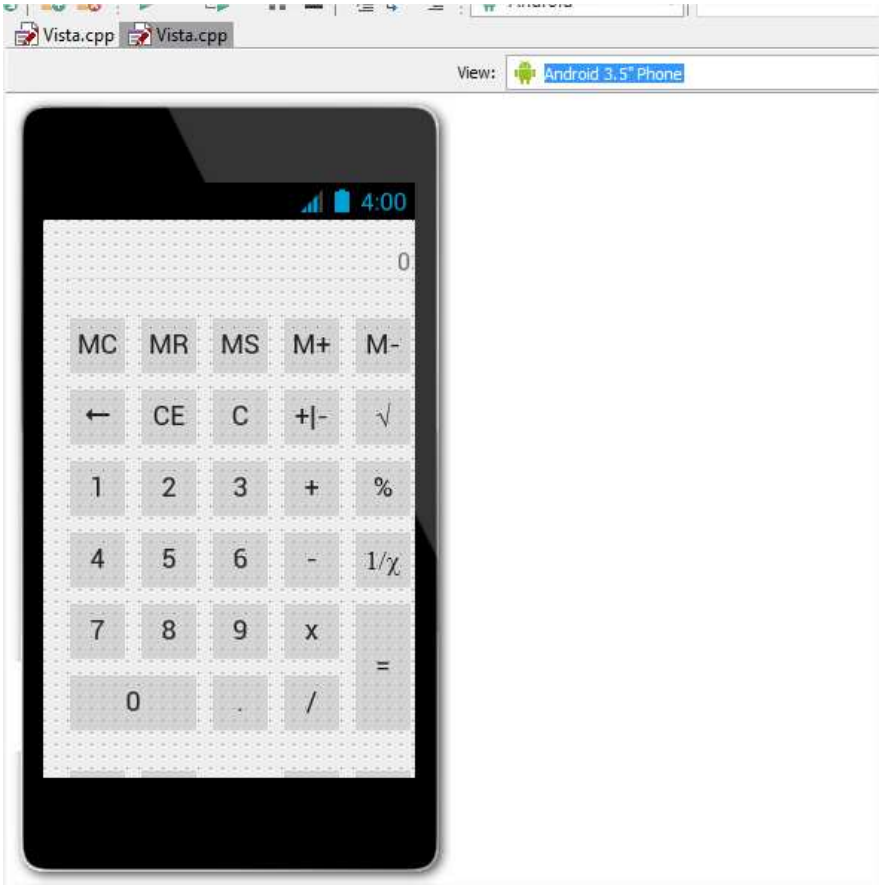


Figura 16. Formulario Android 3,5' sin ajustar

Figura 17. Formulario Android 5' sin ajustar

Por tanto, en el caso del desarrollo para Android es obligatorio rectificar el tamaño y la posición de los componentes en todos los tamaños de pantalla.

Al igual que con el formulario Windows, las modificaciones sobre los formularios de cada dispositivo Android, no modifican el formulario Master.

Figura 18. Formulario Android 3,5' ajustado

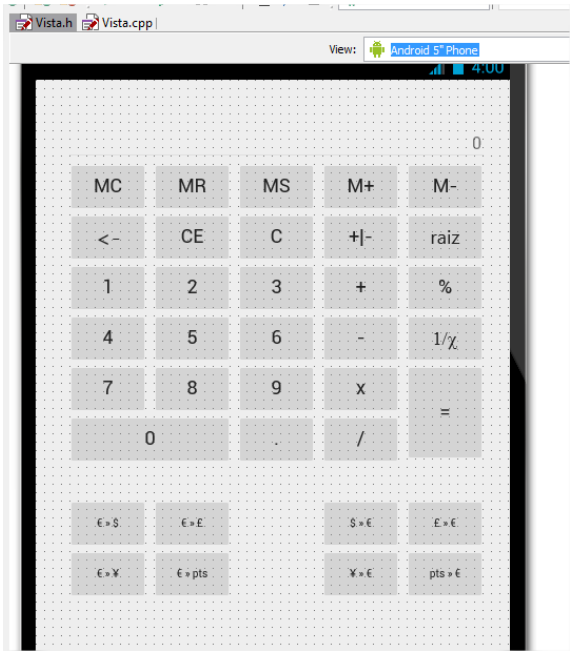


Figura 19. Formulario Android 5' ajustado

Cada uno de los formularios Android genera un archivo de texto diferente, cuyo nombre indica el tamaño de la pantalla. La tabla siguiente muestra los nombres de estos archivos y su tamaño correspondiente.

	Nombre archivo formulario
Master	NombreFormularioMaster.fmx
Phone 3.5"	NombreFormularioMaster. **SmXhdpiPh**.fmx
Phone 4"	NombreFormularioMaster. **NmXhdpiPh**.fmx
Phone 5"	NombreFormularioMaster. **LgXhdpiPh**.fmx
Tablet 7"	NombreFormularioMaster. **LgXhdpiTb**.fmx
Tablet 10"	NombreFormularioMaster. **XLgXhdpiTb**.fmx
Google Glass	NombreFormularioMaster. **GGlass**.fmx

A pesar de que ajusté cada uno de los formularios Android, pude comprobar, que una vez instalada la aplicación en los dispositivos físicos, los tamaños y posición de los componentes, no se ajustan del todo a la pantalla, si el dispositivo no es el que ha realizado la compilación de la aplicación. En el caso del dispositivo que realiza la compilación la pantalla se ve perfeca.

La siguiente imagen muestra un dispositivo (BQ Aquaris X5) con pantalla de 5" pulgadas ejecutando la aplicación compilada con un dispositivo (Nexus 4 LG) con pantalla de 4,7". Se puede ver, que a pesar de disponer una pantalla de mayor tamaño y de ser del tamaño de uno de los formularios Android proporcionado por Rad Studio (Phone 5"), los compontes están un poco recortados en el margen derecho de la pantalla.

Figura 20. Error en el formulario tras la compilación. Captura de pantalla de un dispositivo real, realizada con la aplicación AirDroid

5. Ejecución de la aplicación en las diferentes plataformas.

Una vez hecha la compilación de la aplicación para ***Android***, ésta es instalada en nuestro dispositivo real/virtual Android, lista para poder ser ejecutada.

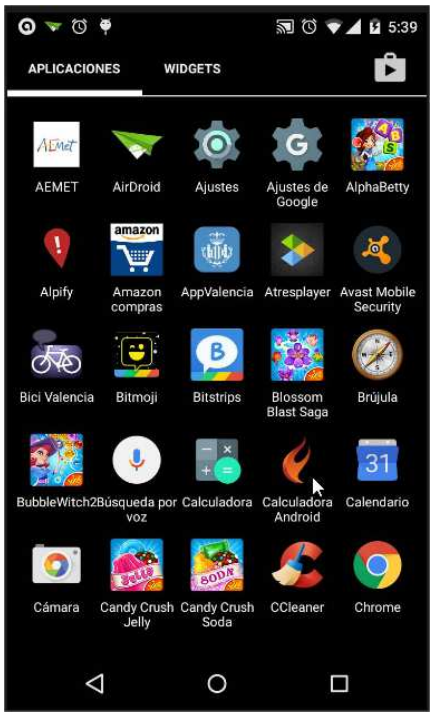


Figura 21. Aplicación Calculadora Android instalada en dispositivo Android real. Captura de pantalla realizada con la aplicación AirDroid

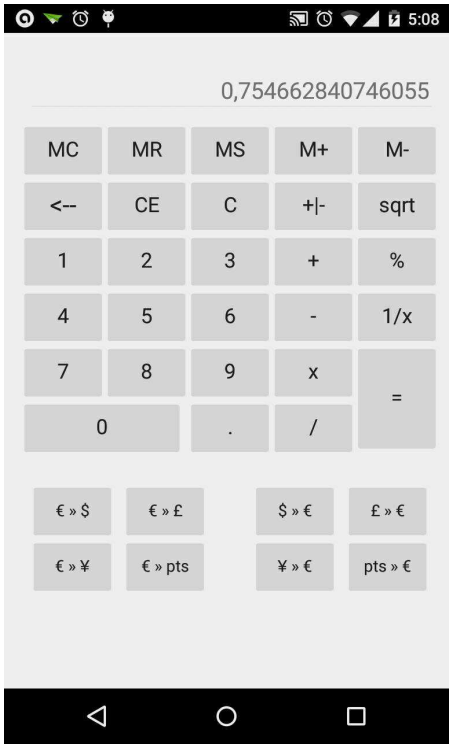


Figura 22. Aplicación ejecutándose en dispositivo Android real. aptura de pantalla de un dispositivo real, realizada con la aplicación AirDroid

Una vez hecha la compilación de la aplicación para ***Windows***, buscamos el archivo ejecutable en la carpeta bin de nuestro proyecto Rad Studio y lo llevamos a la plataforma para ejecutarlo.

Figura 23. Aplicación ejecutándose en dispositivo real.

Comparando las dos aplicaciones, vemos que ambas funcionan igual, excepto en el conversor de divisas.

En el caso de la aplicación de escritorio de Windows, el cálculo tarda algo más de un segundo en aparecer (el tiempo varía) en la conversión a dólares, yenes y libras (el valor del cambio a pesetas está almacenado en una constante y por tanto su acceso es más rápido), por la llamada que se hace al Web Service que nos devuelve el cambio real del día.

En el caso de la app de Android, los cálculos del cambio de divisas se realiza siempre con el mismo valor de cambio, puesto que éste está almacenado en unas constantes de la clase Divisa. Por ello, el cálculo es inmediato.

6. Conclusiones.

El objetivo de este proyecto era evaluar la herramienta Rad Studio, desarrollando una aplicación de ejemplo y probando su funcionamiento en varias plataformas. La complejidad de la aplicación de ejemplo desarrollada ha estado condicionada, tanto por la compatibilidad entre plataformas y las herramientas proporcionadas por Rad Studio, como por mi nivel de conocimiento del lenguaje de programación C++ (principiante).

Por tanto, he desarrollado una calculadora que, además de las operaciones básicas, convierte divisas obteniendo el ratio de cambio a través de un servicio Web.

Para probar alguna de las herramientas avanzadas proporcionadas por Rad Studio, la clase avanzada encargada de la llamada al WSDL en la Calculadora ha sido implementada con la herramienta *WSDL Wizard.*

Esta aplicación ha sido probada en dispositivos Windows y Android. No se ha podido desarrollar y probar en Mac e IOS porque no dispongo de dispositivos con estos sistemas operativos.

Una de las ventajas que destaca Embarcadero de su IDE, es la posibilidad de reutilización del código, por lo que para aprovechar al máximo esta ventaja, diseñé la aplicación usando una Arquitectura en 3 capas. Con la que se puede reutilizar todo el código que compone la capa de Negocio y gran parte del código de la capa de Presentación y la capa de Datos (ya que sólo hay que modificarlas en caso de ser necesaria su adaptación a cada plataforma).

En el desarrollo de mi aplicación he tenido que adaptar la capa de Datos en cada plataforma. Ya que la herramienta *WSDL Wizard* de C++ Builder escribe automáticamente un código que compila y ejecuta sin errores en la plataforma Windows. Pero que, aunque compila sin errores, no llega a ejecutarse en la plataforma Android.

Aun habiendo consultado la documentación sobre la herramienta *WSDL Wizard de* C++ Builder no he sido capaz de resolver esta incidencia. Y aunque la documentación no indica ninguna incompatibilidad de esta herramienta con la plataforma Android, no puedo afirmar que el error se deba a un defecto de Rad Studio. Puesto que podría ser que la información proporcionada en la Wiki de Embarcadero no esté actualizada. En ese caso, el error sería documental y no funcional.

Otro funcionamiento anómalo encontrado a la hora de desarrollar mi aplicación para Android es que, en ocasiones, se producen errores al enlazar el IDE con Genymotion o con el dispositivo físico, sobre todo cuando lo que se está realizando es un "debug" de la aplicación y no sólo su compilación. De hecho, en la versión 10 si pude compilar y ejecutar mi aplicación con un dispositivo virtual Genymotion, pero tras la actualización a la versión 10.1 de Rad Studio, esto ha sido imposible. Tal vez, sea debido a que el uso de dispositivos virtuales en esta nueva versión requiera de alguna configuración extra que no está indicada en la documentación.

Un inconveniente que para mí tiene el IDE, es que Embarcadero saca versiones nuevas con una periodicidad demasiado corta (cada 5-6 meses hay una versión nueva). Estas versiones requieren ser descargadas, instaladas y configuradas cada vez, lo cual te resta tiempo de trabajo. Además, en la actualización de la versión 10 a la 10.1, he podido comprobar que todos los archivos de programa de la versión 10 se han mantenido en el equipo, de hecho, si tuviese una licencia vigente de la versión 10 podría seguir trabajando con ella aun habiendo actualizado a la 10.1.

Puede que sea una opinión más personal que técnica, pero considero más adecuado que una actualización desinstale la versión anterior (o al menos recuerde al usuario que lo haga él mismo) para evitar la acumulación de archivos basura en el equipo (más si se va a estar actualizando cada 5-6 meses).

Por todos los problemas de compatibilidad con Android que he ido encontrando, considero que para poder evaluar la calidad del IDE, sería necesario también el estudio de su módulo Delphi, para poder probar si con las herramientas de este módulo mejora la compatibilidad.

En conclusión, Rad Studio, cumple parcialmente lo que promete, ya que permite desarrollar aplicaciones multi-plataforma con una única herramienta y un único lenguaje de programación, pero estas aplicaciones deben ser básicas si queremos crearlas a partir de un código fuente común y compartido (que es lo que prometen).

Bibliografía.

EMBARCADERO TECHNOLOGIES. *AboutUs*. <http://www.embarcadero.com/company/about-us> [Consulta: Junio 2015]

WIKIPEDIA. *Plataforma informática*.<http://es.wikipedia.org/wiki/Plataforma_(inform%C3%A1tica)> [Consulta: Junio 2015]

WIKIPEDIA. *Android.*<https://es.wikipedia.org/wiki/Android> [Consulta:Junio 2015]

WIKIPEDIA. *iOS*<https://es.wikipedia.org/wiki/IOS> [Consulta: Agosto 2015]

WIKIPEDIA. *Historia de MAC OS*< https://es.wikipedia.org/wiki/Historia_de_Mac_OS> [Consulta: Octubre 2015]

WIKIPEDIA. *Anexo: Historial de versiones de Android.* . <https://es.wikipedia.org/wiki/Anexo:Historial_de_versiones_de_Android> [Consulta: Noviembre 2015]

APPLE. *Development Resources* <https://developer.apple.com/support/resources/index.html> [Consulta: Septiembre 2015]

APPLE. *iOS Technology Overview*. <https://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/iOSTechOverview.pdf> [Consulta: Septiembre 2015]

Nicolás Montés. Blogs CEU (Universidad Cardenal Herrera). *Ranking de sistemas operativos más usados para 2015.* < https://blog.uchceu.es/informatica/ranking-de-sistemas-operativos-mas-usados-para-2015/ > [Consulta: Junio 2015]

Jesús Tomás Gironés. EDX. *Curso Android: Introducción a la Programación.* . <https://www.edx.org/course/android-introduccion-la-programacion-upvalenciax-aip201x> [Consulta: 8 de junio 2015]

Microsoft. *Historia de Windows* . <http://windows.microsoft.com/es-es/windows/history#T1=era0> [Consulta: Octubre 2015]

TUTORIALPOINT. *Learn C++.* . <http://www.tutorialspoint.com/cplusplus/index.htm> [Consulta: Julio y Agosto 2015]

Jesus Conde. *Curso de C++.* . <https://www.youtube.com/watch?v=Dir9aDGAeec&list=PLFDA837BC005D3614> [Consulta: Julio, Agosto y Septiembre 2015]

Web Service. WSDL. . <http://www.webservicex.com/CurrencyConvertor.asmx?wsdl> [Consulta: Enero 2016]