UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERÍAS INDUSTRIALES

INGENIERO INDUSTRIAL

PROYECTO FIN DE CARRERA

APLICACIÓN ANDROID PARA LA RESOLUCIÓN DE ESTRUCTURAS 2D DE BARRAS.

Autor del proyecto:

González Arias, Javier

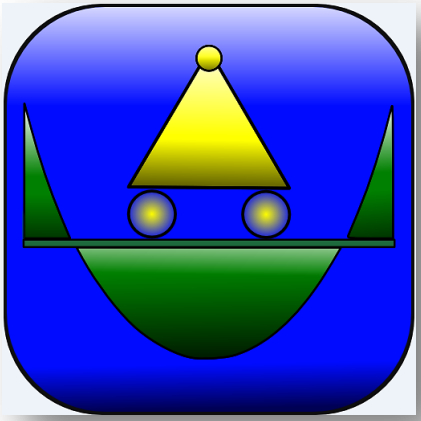
Tutor:

Pereda Llamas, José — **Dpto. Construcciones Arquitectónicas, Ingeniería del Terreno, Mecánica de los Medios Continuos y Teoría de Estructuras**

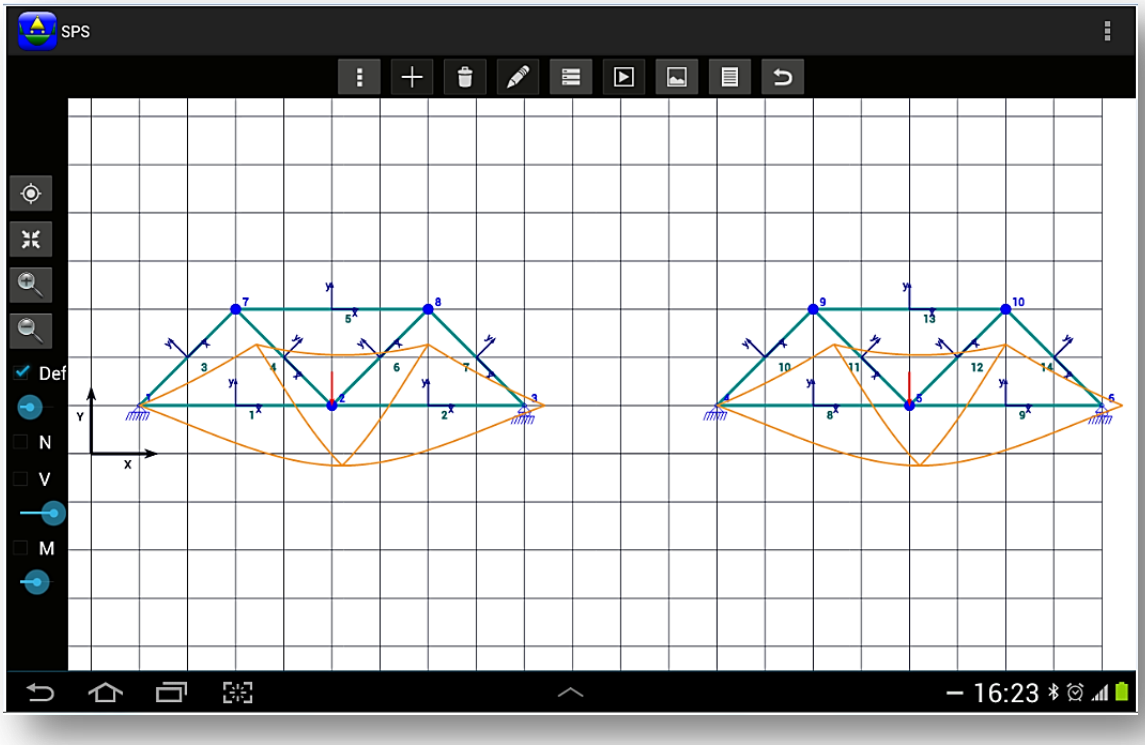
MAYO — 2014

1 RESUMEN

La aplicación Android SPS es una herramienta que nos permitirá resolver estructuras planas 2D que el usuario podrá construir y visualizar a su antojo, el motor de cálculo está basado en el Método Directo de Rigidez y la salida de resultados contempla todo tipo de diagramas, listados y visualización de deformada.



Captura 1: Icono de la aplicación



Captura 2: Deformada de 2 cerchas no articuladas obtenida con la aplicación

2 INTRODUCCIÓN Y JUSTIFICACIÓN DEL PROYECTO

Analizando el desarrollo tecnológico que ha experimentado la Sociedad desde mediados del siglo XX hasta la actualidad, se podría señalar que la verdadera revolución de este periodo ha sido el descubrimiento de la informática y su progresiva implantación en todas las disciplinas, desde las del conocimiento hasta las de producción, así como su introducción en el común de la población, lo que ha provocado un cambio en nuestra sociedad y nuestra economía, este cambio se puede considerar uno de los más rápidos que ha sufrido la sociedad.

El actor inicial de esta gran revolución es sin duda el computador u ordenador. Aparecieron primero como enormes y costosas máquinas que solamente estaban disponibles en importantes universidades o centros de investigación. Y debido a la retroalimentación de esta revolución, provocó el desarrollo de nuevas técnicas de fabricación, como los circuitos integrados, reducción de tamaño, aumento de capacidades, uso de nuevos materiales, lo que desencadenó una reducción de su precio, esto lo catapultó a ser un producto de masas, como podría ser la televisión o la radio. Con la aparición de Internet, el ordenador obtuvo un grandísimo empuje llegando a poder afirmar sin cometer excesivos errores, que los ordenadores están presentes en la vida social, laboral o académica de una persona, y por tanto son imprescindibles en las actividades cotidianas de esta.

Pero la implantación de Internet como servicio abierto, a principios de la década de 1990, llevo consigo el despegue de otro medio de comunicación, que si bien era más antiguo, se reinventaba a sí mismo gracias a los cambios en su tradicional soporte: la telefonía móvil.

El uso de la telefonía móvil, ayudado también por Internet no sólo ha provocado o provoca una revolución tecnológica sino un cambio sociológico donde la comunicación y acceso a la información en cualquier momento y lugar representan la gran demanda del usuario.

Debido a esta imparable evolución y a la gran demanda por parte de la población de comunicación e información, los dispositivos cada vez reducen más el tamaño de sus componentes aumentando a la vez sus prestaciones, lo que ha permitido fusionar las características de un ordenador y de un teléfono móvil en un mismo dispositivo. Dando lugar a un dispositivo que no sólo puede hacer llamadas o enviar SMS, sino que además puede tener un acceso más o menos limitado a Internet, realizar fotografías, vídeos o uso de tecnología GPS.

Así pues, definiremos un dispositivo móvil como una amplísima familia de aparatos electrónicos surgidos en los últimos años, de reducido tamaño, que ofrecen alguna capacidad de procesamiento y almacenamiento de datos y que pueden desempeñar diversas funciones, comprende: desde los smartphones, a ordenadores portátiles, cámaras digitales, reproductores de música, tabletas, netbooks, consolas de videojuegos.

La mayoría de estos aparatos cuentan con un sistema operativo de mayor o menor complejidad, que permita realizar las tareas de gestión de memoria y control de hardware que precisan. En el caso de los ordenadores portátiles, con tanta o incluso mayor capacidad que los de sobremesa, los sistemas operativos habituales son perfectamente compatibles y funcionan sin diferencias. Sin embargo, en otros dispositivos móviles es preciso diseñar nuevos sistemas operativos adaptados específicamente a sus características: restricciones de memoria y procesamiento, consumo mínimo de energía o gran estabilidad en su funcionamiento, entre otros.

Algunos sistemas operativos para dispositivos móviles son: iOS, Windows Phone, BlackBerry OS, Linux, Symbian OS, Firefox OS, Ubuntu Touch (en desarrollo), y como no, Android.

2.1 Motivación del proyecto

Dada la creciente importancia de los dispositivos anteriormente expuestos y su implantación masiva, aumentando cada vez más sus prestaciones y capacidades, surge la idea de desarrollar una aplicación relacionada con la ingeniería, en concreto con el cálculo de estructuras, que se pueda difundir y hacer llegar al público para su uso sencillo y cotidiano, sin más que tener un Smartphone en el bolsillo con la aplicación descargada.

2.2 Objetivos del proyecto

El sistema operativo para dispositivos móviles de Google, Android, centra el desarrollo de este proyecto fin de carrera. Para poder dirigir con mayor éxito los esfuerzos por conocer y comprender las características de este nuevo sistema, es necesario fijar unos objetivos que abarquen las actividades que se pretenden realizar y, además, permitan al final de las mismas conocer el grado de desarrollo y cumplimiento alcanzado.

Por ello, los objetivos perseguidos en el desarrollo de este proyecto son los siguientes:

- Conocer las principales características del lenguaje Java. Este es el paso cero, antes de empezar a desarrollar en Android, debíamos analizar programas desarrollados en lenguaje Java, ya que Android se fundamenta en éste, para ello se estudiaron diversos programas desarrollados para ordenador en el Departamento de Teoría de Estructuras, de la misma índole que el que se trata en esta memoria, para poder hacer uso de lo ya disponible y así hacer uso de la regla de la programación promovida por el lenguaje Java de “escribe una vez, ejecuta en cualquier parte”.

- Conocer las principales características de Android. El primer paso para conocer este nuevo sistema debe consistir en indagar toda la información posible sobre él, a fin de conocer cuál es su arquitectura,

sus componentes básicos, y cuál es su comportamiento al ejecutar las aplicaciones, consultando la documentación pertinente.

- Estudiar el entorno de desarrollo de Android. Al lanzarse bajo una licencia de software libre, el SDK completo está disponible para cualquier desarrollador que desee descargarlo. Este incluye numerosas ayudas para comenzar a crear aplicaciones en Android, desde las API completas con todas las clases y paquetes, hasta herramientas de programación y un completo emulador para poder realizar pruebas. Todos estos elementos han de ser estudiados y explicados.

- Desarrollar la aplicación para Android. Una vez conocidas las características de este sistema, así como el entorno de desarrollo que ofrece y sus posibilidades, se creará la aplicación que aproveche algunas de las características más fundamentales de Android.

3 ESTADO DEL ARTE

A continuación haremos una descripción cualitativa del mundo en el que está enmarcado dicho proyecto fin de carrera. Para ello iremos desde lo más general o lo particular. Comenzaremos con los dispositivos móviles, para pasar a sus sistemas operativos y por ultimo llegar al mundo de las aplicaciones.

3.1 Dispositivos móviles

La definición de dispositivo móvil más sencilla es la siguiente: aquel dispositivo que disfruta de autonomía de movimiento y está libre de cableado.

Pero en el momento actual no podemos hacer una definición rigurosa de que se entiende por dispositivo móvil, ya que hoy en día este término se utiliza para designar únicamente a ciertos modelos de teléfonos móviles con unas determinadas prestaciones. A pesar de esto, un dispositivo móvil no tiene por qué restringirse solo a la telefonía móvil.

Debido a lo anterior, se podría denominar dispositivo móvil a todo aparato electrónico que cumple unas características:

- Tamaño reducido que hace cómodo su transporte.
- Cuenta con una cierta capacidad de computación y almacenamiento de datos.
- Incorpora elementos de Entrada y Salida de Información (por lo general, pantalla y/o algún tipo de teclado).

Pero también existen, además de estas características básicas, otras que están presenten en dichos dispositivos móviles: como la conexión telefónica (incluyendo servicios como el envío de SMS, MMS, y acceso WAP) o la conexión a Internet, cámara fotográfica y de vídeo, pantalla táctil, teclado, receptor de radio, Bluetooth, conexión mediante infrarrojos, dispositivos de memoria extraíbles, localizador GPS, acelerómetro, o aplicaciones para: videojuegos, información meteorológica, etc.

En definitiva un sinfín de características no todas presentes en todos los dispositivos móviles, haciendo que este género sea muy heterogéneo. Esta versatilidad conlleva crear una plataforma común que permita homogeneizar y a la vez la intercambiabilidad entre dispositivos, surgiendo así los sistemas operativos móviles.

3.1.1 Clasificación de los dispositivos móviles

Existen diversas clasificaciones en función de los parámetros y características que se tengan en cuenta

La clasificación propuesta a continuación clasifica los dispositivos según la función o servicio principal del dispositivo en cuestión:

3.1.1.1 Dispositivos de comunicación

Un dispositivo móvil cuya función principal es la comunicación, principalmente comunicación telefónica. Además estos dispositivos pueden disponer de otros servicios de comunicación como SMS, MMS o acceso WAP.

En esta categoría se incluiría el tradicional teléfono móvil, precursor indiscutible dentro de los dispositivos móviles, la BlackBerry y el Smartphone.

3.1.1.2 Dispositivo de computación

Los dispositivos de computación son aquellos dispositivos móviles que ofrecen mayores capacidades de procesamiento de datos y cuentan con una pantalla y teclado más cercanos a un ordenador de sobremesa.

Dentro de este grupo encontramos a las PDA, ordenador portátil, que dentro de los dispositivos móviles quizás puede ser el más disonante con el resto debido a su mayor tamaño, peso y precio, pero como ventajas, está el de mayor rendimiento tanto de computación como de almacenamiento de datos.

3.1.1.3 Reproductor/Grabador multimedia

Un dispositivo móvil que esta específicamente diseñado para reproducir y/o grabar en uno o varios formatos de datos digitales, ya sean de audio, imagen o vídeo.

Dentro de estos dispositivos encontramos reproductores de MP3, DVD portátiles, eBooks, iPod de Apple, cámaras fotográficas digitales y cámaras de video digitales.

3.1.1.4 Consola portátil

Una consola portátil es un dispositivo móvil cuya única función es la de proporcionar al usuario una plataforma de juego. Las consolas portátiles fueron, junto a los teléfonos, los primeros dispositivos móviles en convertirse en un producto de masas. Hoy en día representan un importantísimo volumen de ventas dada su gran aceptación en la sociedad y son objeto de auténticas guerras comerciales entre las principales compañías del sector. Algunos ejemplos de esta categoría son la Nintendo DS de Nintendo, o la PSP de Sony.

3.1.1.5 Tabletas

Mención aparte requieren las tabletas, son dispositivos móviles que si bien en el momento actual su principal función no está muy clara, pero podemos decir que no es la comunicación, como la de un Smartphone. Comparten funciones con el resto de grupos ya que son capaces de reproducir y/o grabar en formatos digitales, soportan videojuegos, etc.

Cierto es, que las prestaciones que ofrecen en comparación con sus rivales para una determinada función no son muy grandes, pero estamos ante un dispositivo muy versátil.

3.1.2 Sistemas operativos para dispositivos móviles

El sistema operativo destinado a controlar un dispositivo móvil necesita ser fiable y tener una gran estabilidad, ya que incidencias habituales y toleradas en ordenadores personales como reinicios o caídas no tienen cabida en un dispositivo de estas características. Además, a de adaptarse adecuadamente a las consabidas limitaciones de memoria y procesamiento de datos, proporcionando una ejecución exacta y excepcionalmente rápida al usuario.

Estos sistemas han de estar perfectamente testeados y libres de errores antes de incorporarse definitivamente a la línea de producción. Las posibilidades que existen en un ordenador estándar de realizar actualizaciones e incluso reinstalar mejores versiones del sistema para cubrir fallos o deficiencias son más limitadas en un dispositivo móvil.

Es posible incluso que un aparato de esta naturaleza deba estar funcionando ininterrumpidamente durante semanas e incluso meses antes de ser apagado y reiniciado, a diferencia de lo que ocurre con un ordenador personal. El consumo de energía es otro tema muy delicado: es importante que el sistema operativo haga un uso lo más racional y provechoso posible de la batería, ya que esta es limitada y el usuario siempre exige una mayor autonomía.

En la actualidad, existen varios sistemas operativos para toda la gama de dispositivos móviles. Más adelante veremos por qué se ha elegido Android para la realización de este proyecto fin de carrera, pero antes veamos las características más importantes de los principales sistemas operativos móviles.

3.1.2.1 Ubuntu Touch

Ubuntu Touch es un sistema operativo móvil basado en Linux desarrollado por Canonical. Culmina el proceso de Canonical para desarrollar una interfaz que pueda utilizarse en ordenadores de sobremesa, portátiles, netbooks, tablets y teléfonos inteligentes. Esta interfaz, Unity, se compone, a grandes rasgos, de un dock a la izquierda, una especie de panel en la parte superior y un sistema de búsqueda que emplea "lentes".

Ubuntu Touch se caracteriza por ser un sistema diseñado para plataformas móviles. Ubuntu Touch utiliza el Qt 5 basado en la interfaz de usuario táctil y varios marcos de software desarrollados originalmente para Maemo y MeeGo como oFono. Además cuenta con un inicio de sesión único, utilizando libhybris, sistema que se usa con núcleos Linux utilizados en Android, lo que hace que sea fácilmente portado a los últimos teléfonos inteligentes Android.

Además, utiliza las mismas tecnologías esenciales del Escritorio de Ubuntu, por lo que las aplicaciones diseñadas para esta plataforma pueden ser usadas en ambas. Además, los componentes de escritorio de Ubuntu vienen con el sistema Ubuntu Touch, permitiendo que los dispositivos táctiles de Ubuntu puedan proporcionar una completa experiencia de escritorio cuando se conecta a un monitor externo. Los dispositivos táctiles de Ubuntu pueden estar equipados con una sesión completa de Ubuntu y pueden cambiar por completo el escritorio del sistema operativo cuando se conecta a una estación de acoplamiento. Si está conectado el dispositivo se pueden utilizar todas las características de Ubuntu y el usuario puede realizar trabajo de oficina o incluso jugar en ARM mediante el dispositivo. Algunas de sus características más destacadas son:

- Pantalla de inicio sin sistema de bloqueo/desbloqueo (que funciona con un nuevo sistema de gestos y se aprovecha para mostrar notificaciones).
- Ubuntu Touch incluye como aplicaciones centrales de medios sociales y medios de comunicación (por ejemplo, aplicaciones de Facebook, YouTube, y un lector de RSS). Las aplicaciones estándar, tales como una calculadora, un cliente de correo electrónico, un despertador, un gestor de archivos, e incluso un terminal están incluidos también.
- Integración con Ubuntu One.

3.1.2.2 Firefox OS (HTML5)

Firefox OS (nombre clave: Boot to Gecko o B2G) es un sistema operativo móvil, basado en HTML5 con núcleo Linux, de código abierto, a diferencia de Android, para varias plataformas. Es desarrollado por Mozilla Corporation bajo el apoyo de otras empresas y una gran comunidad de voluntarios de todo el mundo. El sistema operativo está diseñado para permitir a las aplicaciones HTML5 comunicarse directamente con el hardware del dispositivo usando JavaScript y Open Web APIs.

Inicialmente estuvo enfocado en los dispositivos móviles, smartphones y tabletas, incluidos los de gama baja. El verano de 2013, Telefónica comenzó la venta del primer terminal con Firefox OS, el ZTE Open que fue rápidamente seguido por el teléfono Peak de Geeksphone. También se pudo aplicar a otros dispositivos como Raspberry Pi, y en el desarrollo próximo de computadores de bajo consumo y televisores.

Los antecedentes del Firefox OS estaban relacionados con el futuro de los móviles mediante prototipos.

El proyecto Boot to Gecko se inició en el 2011, el plan era revolucionar el modelo enfocado en plataformas abiertas de bajos recursos económicos. Cuando fracasó la producción de los móviles WAC por conflictos en una de sus API usados en el sistema de pago, al año siguiente, Telefónica confirmó el apoyo a la fundación Mozilla.

Más tarde cambiaron el nombre a Firefox OS.

Se trata por lo tanto de una plataforma en creación y desarrollo que se estima irá creciendo en los próximos años. En el 2014, en la siguiente Mobile World Congress, Mozilla y Telefónica anunciaron nuevos terminales desarrollados por Alcatel, Huawei y ZTE. Según algunos analistas, la colaboración de Spreadtrum, una fabricadora china de hardware, a Mozilla en el desarrollo del modelo SC6821, llamado el celular de 25 dólares, podría determinar el fin de los feature phone.

3.1.2.3 Symbian

Fue producto de la alianza de varias empresas de telefonía móvil, entre las que se encuentran Nokia como la más importante, Sony Ericsson, Samsung, Siemens, BenQ, Fujitsu, Lenovo, LG, Motorola, etc. Esta alianza le permitió en un momento dado ser unos de los pioneros y más usados.

El acuerdo bajo el cual se desarrolló Symbian es bastante simple: Symbian Ltd. desarrolla el sistema operativo Symbian, que incluye el microkernel, los controladores, el middleware y una considerable pila de protocolos de comunicación e interfaces de usuario muy básicas. Los desarrolladores que obtienen la licencia correspondiente para trabajar con Symbian implementan sus propias interfaces de usuario y conjuntos de aplicaciones según las necesidades de sus propios dispositivos. Esto permitió a Symbian posicionarse como un sistema operativo muy flexible, que tenía en cuenta los requisitos de la mayoría de los dispositivos fabricados y, al mismo tiempo, permitía un alto grado de diferenciación.

En Symbian, una mínima porción del sistema tiene privilegios de kernel; el resto se ejecuta con privilegios de usuario en modo de servidores, de forma que los procesos en ejecución y sus prioridades son manejados por este microkernel. Cada una de las aplicaciones corre en su propio proceso y tiene acceso únicamente a una exclusiva zona de memoria.

Symbian contempla cinco tipos de ediciones o series del sistema operativo según las características del dispositivo móvil. La principal diferencia entre ediciones no radica tanto en el núcleo del sistema operativo como en la interfaz gráfica utilizada que dependerá del fabricante de smartphone.

Las aplicaciones compatibles con Symbian se desarrollan a partir de lenguajes de programación orientados a objetos como C++, Java (con sus variantes como PJava, J2ME, etc.), Visual Basic para dispositivos móviles, entre otros, incluyendo algunos lenguajes disponibles en versión libre.

3.1.2.4 BlackBerry O.S

El SO BlackBerry es un sistema operativo móvil desarrollado por Research In Motion para sus dispositivos BlackBerry. El sistema permite multitarea y tiene soporte para diferentes métodos de entrada adoptados por RIM para su uso en computadoras.

Su desarrollo se remonta la aparición de los primeros handheld en 1999. Estos dispositivos permiten el acceso a correo electrónico, navegación web y sincronización

con programas como Microsoft Exchange o Lotus Notes, aparte de poder hacer las funciones usuales de un teléfono móvil.

A parte de los dispositivos de la propia marca, otras marcas utilizan el cliente de correo electrónico de BlackBerry: Siemens, HTC, Sony Ericsson, etc. La mayoría de estos dispositivos cuentan con teclado QWERTY completo.

El SO BlackBerry está claramente orientado a su uso profesional como gestor de correo electrónico y agenda. Desde la versión actual, se puede sincronizar el dispositivo con el correo electrónico, el calendario, tareas, notas y contactos de Microsoft Exchange Server además es compatible también con Lotus Notes y Novell GroupWise.

BlackBerry Enterprise Server (BES) proporciona el acceso y organización del email a grandes compañías identificando a cada usuario con un único BlackBerry PIN. Los usuarios más pequeños cuentan con el software BlackBerry Internet Service, programa más sencillo que proporciona acceso a Internet y a correo POP3 / IMAP / Outlook Web Access sin tener que usar BES.

Al igual que en el SO Symbian desarrolladores independientes también pueden crear programas para BlackBerry pero en el caso de querer tener acceso a ciertas funcionalidades restringidas necesitan ser firmados digitalmente para poder ser asociados a una cuenta de desarrollador de RIM.

3.1.2.5 Windows Phone

Windows Phone es un sistema operativo móvil desarrollado por Microsoft, como sucesor de la plataforma Windows Mobile. Está pensado para el mercado de consumo generalista en lugar del mercado empresarial por lo que carece de muchas funcionalidades que proporciona la versión anterior. Microsoft ha decidido no hacer compatible Windows Phone con Windows Mobile por lo que las aplicaciones existentes no funcionan en Windows Phone haciendo necesario desarrollar nuevas aplicaciones.

Con Windows Phone Microsoft ofrece una nueva interfaz de usuario e integra varios servicios en el sistema operativo. Microsoft planeaba un estricto control del hardware que implementaría el sistema operativo, para evitar la fragmentación con la evolución del sistema, pero han reducido los requisitos de hardware de tal forma que puede que eso no sea posible.

Como se ha comentado antes Windows Phone es el sucesor de la versión del Windows Mobile, sistema operativo diseñado por Microsoft y orientado a una gran variedad de dispositivos móviles. En realidad, Windows Mobile representa una particularización de otro gran sistema de Microsoft llamado Windows CE.

A principios de la década de los 90, cuando comenzaron a aparecer los primeros dispositivos móviles, Microsoft tomó la decisión de crear un sistema operativo capaz de hacer frente al entonces recientemente lanzado por Apple, el sistema Newton MessagePad. Fruto de esta iniciativa surgió Pegasus, cuyo nombre comercial definitivo fue Windows Compact Edition, o Windows CE.

El objetivo principal que buscaba Microsoft era que el nuevo sistema fuera lo suficientemente flexible y adaptable para poder ser utilizados en un amplio abanico de dispositivos, cuyo única característica común es la de ser de reducido tamaño y tener, por tanto, una limitación obvia en sus recursos.

3.1.2.6 iOS

iOS (anteriormente denominado iPhone OS) es un sistema operativo móvil de Apple. Originalmente desarrollado para el iPhone, siendo después usado en dispositivos como el iPod Touch, iPad y el Apple TV. Apple Inc. no permite la instalación de iOS en hardware de terceros. Como ventajas podemos decir que posee una de las tiendas de aplicaciones y juegos más grande del mundo (App Store) y que su diseño y potencia permite disfrutar de una experiencia de usuario bastante agradable. También podemos destacar la integración de todo el sistema con la nube, gracias al iCloud, un nuevo sistema de almacenamiento que permite compartir todas tus aplicaciones, música y fotos con todos tus dispositivos de la misma marca.

La interfaz de usuario de iOS está basada en el concepto de manipulación directa, usando gestos multitáctiles. Los elementos de control consisten de deslizadores, interruptores y botones. La respuesta a las órdenes del usuario es inmediata y provee de una interfaz fluida. La interacción con el sistema operativo incluye gestos como deslices, toques, pellizcos, los cuales tienen definiciones diferentes dependiendo del contexto de la interfaz. Se utilizan acelerómetros internos para hacer que algunas aplicaciones respondan a sacudir el dispositivo (por ejemplo, para el comando deshacer) o rotarlo en tres dimensiones (un resultado común es cambiar de modo vertical al apaisado u horizontal).

iOS se deriva de Mac OS X, que a su vez está basado en Darwin BSD, y por lo tanto es un sistema operativo Unix.

Este sistema operativo no permite Adobe Flash ni Java. En cambio iOS usa HTML5 como una alternativa a Flash. Esta ha sido una característica muy criticada tanto en su momento como en la actualidad.

El jailbreak en iOS, es el proceso de remover las limitaciones impuestas por Apple en dispositivos que usen el sistema operativo a través del uso de kernels modificados. Tales dispositivos incluyen el iPhone, iPod Touch, iPad y la Apple TV de segunda generación. El jailbreak permite a los usuarios acceder al sistema de archivos del sistema operativo, permitiéndoles instalar aplicaciones adicionales, extensiones y temas que no están disponibles en la App Store oficial. Un dispositivo con jailbreak puede seguir usando la App Store, iTunes y las demás funciones normales, como realizar llamadas. El jailbreak es necesario si el usuario quiere ejecutar software no autorizado por Apple.

Las aplicaciones deben ser escritas y compiladas específicamente para la arquitectura ARM, por lo que las desarrolladas para Mac OS X no pueden ser usadas en iOS. Al igual que otros navegadores, Safari admite aplicaciones web. Aplicaciones

nativas de terceros están disponibles para dispositivos corriendo en iPhone OS 2.0 o posterior, por medio del App Store.

El SDK de Apple permite a los desarrolladores hacer aplicaciones para el iPhone e iPod Touch, así como probarlas en el "iPhone simulator". De cualquier manera, solo es posible utilizar la aplicación en los dispositivos después de pagar la cuota del iPhone Developer Program.

Estas aplicaciones, como las de Mac OS X, están escritas en Objective-C.

3.1.2.7 Android

Android en un sistema operativo móvil basado en el núcleo de Linux, mucho más simple que los sistemas operativos para PCs y orientado a la conectividad inalámbrica; que junto con aplicaciones middleware (conjunto de módulos como los motores de mensajería y comunicaciones, códecs multimedia, intérpretes de páginas web, gestión del dispositivo y seguridad) controla los dispositivos móviles, como Smartphone o teléfonos inteligentes, tabletas, Google TV.

El sistema Linux utiliza una máquina virtual que es la responsable de convertir el código escrito en Java de las aplicaciones a código capaz de comprender el Kernel.

Fue desarrollado inicialmente por Android Inc., una firma comprada por Google en 2005. Es el principal producto de la Open Handset Alliance, un conglomerado de 78 fabricantes y desarrolladores de hardware, software y operadores de servicio, las cuales llegaron a un acuerdo para promocionar los estándares de códigos abiertos para dispositivos móviles.

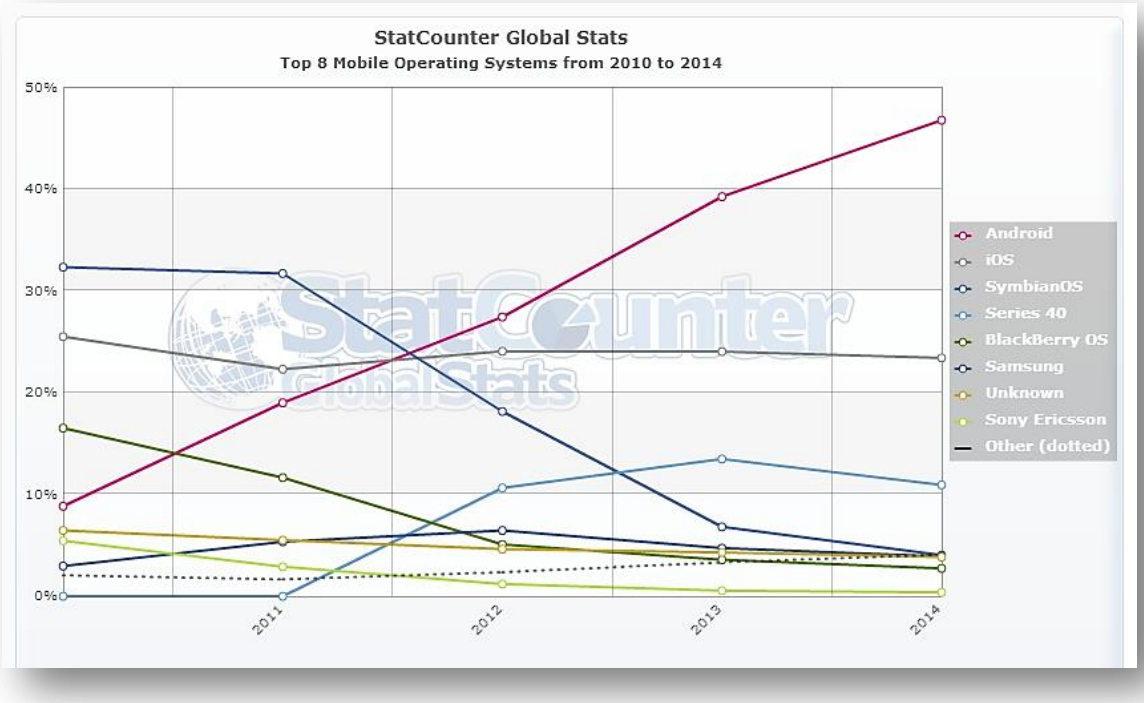
Sin embargo, ha sido Google quien ha publicado la mayoría del código fuente de Android bajo la licencia de Software Apache, una licencia de software libre y de código abierto a cualquier desarrollador.

Dicho código fuente de Android está basado en bibliotecas desarrolladas o adaptadas por Google mediante el lenguaje de programación Java, si bien es cierto que los componentes del sistema operativo están escritos en C o C++. Por lo que programar en Android resulta muy sencillo y no tenemos que invertir en licencias para hacerlo.

Las aplicaciones para Android se escriben y desarrollan en Java aunque con unas APIS propias por lo que las aplicaciones escritas en Java para PC y demás plataformas ya existentes no son compatibles con este sistema

3.1.2.8 Evolución de los sistemas operativos móviles

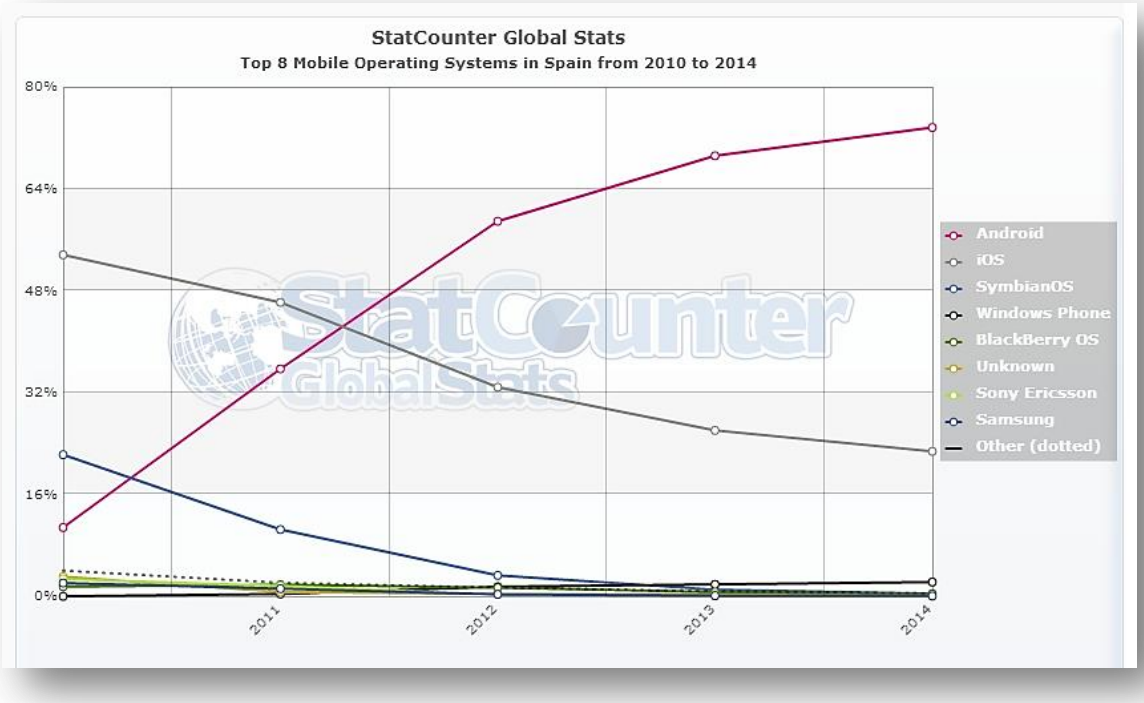
En primer lugar vamos a ver la gráfica que muestra la evolución de los principales sistemas operativos móviles en los últimos años, concretamente desde 2010 hasta el presente 2014.



Gráfica 1: Evolución Mundial SO Móviles desde 2010 a 2014

Se puede ver en la gráfica una clara tendencia desde hace dos años, que muestra a Android como el sistema operativo móvil más extendido a nivel mundial.

Hace varios años el sistema operativo Android desarrollado por Google despegó en el mercado y comenzó a competir con iOs de Apple como sistema operativo de referencia a nivel mundial, y 2013 fue el año en el que Android se afianzó como el sistema operativo móvil con mayor cuota de mercado (en torno al 80%), incluso llegando a desbancar a iOS como sistema operativo móvil más vendido durante ese año en EEUU.



Gráfica 2: Evolución SO Móviles desde 2010 a 2014 en España

Esa misma tendencia se puede ver en España, pero con unos valores de mercado aún más abrumadores, llegando a estar en torno al 75% en nuestro país. Sin duda el punto de inflexión fue el año 2010, a partir de aquí su incremento ha sido imparable hasta el momento.

Para ver mejor esta tendencia a nivel mundial y ver el desarrollo que Android ha tenido en estos últimos años, se han escogido tres mapamundis fechados en 2010, 2012 y 2014. Muestran el sistema operativo más utilizado en los diferentes países del mundo en esas fechas, y los resultados hablan por sí solos.

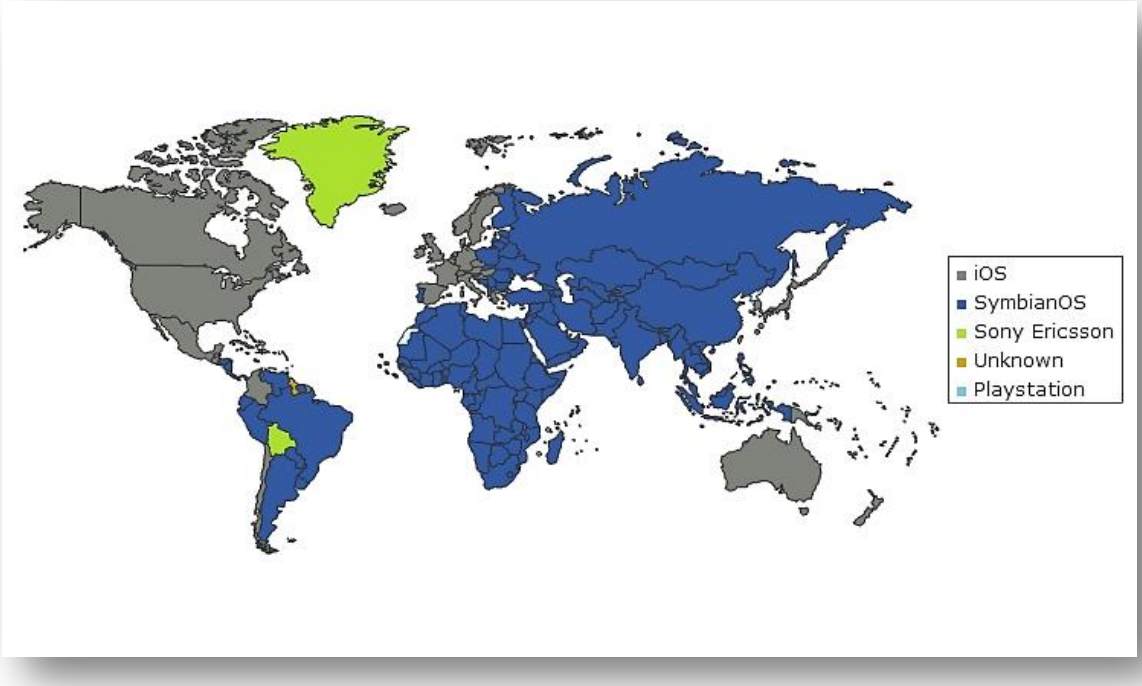


Imagen 1: SO Móviles más utilizados por países año 2010

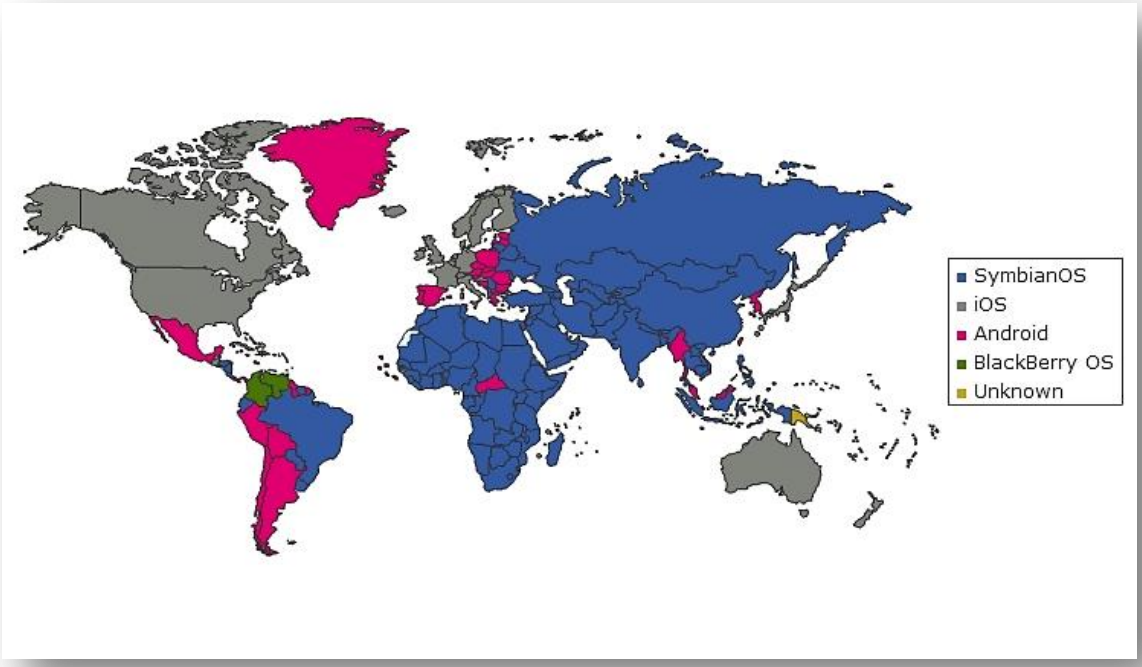


Imagen 2: SO Móviles más utilizados por países año 2012

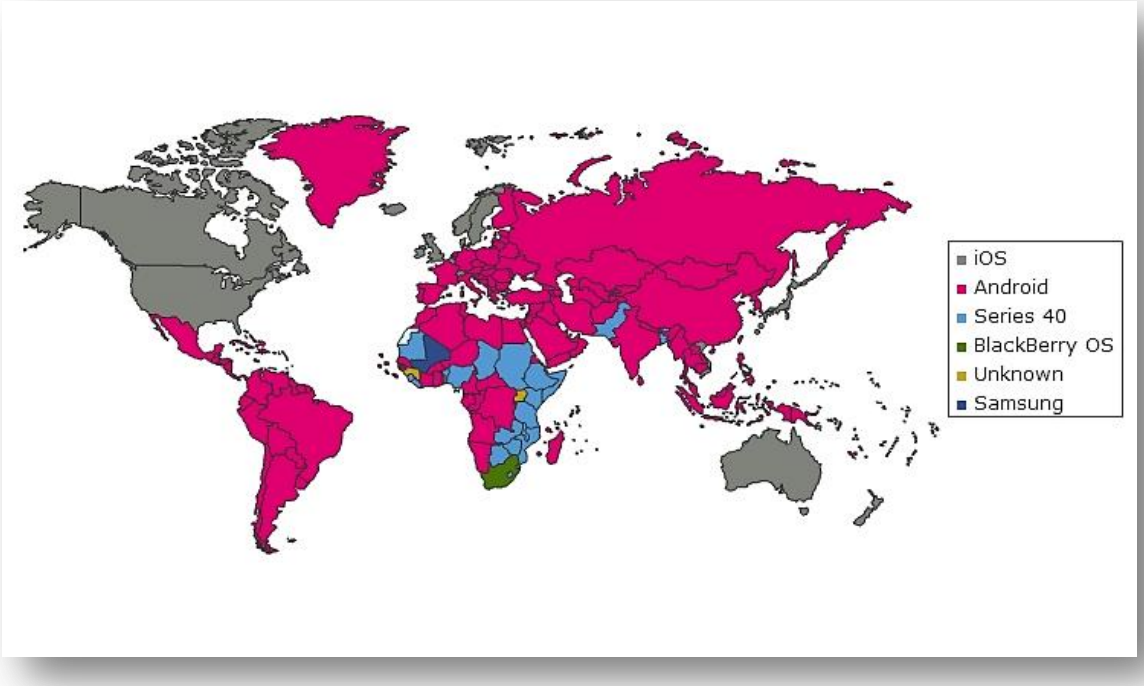


Imagen 3: SO Móviles más utilizados por países año 2014

Como se puede ver, en 2010 Android no reinaba en ninguno de los países. En 2012 comienza a coger importancia en algunos países de Sudamérica y de Europa, entre ellos España. En 2014 ya es el sistema operativo móvil más extendido en el Mundo con diferencia.

3.1.2.9 Estado actual del mercado de desarrollo de aplicaciones móviles

La empresa VisionMobile realiza desde hace tiempo su informe Developer Economics, y en la sexta edición, que estudia el primer trimestre de 2014, se analizan las tendencias del mercado en cuanto al interés de los desarrolladores por el segmento de la movilidad.

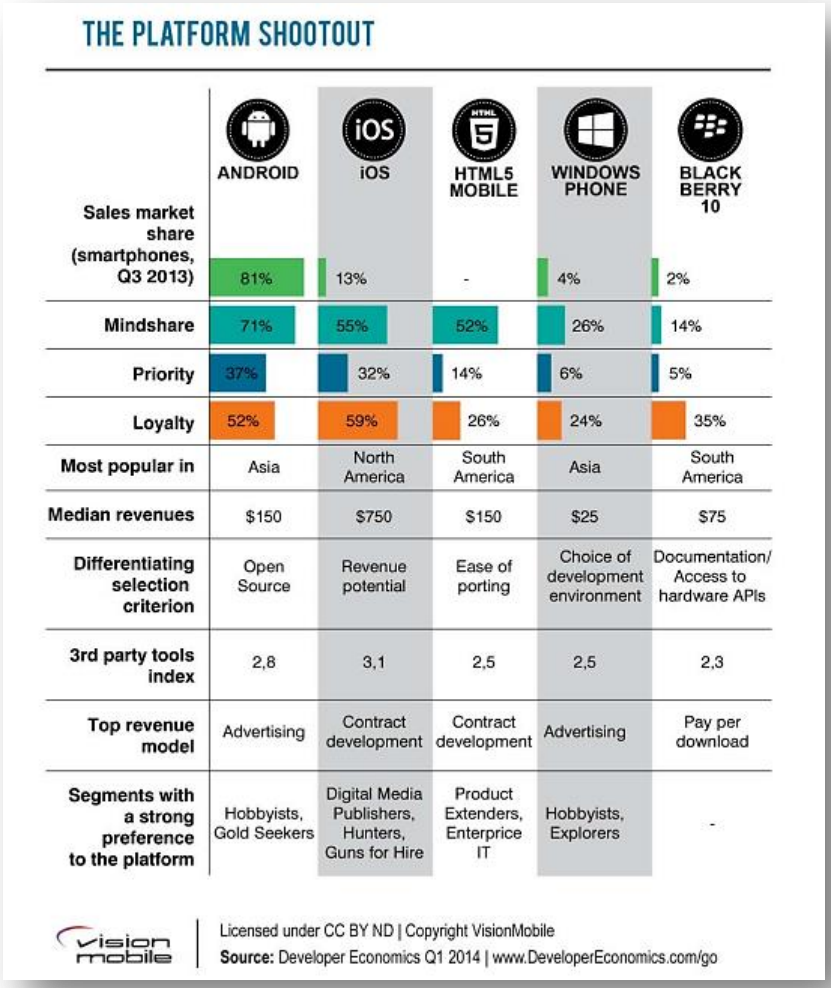
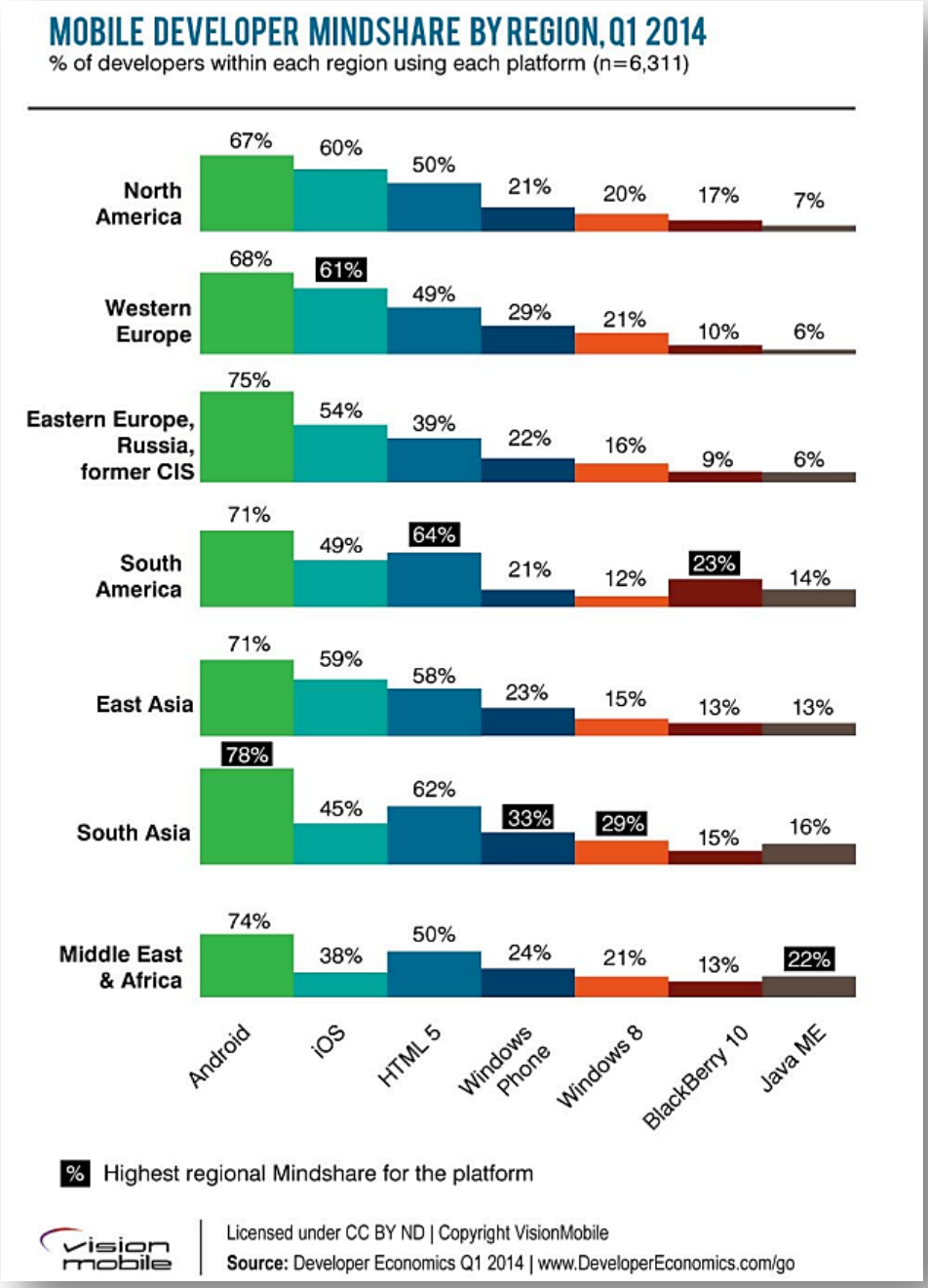


Imagen 4: Tabla general mercado aplicaciones

El estado del desarrollo de aplicaciones móviles parece bastante claro: Android e iOS se repartieron el 94% de las ventas de software en smartphones en el cuarto trimestre de 2013 según este estudio. De ese porcentaje el 71% se dedicó a Android, mientras que el 55% desarrolla en iOS. Como se puede comprobar, parte de los desarrolladores trabaja de forma paralela en ambas plataformas móviles.

También es interesante fijarse un poco en las tendencias que deja entrever el estudio; por ejemplo, iOS es la plataforma preferida en países desarrollados, y los que solo se dedican a dicha plataforma (el 59% de los desarrolladores encuestados) son más en porcentaje que los que solo se dedican a Android (el 52%).

Otra de las cosas importantes que deja a entrever el estudio es la irrupción de la plataforma HTML5 como posible competidora a corto plazo de Android e iOS en el mercado de desarrollo de aplicaciones. Este estándar abierto que es la base de propuestas como Firefox OS (y en menor medida, Tizen, Sailfish OS o Ubuntu) debe aún convencer a ese 69% de la comunidad de desarrolladores que siguen considerando tanto iOS como Android su plataforma de preferencia.



Gráfica 3: % de desarrolladores por regiones que usan cada plataforma

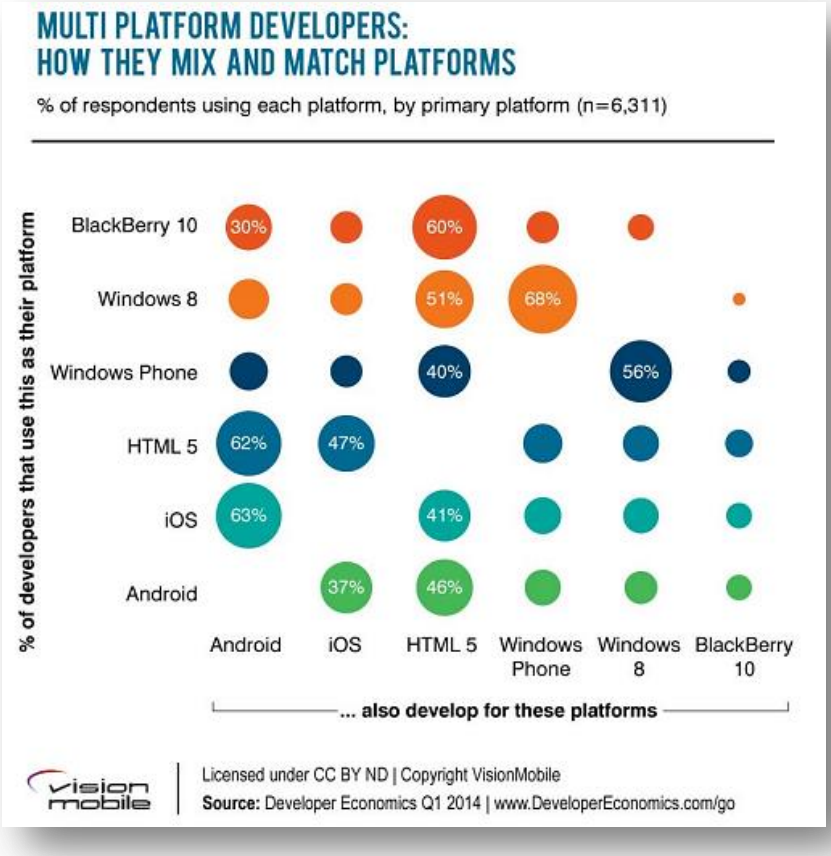
Como se puede ver en la gráfica anterior, de media el 52% de los desarrolladores afirma tenerlo como plataforma de desarrollo alternativa y contemplada en sus proyectos, pero no es actualmente la prioridad para ellos.

Recientemente una encuesta de Sencha decía que en torno al 60% de los desarrolladores móviles han migrado a HTML5, y este año el 75% de ellos, a su vez, apostarán aún más por este estándar, aunque no se puede hablar con total certeza sobre el tema ya que plataformas como BlackBerry son para ciertos desarrolladores más interesantes que HTML5 hoy en día.

Pero la cosa es diferente en cuanto a mercados emergentes, que son justo aquellos a los que Firefox OS va dirigido. El estudio revelaba que en Asia, Sudamérica, Oriente Medio y África, iOS es en realidad la tercera en prioridad por detrás de Android y de HTML5.

Para hacernos una idea, veremos algunos casos concretos. En Asia, Android reina con el 46% del mercado de los desarrolladores, por un 28% de iOS. En África la diferencia aumenta aún más: 47% para Android, 9% para iOS., pero el caso más llamativo es el de Sudamérica. En Sudamérica HTML5 atrae al 64% de los desarrolladores según los índices del Developer Economics en el primer trimestre de 2014, casi tan alto como el de Android (71%) y sensiblemente por encima de iOS (49%).

En el siguiente gráfico además se muestra el porcentaje de desarrolladores que tienen una plataforma como principal, pero que también desarrollan aplicaciones en otras, vemos como HTML5 es escogido como alternativa en todos los casos con mayor o menor éxito.



Gráfica 4: % de desarrolladores que usan una plataforma principal y desarrollan también en otras

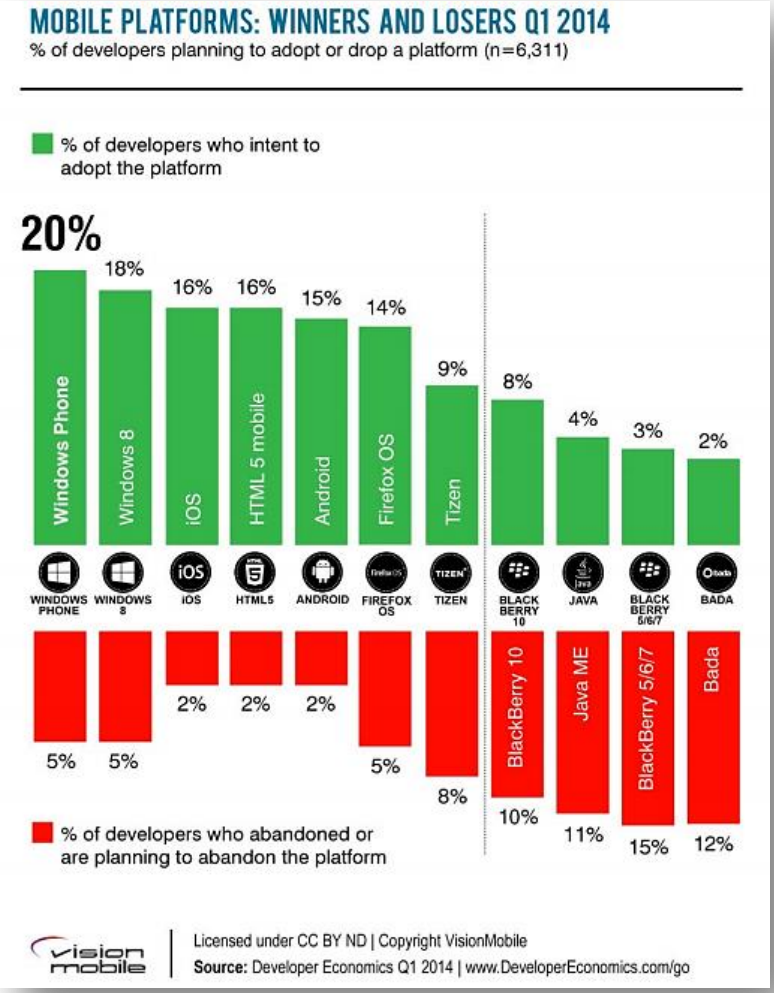
Se debe tener en cuenta que el uso de HTML5 es muy dispar, y que los desarrolladores no solo lo contemplan para aplicaciones nativas, sino también como parte de aplicaciones híbridas (que tienen también código nativo, pero muestran contenido web gracias a HTML5) e incluso como alternativa para, por ejemplo, mostrar la documentación de una aplicación. Por ello, mientras que el 37% de los desarrolladores

móviles utilizan HTML5 como una plataforma para desarrollar sitios web móviles o aplicaciones web, sólo un 15% utiliza este estándar como pilar fundamental de aplicaciones móviles de calidad.

Visto todo esto, las conclusiones del estudio son claras en cuanto a la situación actual de HTML5, aunque es una plataforma tremendamente atractiva, aún le quedan problemas importantes que resolver de cara al futuro, como:

- Carece de servicios de distribución y monetización en forma de una tienda de aplicaciones de gran escala.
- Problemas que han estado presentándose en aplicaciones móviles HTML5 para el navegador persisten, algunos ejemplos serían el déficit de rendimiento con respecto al código nativo, la falta de acceso a recursos hardware específicos a través de la API, la falta de herramientas maduras y también cierto nivel de fragmentación en diversos entornos de navegación.

Al margen de lo expuesto anteriormente, la realidad es que hoy por hoy Android e iOS marcan la pauta, y la seguirán marcando durante al menos los próximos tres años. Las predicciones podrían variar si como parece Android extiende su alcance y llega a conquistar la aún confusa Internet de las Cosas, así como los dispositivos wearable, además de la automoción, o campos como el entretenimiento y la educación.

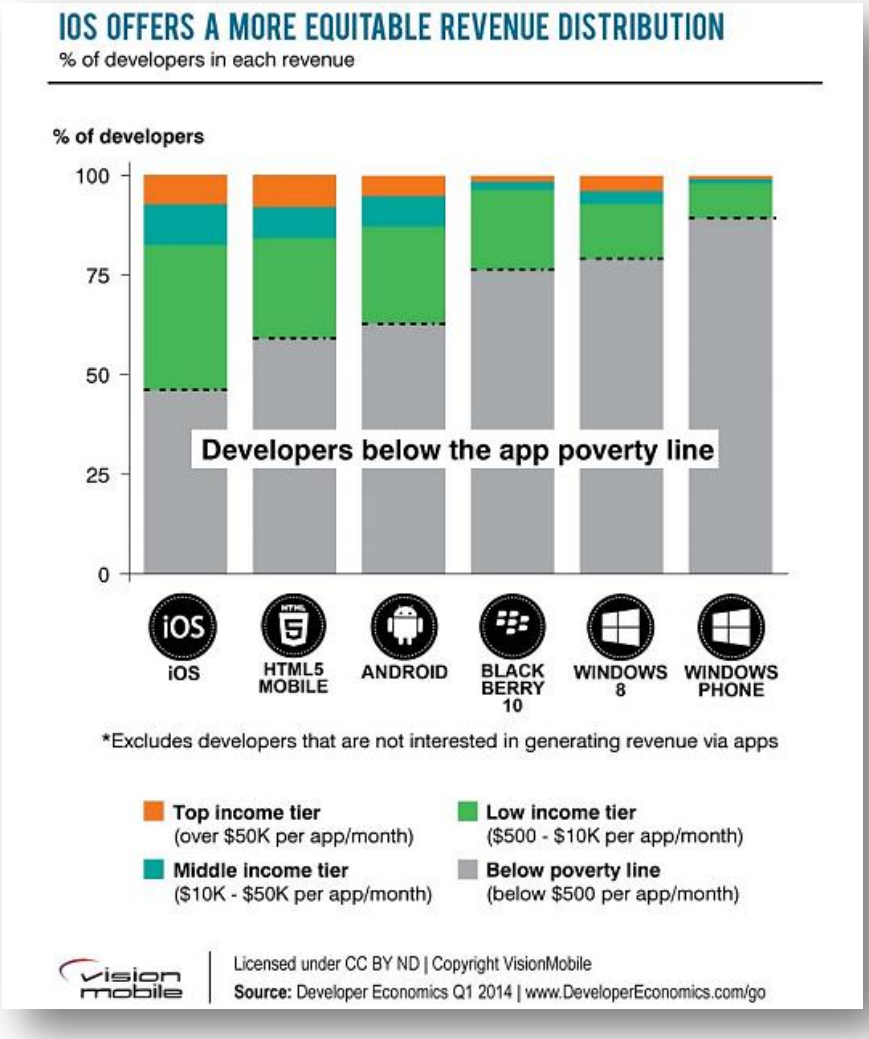


Gráfica 5: % de desarrolladores que acogen o abandonan la plataforma

Android es relevante en cuanto a cifras brutas, pero iOS sigue siendo la preferida por ingresos, algo de lo que siempre se ha presumido en Apple. En el informe se destaca por ejemplo el hecho de que la cuota de desarrolladores que generan "ingresos viables" es del 54% en iOS, mientras que esa cuota baja al 38% en Android.

El estudio no incluyen a desarrolladores que no monetizan sus aplicaciones móviles, algo que según el estudio es bastante más frecuente entre desarrolladores Android, que parecen implementar aplicaciones y juegos más por afición que por el dinero que puedan obtener a cambio.

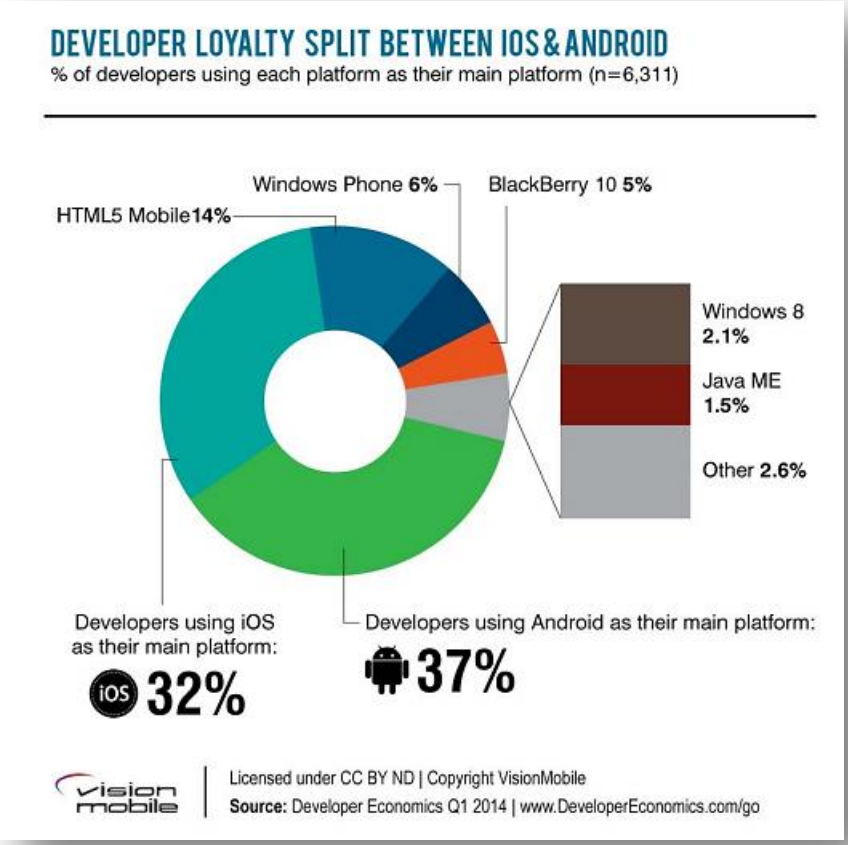
Esto se entiende muy bien en el siguiente gráfico:



Gráfica 6: % de desarrolladores de cada plataforma que obtiene "ingresos viables"

La gráfica es clara, excepto en la plataforma iOS, la mayoría de desarrolladores del resto de plataformas ganan menos de 500$ por aplicación cada mes.

Habiendo comparado y visto muchos pros y contras de las diferentes aplicaciones, el siguiente gráfico muestra cuantos desarrolladores se decantan por cada plataforma según el estudio.



Gráfica 7: % de desarrolladores que eligen cada plataforma como principal

Este gráfico no hace otra cosa sino reforzar lo dicho anteriormente, por el momento, Android e iOs marcan la pauta en cuanto a plataformas para el desarrollo de aplicaciones móviles.

Windows Phone y el resto de alternativas se encuentran en este momento a remolque.

Windows Phone crecería en interés para los desarrolladores según el estudio de VisionMobile, debido entre otras cosas a su notable aumento en cuota de mercado.

Según el estudio, una cuarta parte de los desarrolladores móviles tienen a Windows Phone como plataforma objetivo, pero Microsoft no termina de convencerles del todo.

El estudio también habla de alternativas surgidas hace escasos meses como son Firefox OS, Tizen, Sailfish OS y Ubuntu. De ellas la que parece tener mayores expectativas a medio plazo es Firefox OS. Un 7% de los desarrolladores muestran interés por la plataforma móvil impulsada por Mozilla, pero ese porcentaje aumenta hasta el 14% en desarrolladores que están interesados en adoptarla en el futuro. El estar basada en HTML5 favorece este interés.

3.2 Java

Es un lenguaje de programación de propósito general, concurrente, orientado a objetos y basado en clases originalmente desarrollado por James Gosling de Sun Microsystems que fue diseñado con la finalidad de tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en inglés como WORA, o "write once, run anywhere"), lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Java es, a partir de 2012, uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web, con unos 10 millones de usuarios reportados.

Esta portabilidad, que lo hizo tan importante, viene dada por la máquina virtual de Java o JVM (Java Virtual Machine). Al compilar un fichero fuente en Java no se crea un binario directamente ejecutable, sino un código intermedio llamado *bytecode* en ficheros cuya extensión es .class. Para cada hardware debe haber una JVM específica, ya sea un teléfono móvil, o un ordenador con Windows. Cada máquina virtual conoce el conjunto de instrucciones de la plataforma sobre la que está instalada, y puede traducir el bytecode, común para todas las máquinas, al código nativo que es capaz de entender el hardware en cuestión de dicha plataforma.

La revolución tecnológica sufrida en los últimos años propició que los responsables de Java ofrecieran soluciones personalizadas a cada ámbito tecnológico.

Sun decidió crear una edición distinta de Java según las necesidades del entorno y la tecnología utilizada:

- Java Enterprise Edition (Java EE), orientada al entorno empresarial.
- Java Standard Edition (Java SE), orientada al desarrollo con independencia de la plataforma.
- Java Micro Edition (Java ME), orientada a dispositivos con capacidades restringidas.
- Java Card, orientada a tarjetas inteligentes o smart cards.

Todas las ediciones comparten un conjunto más o menos amplio de las API básicas de Java, agrupadas principalmente en los paquetes java.lang y java.io.

Por lo tanto, Java Micro Edition es una versión muy específica del lenguaje Java, creado para desarrollar, instalar y ejecutar software escrito en Java en aparatos electrónicos de baja capacidad, como electrodomésticos, PDA, teléfonos móviles u ordenadores de bolsillo.

3.2.1 Arquitectura de Java Micro Edition

Como hemos dicho, JME es un subconjunto de la plataforma Java preparado para desarrollar aplicaciones para dispositivos móviles con escasos recursos, ya sean de memoria, pantalla, dispositivos de entrada, etc.

Por ello, la arquitectura de JME busca ser los más modular y escalable posible para conservar su funcionalidad y portabilidad entre la muy heterogénea gama de dispositivos objetivo.

Para ello se divide en capas según el tipo de dispositivo (características y funcionalidad) sobre el cual se va a implementar la plataforma.

A consecuencia de lo anterior, antes de desarrollar una aplicación en Java ME, es necesario tomar una serie de decisiones según el dispositivo a utilizar y las necesidades requeridas. No todos los desarrollos realizados en Java ME utilizan los mismos componentes. A grandes rasgos, una aplicación en Java ME se desarrolla a partir de una combinación de sistema operativo, máquina virtual, configuración y perfil.

El sistema operativo es desarrollado por el proveedor del dispositivo.

Sobre el sistema operativo se implementa la máquina virtual de Java, la cual es también desarrollada por el proveedor del dispositivo adaptándose a las particularidades de este y cumpliendo con los requisitos mínimos especificados por Sun Microsystems. Existen disponibles dos máquinas virtuales de Java ME con diferentes requisitos, cada una pensada para tipos distintos de pequeños dispositivos:

- KVM, o Kilobyte Virtual Machine, se corresponde con la máquina virtual más pequeña desarrollada por Sun. Se trata de una implementación de máquina virtual reducida y orientada a dispositivos de 16 o 32 bits con al menos 25 Mhz de velocidad y hasta 512 Kb de memoria total disponible.

- CVM, o Compact Virtual Machine, soporta las mismas características que la Máquina Virtual de Java SE. Está orientada a dispositivos electrónicos con procesadores de 32 bits de gama alta y en torno a 2Mb o más de memoria RAM.

La configuración se relaciona con el tipo de dispositivo y hace referencia al conjunto mínimo de características de la máquina virtual y de las librerías de clases que van a estar disponibles para un conjunto de dispositivos según sus características. En definitiva, una configuración consiste en un conjunto de clases básicas destinadas a conformar el corazón de la aplicación. En concreto, dentro de Java ME existen dos configuraciones:

- Connected Limited Device Configuration (CLDC) enfocada a dispositivos con restricciones de procesamiento y memoria.

- Connected Device Configuration (CDC) enfocada a dispositivos con más recursos.

El perfil se relaciona con el nivel de aplicación y define el conjunto mínimo de APIs disponibles para una determinada familia de dispositivos, de manera más específica y especializada que la proporcionada por la configuración.

Cabe mencionar, para entender mejor el funcionamiento, que la máquina virtual y los perfiles se implementan para una configuración específica y las aplicaciones se desarrollan posteriormente para un perfil determinado. Un dispositivo pertenece a una única configuración pero puede soportar diferentes perfiles.

Es decir, cada una de las dos configuraciones mencionadas requiere en realidad de una determinada máquina virtual. De esta forma, si es escoge la configuración CLDC, será necesario utiliza la máquina virtual denominada CVM; si, por el contrario, se decide utilizar la configuración CDC, la máquina virtual necesaria es la conocida como KVM.

Con la elección de perfiles, se da una situación similar. Existen unos perfiles que se utilizan sobre la configuración CDC (Foundation Profile, Personal Profile y RMI Profile) y otros que lo hacen sobre CLDC (PDA Profile y Mobile Information Device Profile, conocido como MIDP).

3.2.2 Fragmentación en Java ME y Android

La popularidad de la tecnología Java hizo que pronto todos los fabricantes y desarrolladores se interesaran en su versión para dispositivos móviles. La creciente demanda de estos aparatos impulsó el uso de Java ME para el desarrollo de todo tipo de aplicaciones y juegos.

A consecuencia de esta diversidad, en muchos casos las aplicaciones desarrolladas en Java ME ya no podían ser ejecutadas en cualquier parte como anunciaba su ya expuesta filosofía. Con frecuencia, cada aplicación debía ser adaptada según el dispositivo móvil al que estuviera destinada. Es decir, una misma aplicación requería varios desarrollos para poder llegar al mayor número de usuarios posible.

Esta diferenciación recibe el nombre de fragmentación y es un problema en el mundo del desarrollo de aplicaciones móviles, recibe el nombre de fragmentación. La fragmentación aumenta costes y tiempo de desarrollo de una aplicación, que necesita ser adaptada para poder llegar al máximo de usuarios y maximizar así su rentabilidad.

Esta fragmentación se dio en Java ME pero no en otras versiones de Java, debido a que esta se destina en gran medida a dispositivos móviles y estos, con sus características peculiares, propician este fenómeno. Entre ellas:

- Hardware muy heterogéneo entre distintos dispositivos.
- Capacidades de procesador y memoria muy distantes.
- Gran variedad de tipos de pantalla, con resoluciones y formas diversas.
- Sistemas operativos diferentes en función de cada fabricante

En resumen, las diferentes implementaciones de hardware, API y JVM de los fabricantes hace que desarrollar aplicaciones verdaderamente portables a Java ME sea prácticamente imposible, obligando que cada desarrollo de aplicación este siempre dirigida a un determinado grupo de dispositivos y a tener que traducir para poder cubrir mayores cuotas de mercado.

Al igual que ocurrió con Java ME, Android, casi desde su aparición, ha convivido con varias versiones diferentes del mismo SO funcionando al mismo tiempo en un parque completamente heterogéneo de smartphones, con distintas gamas, resoluciones, prestaciones, etc.

Los principales damnificados de esta fragmentación masiva y, por ende, del alcance, diseminación y densidad del sistema operativo son los desarrolladores, que tienen que probar y optimizar su trabajo para un número de dispositivos cada vez más grande, con el consiguiente tiempo que deben invertir para ello.

Una vez que somos conscientes de que la fragmentación conlleva sus inconvenientes y siendo fieles a la verdad, tendremos que asumir igualmente la existencia de ciertas "ventajas" tanto para desarrolladores como para usuarios. La existencia de un amplísimo catálogo de dispositivos que ejecutan el sistema operativo Android no sólo supone una mayor cuota de mercado, sino que la posibilidad de que terminales de menores prestaciones puedan estar equipados con Android – aunque se trate de versiones desactualizadas – proporciona un alcance global al sistema operativo del que carece iOS, por ejemplo.

Desde el punto de vista del usuario, la fragmentación le permite acceder a un dispositivo adaptado a sus necesidades tanto en precio como en tamaño, prestaciones, estética, en vez de tener que adaptarse a lo que pone a tu alcance un único fabricante.

4 JUSTIFICACIÓN SISTEMA OPERATIVO ESCOGIDO

4.1 Justificación

Como se ha dicho en puntos anteriores Android tiene la mayor cuota de mercado actualmente en el mundo, con lo que, en teoría el número de usuario que podrán disfrutar de nuestra aplicación será mayo en este sistema operativo que en otros, al menos por el momento.

Además tenemos que tener en cuenta sus características, como por ejemplo el hecho de que se trata de una plataforma de código abierto, lenguaje de programación Java, mayor facilidad y economía a la hora de hacer llegar las aplicaciones al usuario final, en principio menor coste de dispositivos que favorezcan el desarrollo de la aplicación en comparación con la otra gran plataforma actual, iOS.

Por estos motivos se ha decidido utilizar Android como plataforma para el desarrollo de este proyecto.

4.2 Android SO

Antes de empezar con el desarrollo de aplicaciones en Android es importante conocer cómo está estructurado este sistema operativo. A esto se le llama arquitectura y en el caso de Android está formada por varias capas que facilitan al desarrollador la creación de aplicaciones. Además, esta distribución permite acceder a las capas más bajas mediante el uso de librerías para que así el desarrollador no tenga que programar a bajo nivel las funcionalidades necesarias para que una aplicación haga uso de los componentes de hardware de los teléfonos.

4.2.1 Arquitectura Android

Cada una de las capas utiliza elementos de la capa inferior para realizar sus funciones, es por ello que a este tipo de arquitectura se le conoce también como pila. Para entenderlo mejor, nos podemos fijar en el siguiente diagrama:

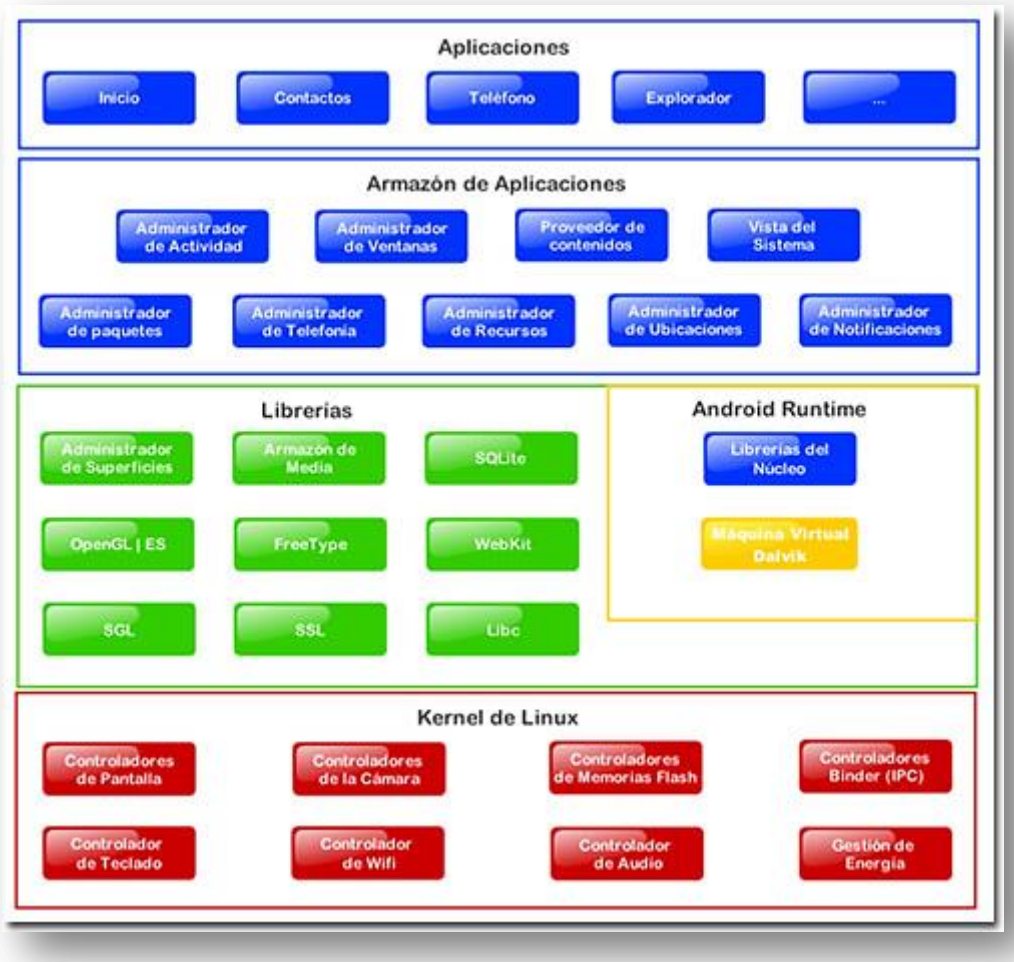


Imagen 5: Arquitectura Android

En general, la estructura del sistema operativo Android se compone de aplicaciones que se ejecutan en un framework de aplicaciones orientado a objetos sobre el núcleo de las bibliotecas (API) en una máquina virtual denominada Dalvik, con compilación en tiempo de ejecución. Las bibliotecas escritas en lenguaje C incluyen un administrador de interfaz gráfica (surface manager), un framework OpenCore, una base de datos relacional SQLite, una Interfaz de programación de API gráfica OpenGL ES 2.0 3D, un motor de renderizado WebKit, un motor gráfico SGL, SSL y una biblioteca estándar de C Bionic.

Explicamos ahora cada una de las capas que muestra la imagen un poco más en profundidad.

4.2.1.1 Kernel de Linux

El núcleo del sistema operativo Android está basado en el kernel de Linux en su versión 2.6, similar al que puede incluir cualquier distribución de Linux, como Ubuntu, solo que adaptado a las características del hardware en el que se ejecutará Android, es decir, para dispositivos móviles.

La elección de Linux 2.6 es debida principalmente a dos razones: la primera, su naturaleza de código abierto y libre se ajusta al tipo de distribución que persigue Android; la segunda es que este kernel de Linux incluye de por sí numerosos drivers, además de contemplar la gestión de memoria, gestión de procesos, módulos de seguridad, comunicación en red y otras muchas responsabilidades propias de un sistemas operativo.

El núcleo actúa como una capa de abstracción entre el hardware y el resto de las capas de la arquitectura. El desarrollador no accede directamente a esta capa, sino que debe utilizar las librerías disponibles en capas superiores. De esta forma también nos evitamos el hecho de quebrarnos la cabeza para conocer las características precisas de cada teléfono. Si necesitamos hacer uso de la cámara, el sistema operativo se encarga de utilizar la que incluya el teléfono, sea cual sea. Para cada elemento de hardware del teléfono existe un controlador (o driver) dentro del kernel que permite utilizarlo desde el software.

El kernel también se encarga de gestionar los diferentes recursos del teléfono (energía, memoria, etc.) y del sistema operativo en sí: procesos, elementos de comunicación, etc.

4.2.1.2 Librerías

La siguiente capa que se sitúa justo sobre el kernel la componen las bibliotecas nativas de Android, también llamadas librerías. Están escritas en C o C++ y compiladas para la arquitectura hardware específica del teléfono. Estas normalmente están hechas por el fabricante, quien también se encarga de instalarlas en el dispositivo antes de ponerlo a la venta. El objetivo de las librerías es proporcionar funcionalidad a las aplicaciones para tareas que se repiten con frecuencia, evitando tener que codificarlas cada vez y garantizando que se llevan a cabo de la forma "más eficiente".

Entre las librerías más importantes de este nivel, se pueden mencionar las siguientes:

- La librería libc incluye todas las cabeceras y funciones según el estándar del lenguaje C. Todas las demás librerías se definen en este lenguaje.
- La librería Surface Manager es la encargada de componer los diferentes elementos de navegación de pantalla. Gestiona también las ventanas pertenecientes a las distintas aplicaciones activas en cada momento.
- OpenGL/SL y SGL representan las librerías gráficas y, por tanto, sustentan la capacidad gráfica de Android. *OpenGL/SL* maneja gráficos en 3D y permite utilizar, en caso de que esté disponible en el propio dispositivo móvil, el hardware encargado de proporcionar gráficos 3D. Por otro lado, *SGL* proporciona gráficos en 2D, por lo que será la librería más habitualmente utilizada por la mayoría de las aplicaciones. Una característica importante de la capacidad gráfica de Android es que es posible desarrollar aplicaciones que combinen gráficos en 3D y 2D.

- La librería Media Libraries proporciona todos los códecs necesarios para el contenido multimedia soportado en Android (vídeo, audio, imágenes estáticas y animadas, etc.)

- FreeType, permite trabajar de forma rápida y sencilla con distintos tipos de fuentes.

4.2.1.3 Entorno de ejecución

Como podemos apreciar en la imagen, el entorno de ejecución de Android no se considera una capa en sí mismo, dado que también está formado por librerías. Aquí encontramos las librerías con las funcionalidades habituales de Java así como otras específicas de Android.

El componente principal del entorno de ejecución de Android es la máquina virtual Dalvik. Las aplicaciones se codifican en Java y son compiladas en un formato específico para que esta máquina virtual las ejecute. La ventaja de esto es que las aplicaciones se compilan una única vez y de esta forma estarán listas para distribuirse con la total garantía de que podrán ejecutarse en cualquier dispositivo Android que disponga de la versión mínima del sistema operativo que requiera la aplicación.

Cabe aclarar que Dalvik es una variación de la máquina virtual de Java, por lo que no es compatible con el bytecode Java. Java se usa únicamente como lenguaje de programación, y los ejecutables que se generan con el SDK de Android tienen la extensión .dex (Dalvik Executable) que es específico para Dalvik, y por ello no podemos correr aplicaciones Java en Android ni viceversa.

4.2.1.4 Framework de aplicaciones

La siguiente capa está formada por todas las clases y servicios que utilizan directamente las aplicaciones para realizar sus funciones. La mayoría de los componentes de esta capa son librerías Java que acceden a los recursos de las capas anteriores a través de la máquina virtual Dalvik. Siguiendo el diagrama encontramos:

1. Activity Manager. Se encarga de administrar la pila de actividades de nuestra aplicación así como su ciclo de vida.

2. Windows Manager. Se encarga de organizar lo que se mostrará en pantalla. Básicamente crea las superficies en la pantalla que posteriormente pasarán a ser ocupadas por las actividades.

3. Content Provider. Esta librería es muy interesante porque crea una capa que encapsula los datos que se compartirán entre aplicaciones para tener control sobre cómo se accede a la información.

4. Views. En Android, las vistas los elementos que nos ayudarán a construir las interfaces de usuario: botones, cuadros de texto, listas y hasta elementos más avanzados como un navegador web o un visor de Google Maps.

5. Notification Manager. Engloba los servicios para notificar al usuario cuando algo requiera su atención mostrando alertas en la barra de estado. Un dato importante es que esta biblioteca también permite jugar con sonidos, activar el vibrador o utilizar los LEDs del teléfono en caso de tenerlos.

6. Package Manager. Esta biblioteca permite obtener información sobre los paquetes instalados en el dispositivo Android, además de gestionar la instalación de nuevos paquetes. Con paquete nos referimos a la forma en que se distribuyen las aplicaciones Android, estos contienen el archivo .apk, que a su vez incluyen los archivos .dex con todos los recursos y archivos adicionales que necesite la aplicación, para facilitar su descarga e instalación.

7. Telephony Manager. Con esta librería podremos realizar llamadas o enviar y recibir SMS/MMS, aunque no permite reemplazar o eliminar la actividad que se muestra cuando una llamada está en curso.

8. Resource Manager. Con esta librería podremos gestionar todos los elementos que forman parte de la aplicación y que están fuera del código, es decir, cadenas de texto traducidas a diferentes idiomas, imágenes, sonidos o layouts. En un post relacionado a la estructura de un proyecto Android veremos esto más a fondo.

9. Location Manager. Permite determinar la posición geográfica del dispositivo Android mediante GPS o redes disponibles y trabajar con mapas.

10. Sensor Manager. Nos permite manipular los elementos de hardware del teléfono como el acelerómetro, giroscopio, sensor de luminosidad, sensor de campo magnético, brújula, sensor de presión, sensor de proximidad, sensor de temperatura, etc.

11. Cámara. Con esta librería podemos hacer uso de la(s) cámara(s) del dispositivo para tomar fotografías o para grabar vídeo.

12. Multimedia. Permiten reproducir y visualizar audio, vídeo e imágenes en el dispositivo.

4.2.1.5 Aplicaciones

En la última capa se incluyen todas las aplicaciones del dispositivo, tanto las que tienen interfaz de usuario como las que no, las nativas (programadas en C o C++) y las administradas (programadas en Java), las que vienen preinstaladas en el dispositivo y aquellas que el usuario ha instalado.

En esta capa encontramos también la aplicación principal del sistema: Inicio (Home) o lanzador (launcher), porque es la que permite ejecutar otras aplicaciones mediante una lista y mostrando diferentes escritorios donde se pueden colocar accesos directos a aplicaciones o incluso widgets, que son también aplicaciones de esta capa.

4.2.2 Fundamentos de la creación de aplicaciones Android

Hemos comentado que las aplicaciones Android deben estar escritas en el lenguaje de programación Java. El SDK de Android, o mejor dicho las herramientas del SDK de Android compilan el código, junto con los datos y archivos de recursos en un paquete de Android, en un archivo comprimido con extensión .apk. Todo el código queda reducido a un solo archivo .apk que se considera en sí mismo la aplicación, y es este archivo el que se utiliza para la instalación de aplicaciones en el dispositivo.

Ahora nos centramos en explicar cómo la aplicación interactúa con el sistema operativo Android

Una vez instalada en el dispositivo la aplicación corre en su propio recinto o entorno limitado de seguridad:

- El sistema operativo Android es un sistema multi-usuario de Linux en el que cada aplicación es un usuario diferente.
- Por defecto, el sistema asigna a cada aplicación de una única ID de usuario de Linux (el ID es utilizado únicamente por el sistema y desconocido para la aplicación). El sistema establece permisos para todos los archivos en una aplicación para que sólo el ID de usuario asignado a esa aplicación puede acceder a ellos.
- Cada proceso tiene su propia máquina virtual, por lo que el código de una aplicación se ejecuta de forma aislada de otras aplicaciones.
- Por defecto, cada aplicación se ejecuta en su propio proceso de Linux. Android lo inicia cuando alguno de los componentes de la aplicación se ejecuta, a continuación, cierra el proceso cuando ya no se necesita o cuando el sistema debe recuperar la memoria para otras aplicaciones, este caso en concreto lo explicaremos más en detalle en el ciclo de vida de una actividad, ya que es un rasgo distintivo de Android.

Con esto, el SO Android implementa el principio de privilegios mínimos, que consiste en que cada aplicación, por defecto, sólo puede acceder a los componentes requeridos para hacer su trabajo. Esto genera un entorno muy seguro en el que la aplicación en cuestión no puede acceder a partes del sistema para las cuales no se le ha otorgado permiso.

De cualquier modo, existen manera para que una aplicación pueda compartir datos con otras aplicaciones, o también que, una aplicación pueda acceder a los servicios del sistema:

- En el caso en el que dos aplicaciones compartan el mismo ID de usuario de Linux, son capaces de acceder cada una a los archivos de la otra. Para ahorrar recursos del sistema, en este caso en el que dos aplicaciones tienen el mismo ID de usuario también se pueden organizar

para ejecutar en el mismo proceso de Linux y compartir la misma máquina virtual (para ello las solicitudes deben estar firmadas con el mismo certificado).

- Una aplicación puede solicitar permiso para acceder a los datos del dispositivo, tales como los contactos del usuario, mensajes SMS, el almacenamiento externo (tarjeta SD), cámara, Bluetooth, y mucho más. Todos los permisos de la aplicación debe ser autorizados por el usuario durante la instalación.

Vamos a centrarnos a continuación en los fundamentos para el desarrollo de aplicaciones en Android.

4.2.2.1 Componentes de las Aplicaciones Android

Los principales componentes de toda aplicación Android son:

- Actividades (Activities): Cada pantalla de una aplicación. Utilizan Vistas (Views) como componentes que muestran información y responden a las acciones del usuario

- Servicios (Services): Componentes de la aplicación que se ejecutan deforma invisible, actualizando los datos y las Actividades, y disparando Notificaciones. Realizan el procesamiento normal de la aplicación que debe continuar incluso cuando las Actividades de la aplicación no están visibles.

- Proveedores de Contenidos (Content Providers): Almacenes de datos compartidos. Gestionan las Bases de Datos para las aplicaciones.

- Intenciones (Intents): Mecanismo que permite el paso de mensajes destinados a ciertas Actividades o Servicios, o a todo el sistema (Intenciones de broadcast). Exponen la intención de que se haga algo. El sistema determinará el destinatario que lo efectuará.

- Receptores de Broadcast (Broadcast Receivers): Los crean las aplicaciones como consumidores de las Intenciones de broadcast que cumplan ciertos criterios.

- Notificaciones (Notifications): Mecanismo que permite a las aplicaciones señalar algo a los usuarios sin interrumpir la Actividad en primer plano.

4.2.2.2 El Manifiesto de una Aplicación Android (Android Manifest)

Toda aplicación desarrollada en Android incluye un fichero de Manifiesto, el AndroidManifest.xml. Este fichero define la estructura de la aplicación y sus componentes. Incluye un nodo raíz y un nodo para cada uno de sus tipos de

componentes. A través de filtros de intenciones determina como interactuará la aplicación en cuestión con otras aplicaciones.

Algunos de los nodos más importantes serán:

- Nodo raíz **manifest.** Incluye el nombre del paquete de la aplicación.
- Nodo **application.** Indica metadatos de la aplicación (título, icono, tema, etc.) y contiene los nodos de actividades, servicios, proveedores de contenido y receptores de broadcast.
- Nodo **uses-permission.** Declara loes permisos que la aplicación necesita para operar. Estos permisos serán presentados al usuario durante la instalación para que los acepte o deniegue, estos permisos serán necesarios para infinitud de servicios nativos de Android, como enviar SMS, hacer llamadas, uso de servidor, etc.
- Nodo **permission.** Define un permiso que se requiere para que otras aplicaciones puedan acceder a partes restringidas de la aplicación. Las otras aplicaciones necesitarán poner un uses-permission en su Manifiesto para utilizar este permiso.
- Nodo **instrumentation.** Permite definir test de ejecución para las Actividades y Servicios.

4.2.2.3 Creación y destrucción de Aplicaciones y Actividades (Ciclo de vida)

Las aplicaciones Android no son iguales a las aplicaciones en los sistemas operativos tradicionales, en Android sólo hay una aplicación en primer plano que normalmente estará ocupando toda la pantalla. Las aplicaciones estarán formadas por Actividades (pantallas).

Al arrancar una nueva aplicación, pasa a primer plano situando una Actividad encima de la que hubiera, formándose así una pila de actividades. En el momento en el que el usuario presiona el botón “back”, se cierra la actividad en primer plano y recupera la de la cima de la pila.

Las aplicaciones Android no tienen control ninguno sobre su propio ciclo de vida, esto implica que deben estar pendientes de posibles cambios en su estado y reaccionar como corresponda. En particular deben estar preparadas para su terminación o destrucción en cualquier momento.

En general, cada aplicación se ejecuta en un proceso que ejecuta una instancia de Dalvik. El runtime de Android gestiona el proceso de cada aplicación, y por extensión de cada actividad que contenga.

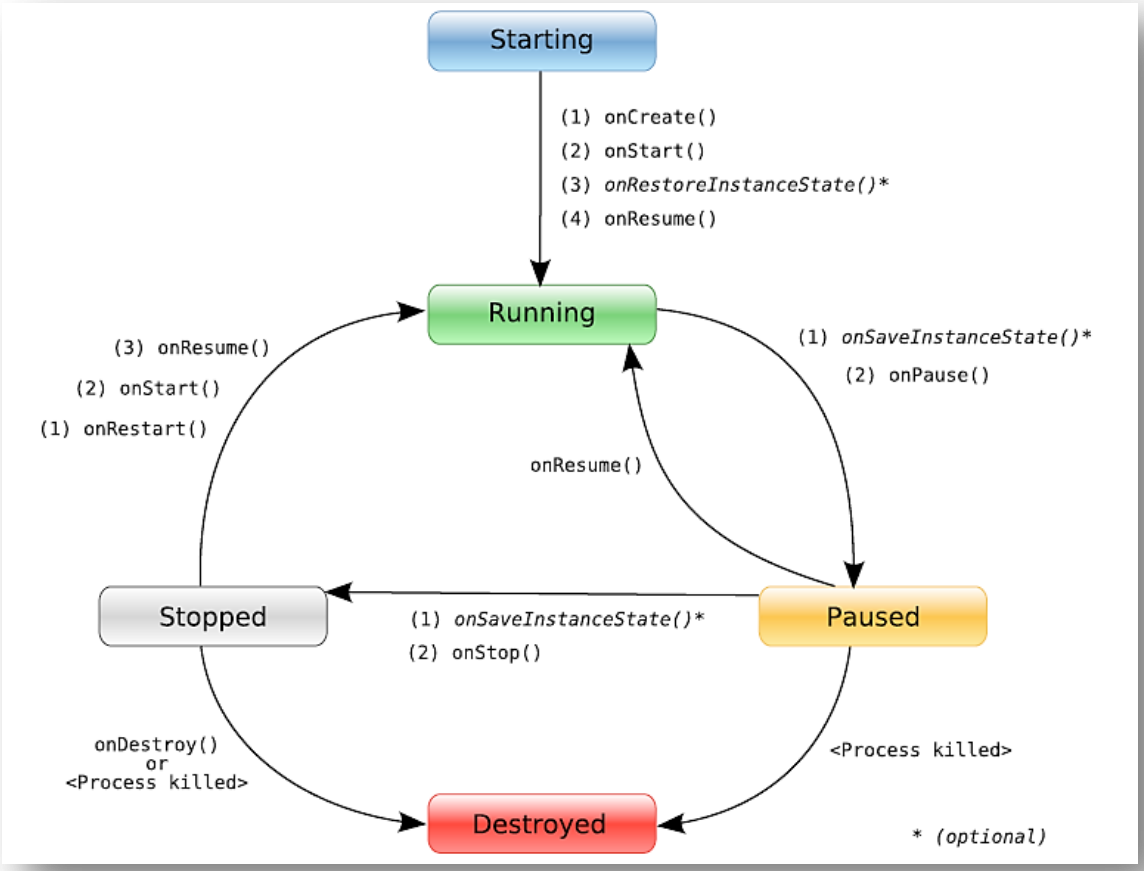


Imagen 6: Estados de una actividad

Como se puede ver en la imagen, la actividad se puede encontrar en los siguientes estados:

- Activo (Running). La actividad está encima de la pila, es visible, tiene el foco (recibe la entrada del usuario). Cuando otra actividad pase a estar activa, esta pasará a estar pausada.
- Pausado (Paused). La actividad es visible pero no tiene el foco. Se alcanza este estado cuando pasa a activa otra actividad transparente o que no ocupa toda la pantalla. Cuando una actividad es tapada por completo pasa a estar parada.
- Parado (Stopped). Cuando la actividad no es visible. Permanece en memoria reteniendo su estado. Cuando una actividad entra en parada puede ser bueno que salve todos sus datos y el estado de la interfaz de usuario.
- Destruido (Destroyed). Cuando la actividad termina, o es matada por el runtime de Android. Sale de la pila de actividades. Necesita ser reiniciada para volver a estar activa.

Y del mismo modo existirán una serie de métodos de transición entre unos estados y otros:

- onCreate (Bundle). Se invoca cuando la Actividad se arranca por primera vez. Se utiliza para tareas de inicialización a realizar una sola vez, como crear la interfaz de usuario de la Actividad. Su parámetro es null o información de estado guardada previamente por onSaveInstanceState ().
- onStart (). Se invoca cuando la Actividad va a ser mostrada al usuario.
- onResume (). Se invoca cuando la Actividad va a empezar a interactuar con el usuario.
- onPause (). Se invoca cuando la actividad va a pasar al fondo porque otra actividad ha sido lanzada para ponerse delante. Se utiliza para guardar el estado persistente de la Actividad
- onStop (). Se invoca cuando la actividad va a dejar de ser visible y no se necesitará durante un tiempo. Si hay escasez de recursos en el sistema, este método podría no llegar a ser invocado y la Actividad ser destruida directamente.
- onRestart (). Se invoca cuando la Actividad va a salir del estado de parada para volver a estar activa.
- onDestroy (). Se invoca cuando la Actividad va a ser destruida. Si hay escasez de recursos en el sistema, este método podría no llegar a ser invocado y la Actividad ser destruida directamente.
- onSaveInstanceState(Bundle). Se invoca para permitir a la actividad guardar su estado, por ejemplo la posición del cursor en una caja de texto. Normalmente no necesita ser redefinido porque la implementación de la clase activity ya guarda todo el estado de todos los componentes de la Interfaz de Usuario.
- onRestoreInstanceState(Bundle). Se invoca para recuperar el estado guardado por onSaveInstanceState(). Normalmente no necesita ser redefinido porque la implementación de la clase activity ya recupera todo el estado de todos los componentes de la Interfaz de Usuario.

Como se ve en la imagen, el proceso que mantiene a la actividad en cuestión puede ser eliminado cuando se encuentra en onPause() o en onStop(), es decir, cuando no tiene el foco de la aplicación. Android nunca elimina procesos con los que el usuario está interactuando en ese momento. Una vez se elimina el proceso, el usuario desconoce dicha situación y puede incluso volver atrás y querer usarlo de nuevo. Entonces el proceso se restaura gracias a una copia y vuelve a estar activo como si no hubiera sido eliminado. Además, la actividad puede haber estado en segundo plano, invisible, y entonces es despertada pasando por el estado onRestart().

Pero, ¿qué ocurre en realidad cuando no existen recursos suficientes? Obviamente, los recursos son siempre limitados, más aún cuando se está hablando de dispositivos móviles. En el momento en el que Android detecta que no hay los recursos necesarios para poder lanzar una nueva aplicación, analiza los procesos existentes en ese momento y elimina los procesos que sean menos prioritarios para poder liberar sus recursos.

Cuando el usuario regresa a una actividad que está dormida, el sistema simplemente la despierta. En este caso, no es necesario recuperar el estado guardado porque el proceso todavía existe y mantiene el mismo estado. Sin embargo, cuando el usuario quiere regresar a una aplicación cuyo proceso ya no existe porque se necesitaba liberar sus recursos, Android lo crea de nuevo y utiliza el estado previamente guardado para poder restaurar una copia fresca del mismo. El usuario no percibe esta situación ni conoce si el proceso ha sido eliminado o está dormido.

4.2.2.4 Recursos de las aplicaciones Android

Una aplicación para Android se compone de algo más que código que requiere de recursos que están separados del código fuente, como imágenes, archivos de audio, y todo lo relativo a la presentación visual de la aplicación. Por ejemplo, se deben definir las animaciones, menús, estilos, colores y el diseño de interfaces de usuario de la actividad con archivos XML.

Para todos estos recursos Android proporciona un identificador único entero dentro de la aplicación, que puede utilizarse para hacer referencia al recurso en el código de aplicación o de otros recursos definidos en XML.

Hay distintos tipos de recursos, que se definen en ficheros XML alojados en una cierta subcarpeta de res:

- Valores simples (carpeta values). Strings, colores y dimensiones. Cada uno de ellos se definen como un elemento. Cada fichero XML contiene la definición de uno o más de estos elementos. Todos estos recursos se identifican con el valor de su atributo name.

- Recursos dibujables (carpeta drawable). Ficheros con imágenes, incluyendo el icono de la aplicación. Son usados para ficheros de bitmaps o de imágenes "estirables" (.9.png). Estos recursos se identifican con su nombre de fichero, y los recuadros de color con el valor de su atributo name.

- Animaciones (carpeta anim). Usados para animaciones sencillas sobre uno o varios gráficos: rotaciones. Fading, movimiento, etc. Cada animación se define en un fichero xML. Se identifican con su nombre de fichero.

- Menús (carpeta menu). Existen tres tipos de menús: de opciones, contextual y submenú. El menú de opciones y el menú contextual se

identifican con su nombre de fichero y el submenú con el valor de sui atributo id.

- Diseños (carpeta layout). Cada layout se define en un fichero XML. Dentor del layout se definen los elementos que lo componen, como puedan ser los Views o los ViewGroups. Se identificará por su nombre de fichero y los elementos del layout se podrán identificar con su atributo id.

- Estilos (carpeta values). Un estilo es uno o más atributos que se aplican a un elemento. El tema se define como uno o más atributos que se aplican a todo lo que hay en pantalla, este se asigna como atributo a una actividad en el Manifiesto. El estilo se referencia con el valor de su atributo name.

- Varios (carpeta xml). Usado para recursos con características especiales.

4.2.2.5 Procesos y prioridades

Tal y como se ha explicado, cada aplicación Android se ejecuta en su propio proceso. Este proceso se crea cada vez que una aplicación necesita ejecutar parte de su código, y seguirá existiendo hasta que la aplicación finalice o hasta que el sistema necesite utilizar parte de sus recursos para otra aplicación considerada prioritaria.

El orden en que los procesos se van matando para liberar recursos lo determina la jerarquía que Android establece evaluando la case de componentes que están ejecutándose y el estado de los mismos, en orden de importancia serán:

1. Prioridad critica:
    - *Procesos activos o en primer plano:* Los que están interactuando con el usuario, hay muy pocos y se les mata solo como último recurso.
2. Prioridad alta:
    - *Procesos visibles:* Visibles, pero no activos, por ejemplo tapados parcialmente por otra actividad, se les destruirá solo en situaciones extremas.
    - *Procesos de servicio arrancados:* Procesamiento que debe continuar aunque no tenga interfaz visible. Se les matara solo en casos extremos.
3. Prioridad baja:
    - *Procesos de fondo o en segundo plano:* Alojan actividades que no caen en las categorías anteriores. Hay muchos de estos procesos. Se les mata cuando se necesitan recursos para el resto de procesos.
    - *Procesos vacíos:* Android mantiene en memoria las aplicaciones que han terminado, por si son relanzadas. Se destruirán estos procesos rutinariamente.

Esto se puede ver en la siguiente imagen:

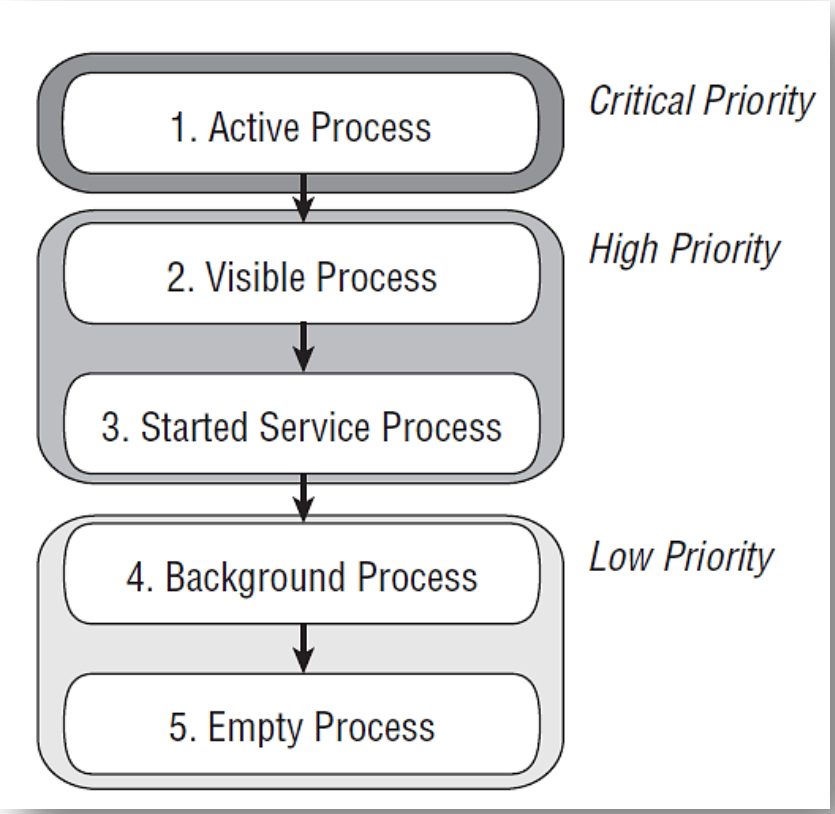


Imagen 7: Jerarquía de procesos en Android

4.3 Entorno de Desarrollo Integrado (IDE)

Un entorno de desarrollo integrado, llamado también IDE (siglas en inglés de Integrated Development Environment), es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien puede utilizarse para varios.

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación; es decir, que consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes.

4.3.1 Justificación IDE escogido

El primer IDE que se ha venido utilizando desde la aparición de Android para el desarrollo de sus aplicaciones ha sido Eclipse. Google proporciona de forma gratuita el SDK o Kit de Desarrollo de Software, y los plug-in para el Entorno de Desarrollo Integrado (IDE) Eclipse. Pero en Junio de 2013 Google sacó a la luz su propio IDE llamado Android Studio.

Ambos IDEs proporcionan un marco de desarrollo amigable que nos asiste en todas las etapas de la creación de una aplicación para Android, desde su programación en código, pasando por su emulación mediante un emulador que simula el comportamiento de la aplicación en un dispositivo real, hasta la firma digital de la aplicación para su posterior difusión al mundo.

En el momento que se comienza este proyecto (Octubre de 2013), nos decantamos por Android Studio por diversas razones:

- Es el futuro
- En breve será lo que más y tal vez lo único que el equipo de Android recomiende, y puede que para lo único para lo que Google lance plug-in, etc.
- Está basado en IntelliJ IDEA, uno de los IDE para java de primer nivel (entre los mejores, con Eclipse, netbeans, y Oracle JDeveloper), y es de JetBrains, cuyo PHPStorm está considerado como uno de los mejores para programar PHP
- Su particularidad más reseñable podría ser su nueva forma de construir los apk. Más serio, más versátil, más potente, más actual, y más parecido a un proyecto en java. Esto es debido a que utiliza Gradle. Las ventajas de esto son claras:
    - Facilita reusar código y recursos
    - Facilita configurar, extender y personalizar el proceso.
    - Facilita la distribución del código y por tanto trabajar en equipos.
    - Gestiona las dependencias de una forma cómoda y potente (está basado en Maven).
    - Nos permite compilar desde línea de comandos, lo cual es una gran ventaja en una máquina en la que no tenemos todo el entorno montado.
    - Facilita mucho crear distintas versiones de la aplicación.

Los pasos para tener operativo el IDE con su SDK están claramente explicados en la página web de Android www.android.com en el apartado para desarrolladores (Developers), concretamente para el caso de Android Studio en el siguiente enlace: http://developer.android.com/sdk/installing/studio.html, con lo que no entraremos en el proceso de instalación en esta memoria.

5 APLICACIÓN SPS

5.1 Introducción

Ya hemos hablado anteriormente de la fragmentación en Android, a raíz de ese problema surgen las siguientes cuestiones a tener en cuenta a la hora de crear nuestra aplicación para Android, las más importantes serán:

5.1.1 Tamaño de la pantalla y densidad de pixeles.

Uno de los rasgos más importantes de Android es su implantación en dispositivos con gran variedad de tamaño y densidad de pixeles. Con el fin de clasificar los dispositivos, Android define dos características de cada dispositivo: Tamaño de la pantalla (las dimensiones físicas de la pantalla) y la densidad de la pantalla (la densidad física de los píxeles en la pantalla, o dpi-puntos por pulgada).

Los tamaños de pantalla son: grande pequeño, normal, grande y extra grande. Las densidades de la pantalla son: baja densidad, de densidad media, de alta densidad y alta densidad extra.

Por defecto, la aplicación es compatible con todos los tamaños de pantalla y densidades, ya que el sistema Android hace los ajustes necesarios al diseño de interfaz de usuario y los recursos de imagen.

5.1.2 Configuraciones de entrada

Muchos dispositivos proporcionan un tipo diferente de mecanismo de entrada de usuario, tales como un teclado de hardware, una bola de seguimiento, o una almohadilla de cinco vías de navegación. Si la aplicación requiere un determinado tipo de hardware de entrada, entonces se debe declarar en el archivo manifiesto (AndroidManifest.xml).

5.1.3 Características del dispositivo

Hay muchos equipos y componentes de software que pueden o no pueden existir en un determinado dispositivo Android, tales como una cámara, un sensor de luz, bluetooth, una cierta versión de OpenGL, etc. Nunca se debe asumir que una determinada característica está disponible en todos los dispositivos con Android, pues puede que nuestra aplicación exija una característica no necesaria al dispositivo y por consiguiente sean incompatibles.

5.1.4 Versión de la plataforma

Sin duda uno de los aspectos más importantes a tener en cuenta por la fragmentación sufrida por Android en cuanto a su versión de plataforma. En nuestro caso, existen multitud de dispositivos con Android que a menudo montan diferentes versiones de Android, desde Android 1.6 hasta la última versión Android 4.4. El principal

problema reside en que cada versión sucesiva a menudo incluye APIs adicionales no disponible en anteriores versiones.

Este problema podría estar resuelto realizando la aplicación en la versión de Android más baja. No es la solución más adecuada debido a que a medida que se lanzan al mercado nuevas versiones se ofrecen más prestaciones y/o mejoras, y no podemos quedaron estancados en la versión más baja si queremos que nuestra aplicación tenga un nivel alto de competitividad en el mercado actual.

Por lo tanto este problema será de optimización. Queremos llegar al mayor número de usuarios y utilizando la versión más alta posible para no perder en prestaciones. Para ello nos basaremos en la información que nos ofrece la página web oficial de Android. Los datos actuales son los siguientes:

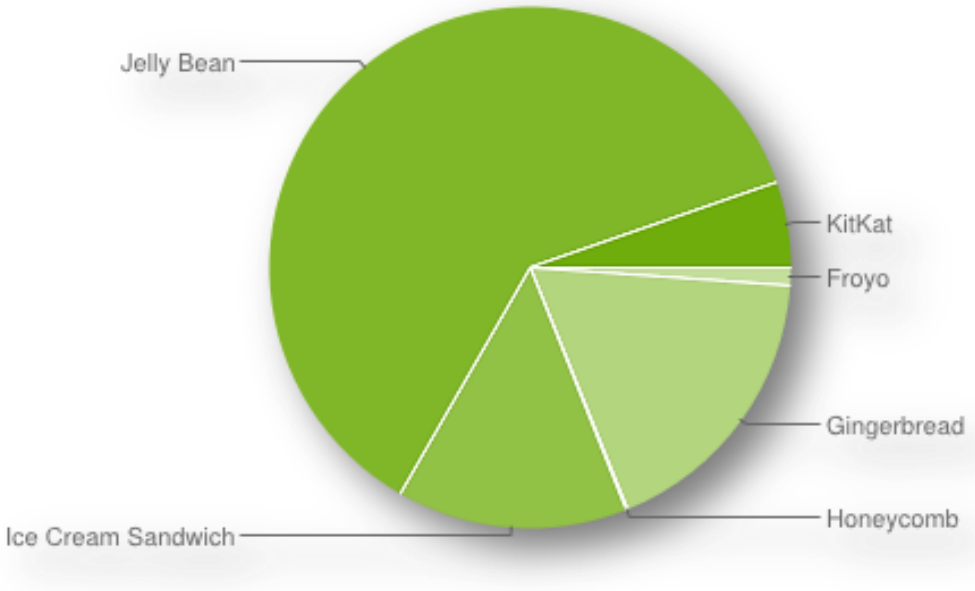
Version	Codename	API	Distribution
2.2	Froyo	8	1.1%
2.3.3 - 2.3.7	Gingerbread	10	17.8%
3.2	Honeycomb	13	0.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	14.3%
4.1.x	Jelly Bean	16	34.4%
4.2.x		17	18.1%
4.3		18	8.9%
4.4	KitKat	19	5.3%

Tabla 1: Distribución de versiones de plataforma

Estos datos, según informa la propia página web, han sido recogidos en un periodo de tiempo de 7 días, acabando el mismo el 1 de Abril de 2014. También menciona que versiones con una distribución menor del 0.1% no se muestran.

La página web también nos indica que como estos datos han sido recogidos de la nueva tienda online de Google (Google Play Store app), la cual solo contiene Android 2.2, y superiores, los dispositivos que usan versiones anteriores no están incluidos. Sin embargo, dicen, en Agosto de 2013, las versiones anteriores a Android 2.2 las montaban solamente en torno al 1% de dispositivos que se comprobaron en los servidores de Google, no en la Google Play Store.

El diagrama de sectores formado con los datos de la tabla anterior que nos ofrece la web de Android es el siguiente:



Gráfica 8: Distribución de versiones de plataformas

Como vemos, si queremos que actualmente nuestra aplicación llegue a casi el 100% de los dispositivos actualmente en el mercado debemos desarrollar la aplicación para un Android 2.3 (Gingerbread), API 10.

En este punto debemos preguntarnos para qué dispositivos vamos a desarrollar nuestra aplicación, para tener en cuenta su tamaño de pantalla, resolución, etc. Como no queremos limitarnos a un único tipo de dispositivo, probaremos la aplicación y la desarrollaremos para smartphones y tablets, probaremos la aplicación en dispositivos reales de ambos tipos y con diferentes versiones de Android.

5.2 Descripción de la aplicación

La aplicación SPS ha sido desarrollada para el cálculo de estructuras, el motor de cálculo de la misma se fundamenta en el Método Directo de Rigidez. Las estructuras resolubles con esta aplicación serán todo tipo de estructuras planas o 2D formadas por barras. La modelización en los extremos de barras es rígida. Las solicitaciones que permite introducir la aplicación son: peso propio, cargas puntuales nodales, cargas puntuales en barra, cargas distribuidas en barra, momentos en barra y cargas térmicas. Las vinculaciones exteriores que permite la aplicación son apoyos empotrados, fijos, móviles, elásticos y también se pueden introducir movimientos prescritos en los nodos.

5.2.1 Estructura de la aplicación SPS

Para el desarrollo de la aplicación SPS se ha decidido dividirla en diferentes partes según la función que desarrollan dentro de la propia aplicación, así la estructura de la misma sigue el siguiente esquema:

Gráfica 9: Esquema estructura aplicación SPS

Vemos brevemente la función que desempeñan cada una de ellas dentro de la aplicación:

- Activities. Son cada una de las pantallas que componen la aplicación, usadas tanto para recogida de datos e información que deberá introducir el usuario como para mostrar los resultados. Comúnmente se conocen como diálogos, son pantallas de interacción con el usuario.

- BaseDatos. Este paquete contiene toda la información sobre perfiles de acero que ha sido guardad en la base de datos. De esta forma el usuario a la hora de crear una barra podrá elegir uno de los perfiles predeterminados que existen en la base de datos o podrá crear un perfil personalizado y guardarlo para poder utilizarlo en barras posteriores.

- Datos. Uno de los paquetes fundamentales de la aplicación. Es el paquete que contiene el modelo específico para el almacenamiento y persistencia de la información. En él se guarda la información que el usuario va introduciendo al construir la estructura, se guardan tanto los elementos que componen la estructura (nodos y barras), como todas las cargas introducidas, condiciones de contorno, características geométricas, etc.

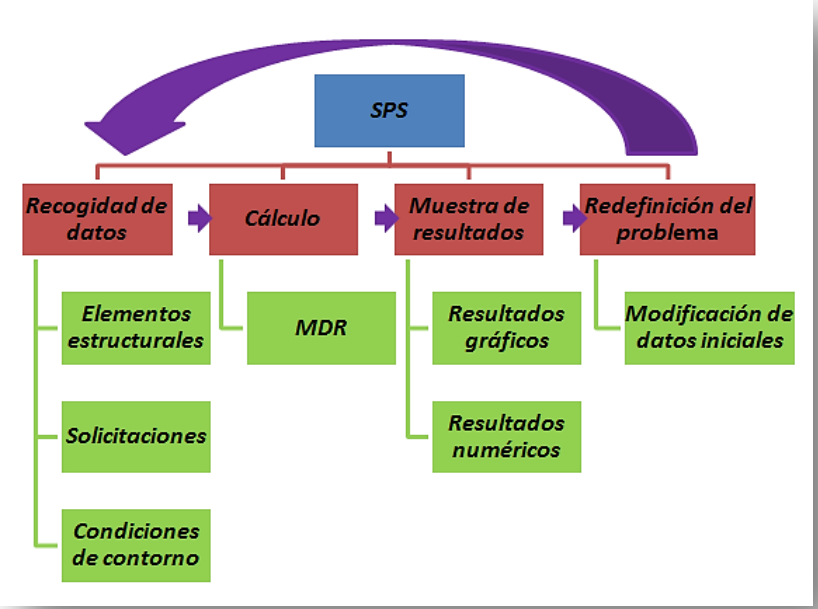
- DatosBean. Clases que contienen una serie de métodos útiles para la manipulación de los datos geométricos, como son los nodos y las barras. Contiene métodos de ordenación de estos datos por diferentes criterios, también contiene métodos de comparación entre datos, etc.

- Grafica. Este paquete contiene las clases encargadas de mostrar gráficamente toda la información que el usuario vaya introduciendo de tal manera que el usuario pueda visualizar su estructura en la pantalla del dispositivo, pudiendo hacer así comprobaciones visuales durante el proceso de construcción o introducción de datos. También contiene este paquete las clases que permiten visualizar por pantalla algunos de los resultados obtenidos, como la deformada, o diagramas de esfuerzos.

- MDR. Es el paquete de cálculo, contiene todas las clases que hacen posible la resolución de la estructura. Como se mencionó anteriormente se fundamenta en el Método Directo de Rigidez. Este paquete ha sido desarrollado en el departamento de Construcciones Arquitectónicas, Ingeniería del Terreno, Mecánica de los Medios Continuos y Teoría de Estructuras de la EII.

5.2.2 Funcionamiento de la aplicación

El proceso desde que el usuario introduce los datos geométricos de la estructura hasta que se obtienen los resultados lleva una serie de fases en la aplicación, la mejor forma de comprenderlo puede ser el siguiente esquema:



Gráfica 10: Esquema de funcionamiento de la aplicación

La recogida de datos consiste en la obtención de todos los datos necesarios para la correcta definición del problema, elementos estructurales (nodos y barras), solicitaciones, vinculaciones de la estructura con el exterior, etc. Para esta parte del proceso el usuario puede utilizar diferentes funciones, como son: crear datos nuevos, editar datos existentes, borrar datos erróneos y ver las características y propiedades de los datos que ya están introducidos. Además el usuario también podrá ver gráficamente

todos los datos introducidos ya que la aplicación va pintando todos los datos que se hayan creado hasta ese momento.

Una vez el usuario ha generado la estructura y la ha definido correctamente, es posible realizar el cálculo. Como se indicó anteriormente, el cálculo se basa en el MDR.

Como se ve en el diagrama anterior, la aplicación ofrece la posibilidad de visualizar los resultados gráficamente o bien numéricamente. En la parte gráfica la aplicación muestra diagramas de esfuerzos axiles, de esfuerzos cortantes y también de momentos flectores de cada una de las barras, también se puede visualizar la deformada de la estructura. En la parte de resultados numéricos, el usuario puede obtener un completo listado de todos los nodos que conforman la estructura y sus características más importantes. También se puede pedir a la aplicación que muestre listados de barras pudiendo elegir el usuario tres opciones diferentes; listados de desplazamientos a lo largo de barra, listados de esfuerzos a lo largo de la barra y listados de valores máximos obtenidos en la barra.

Finalmente si el usuario se da cuenta de algún tipo de fallo que fue cometido a la hora de aportar datos a la aplicación, puede redefinir el problema cuantas veces quiera aunque el proceso de cálculo ya se hubiera producido.

5.3 Guía de usuario de la aplicación

Para iniciar la aplicación simplemente es necesario tocar el icono de la misma que se encuentra en el menú de aplicaciones de nuestro dispositivo:

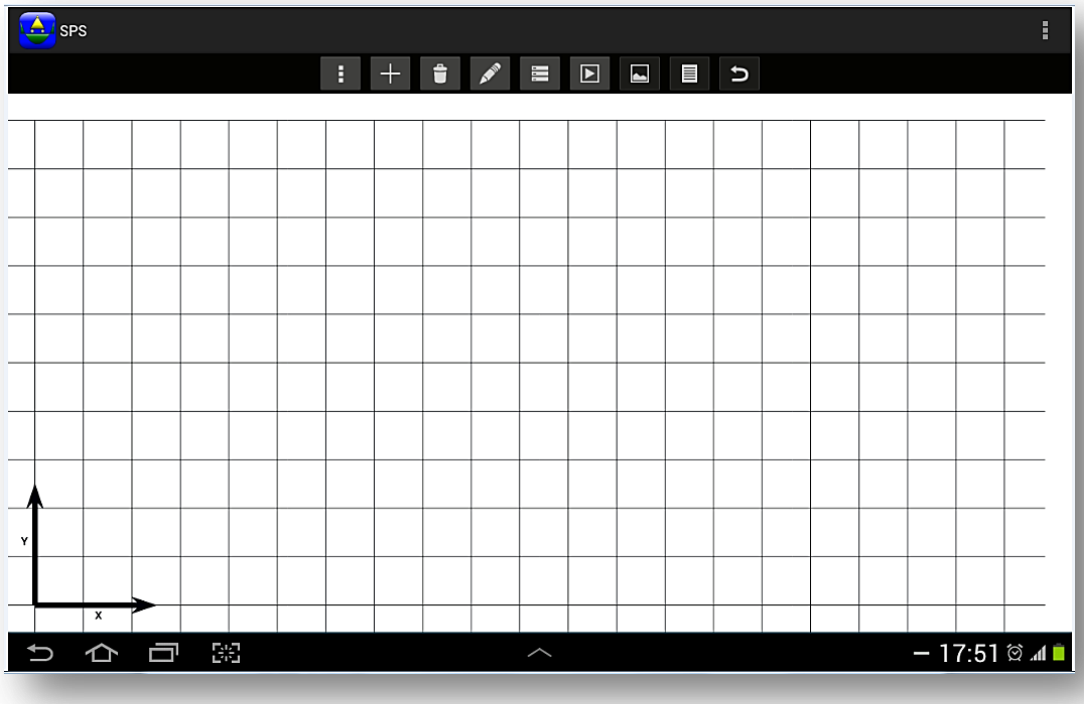
Captura 3: Menú de aplicaciones

5.3.1 Pantalla inicial

En ese momento aparecerá la pantalla inicial de nuestra aplicación. En ella aparecen una serie de iconos, y en la parte inferior se ve el lienzo sobre el que se irá pintando la estructura a medida que la vamos generando.

El color predeterminado para dicho lienzo es negro, pero como en esta guía vamos a introducir diferentes capturas de pantalla para una mejor comprensión de la misma, se ha decidido cambiar ese color por blanco, pensando en generar un documento más claro y legible en el momento de la impresión de este proyecto. Debido a ese cambio, también se han cambiado colores de otros elementos para que el contraste de estos con el fondo sea suficiente para una correcta visualización de los mismos.

La pantalla inicial a la que se hace referencia, ya con los colores modificados es la siguiente:



Captura 4: Pantalla inicial aplicación

El icono que se ve en la parte superior izquierda en un color azul es el icono de la aplicación, no tiene ninguna función más que mostrar qué aplicación se está usando en ese momento.

El de la esquina superior derecha es el icono que corresponde con el botón de menú de nuestro dispositivo, pero no tiene por qué ser en todos los dispositivos en los que ejecutemos la aplicación igual, ni estar situado en la misma posición que en este caso, este botón es característico de cada dispositivo. Si la aplicación está abierta en el momento en el que se pulsa este icono, se mostrara el menú general de nuestra

aplicación, si no es así, se mostrara el menú asociado a la aplicación o activity que se encuentre en la parte más alta de la pila en ese momento.

El resto de iconos que se muestran en una fila horizontal en la pantalla son los que nos permiten interactuar, aportando datos a la aplicación y obteniendo a partir de ellos los resultados que necesitemos. Se puede apreciar que los tres últimos en este momento no se podrían seleccionar, esto es debido a que son los botones correspondientes a la obtención de resultados, y solo es posible ejecutar estas funciones en el momento en el que el cálculo se ha realizado con éxito, y previo a esto, obviamente, se debe generar una estructura.

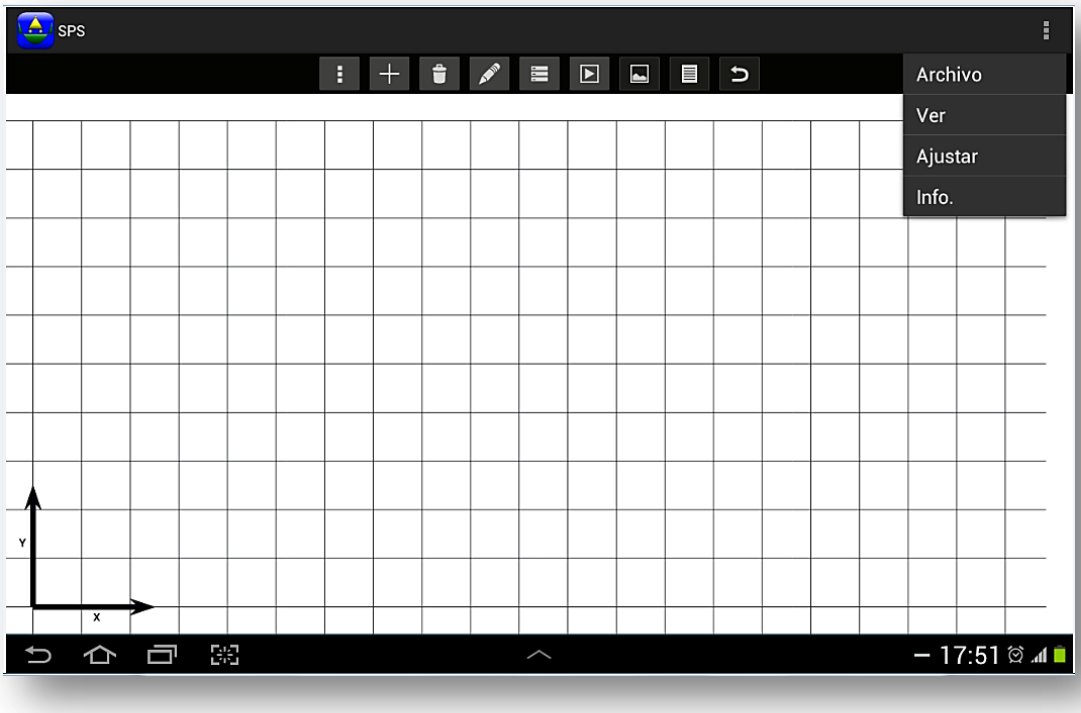
Debajo de esta fila de iconos vemos el lienzo en blanco, que de manera predeterminada viene relleno con una malla que nos permitirá ubicarnos en la construcción de la estructura y hará que posibles errores geométricos en la definición de la estructura sean mucho más detectables.

Y en la parte inferior se encuentran botones característicos de Android que no entraremos a explicar en esta guía ya que, aunque tienen funciones en la aplicación, estas no han sido desarrolladas para la misma, sino que son genéricas en este sistema operativo.

Describimos las funciones de los iconos anteriores detalladamente.

5.3.2 Menú general de la aplicación

Se abre en el momento en el que pulsamos el botón de menú de nuestro dispositivo.



Captura 5: Menú general de la aplicación

El esquema del menú es el siguiente:

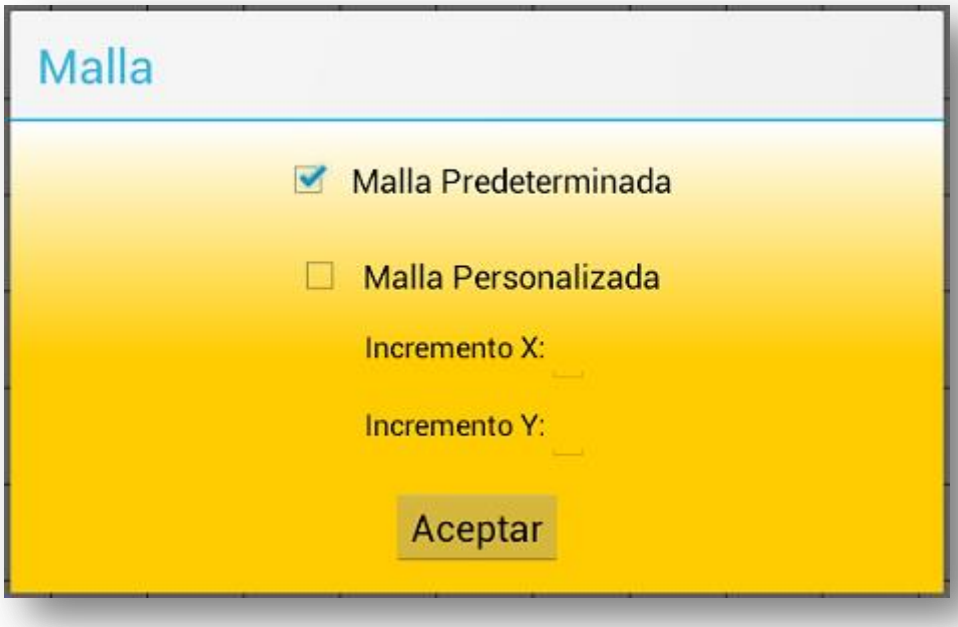
- Archivo. Al pulsarlo se abre un submenú con diferentes opciones:
  - Nuevo proyecto. Al pulsar esta opción la aplicación nos preguntará si queremos crear un nuevo proyecto, si confirmamos genera un nuevo proyecto desde cero, sin guardar el anterior y sin cerrar la aplicación.
  - Abrir proyecto. En esta opción se puede abrir un proyecto que se ha guardado con anterioridad. La aplicación nos muestra la lista de proyectos guardados de la cual tendremos que seleccionar aquel que queramos abrir.
  - Guardar proyecto. Una vez que tenemos un proyecto iniciado podemos guardarlo en nuestro dispositivo pulsando esta opción. Si es el primer proyecto que guardamos, la aplicación se encarga de crear un directorio en la memoria interna del dispositivo, en la cual se guardará el proyecto en cuestión, antes de hacerlo la aplicación nos pide que le demos un nombre al mismo
  - Salir. Si pulsamos esta opción, la aplicación nos mostrará un mensaje de confirmación, si confirmamos, la aplicación se cerrará por completo sin guardar los cambios.

    Nota: No se incluyen capturas de pantalla ya que es muy intuitivo el uso de estos apartados.
- Ver. Presionando esta opción del menú se abre un diálogo en el cual el usuario puede elegir todos y cada uno de los elementos que se pueden crear y visualizar en la aplicación, el diálogo que aparece es el siguiente:

Captura 6: Menú Ver

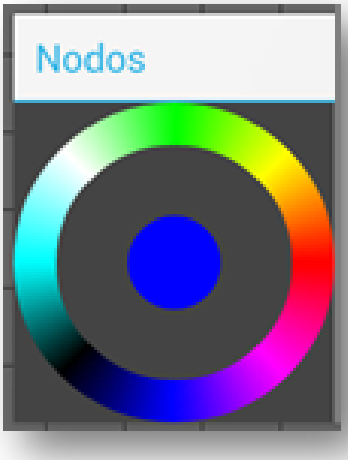
El usuario puede marcar o desmarcar las casillas de cada elemento en función de que quiera o no que se muestren en el lienzo.

- Ajustar. Este botón nos permite ajustar una serie de características en la aplicación, estas serán:
    - Malla. La malla es la cuadricula que se muestra en el lienzo al arrancar la aplicación. En este punto podemos dejar activa la malla predeterminada, en la cual la separación tanto entre líneas verticales como horizontales es de 1m. O bien podemos personalizar la malla ajustando la separación entre líneas verticales y horizontales a nuestro antojo.



Captura 7: Menú Malla

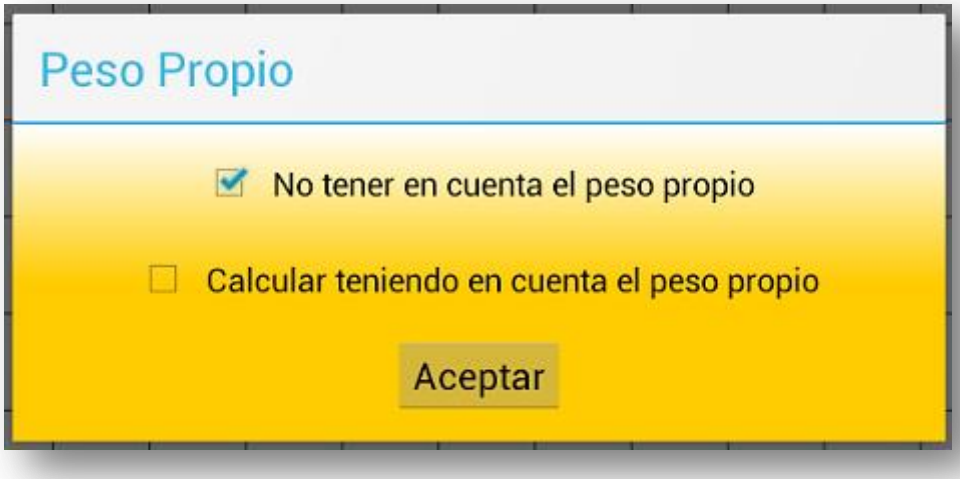
- Colores. Al pulsar esta opción se abre un diálogo en el que tenemos que elegir el elemento al que deseamos cambiar el color, una vez que lo pulsamos se abre una ruleta con diferentes colores, para modificar el color predeterminado del elemento correspondiente, debemos deslizar el dedo sobre la ruleta, viendo como en el cirulo interior va apareciendo el color sobre el que estamos tocando en la ruleta, en el momento en el que se visualiza el color deseado, lo seleccionamos pulsando el círculo interior.



Captura 8: Diálogo modificación de color

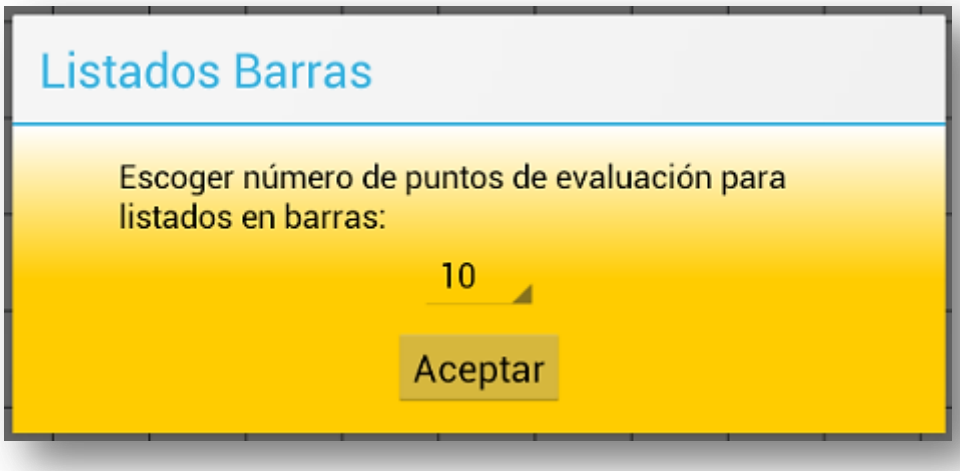
- Peso propio. Esta opción es la que nos permite calcular teniendo en cuenta el peso propio de la estructura, siempre y cuando antes hallamos definido la densidad en todas y cada una de las

barras que componen la estructura. Esta opción se puede activar en cualquier momento antes de pulsar el botón de cálculo, si se hiciera después, deberíamos realizar de nuevo el cálculo para que se tenga en cuenta la modificación en los resultados.



Captura 9: Menú Peso Propio

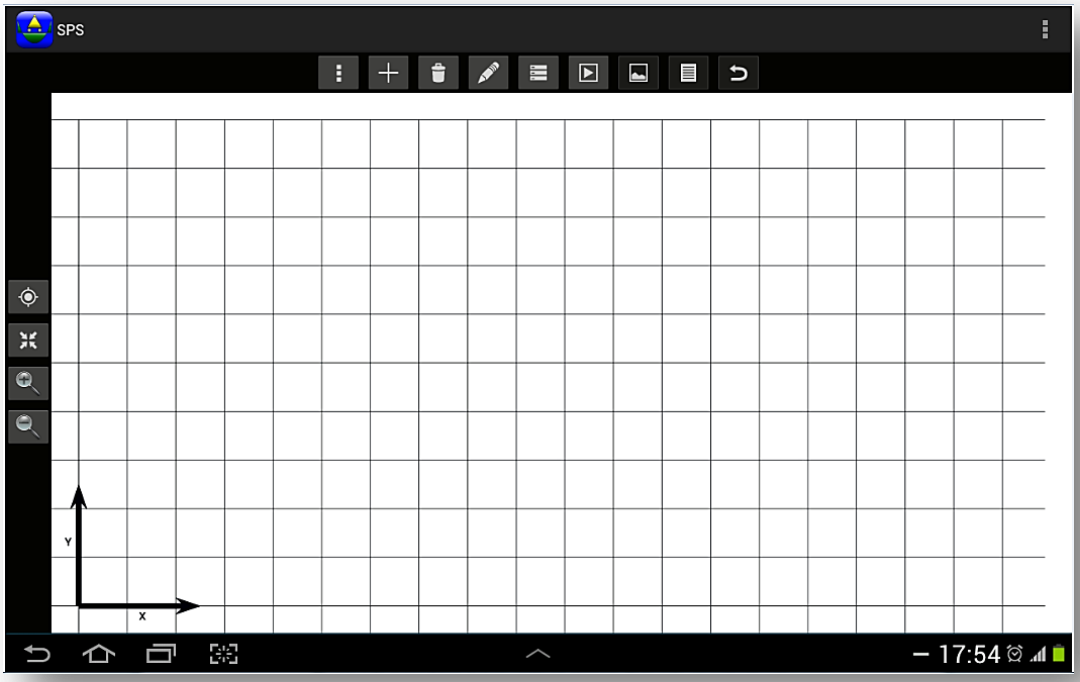
- Listado de barras. Esta opción nos permite ajustar el número de puntos en los que se evaluará cada barra para mostrar los listados tanto de desplazamientos como de esfuerzos en todos esos puntos de cada barra. El número de puntos de evaluación esta predeterminado en 10, pero se podrá modificar desde 5 hasta 300.



Captura 10: Menú Listado Barras

5.3.3 Icono de apertura de menú de opciones gráficas

Es el primer icono empezando por la izquierda de la fila de iconos horizontal que se encuentra en la parte de arriba de la pantalla. Cuando pulsamos este icono, la aplicación muestra una nueva fila de iconos vertical en la parte izquierda de la pantalla, si lo volvemos a pulsar, esta fila desaparece de nuevo. Esto se ha hecho así para aprovechar al máximo el tamaño de la pantalla, de tal manera que esta fila de iconos no nos robe espacio cuando no necesitamos utilizar alguno de sus comandos, para dispositivos con pantallas más o menos grandes esta opción parece innecesaria, pero se hace muy útil cuando estamos utilizando dispositivos con menor tamaño de pantalla como pueden ser los teléfonos móviles.



Captura 11: Pantalla inicial con menú Opciones Gráficas desplegado

Estos iconos verticales controlan la parte gráfica de la aplicación. El primero de ellos, actúa como un conmutador mostrando y ocultando la malla, también podemos mostrar u ocultar la malla desde el submenú *ver* del menú principal de la aplicación.

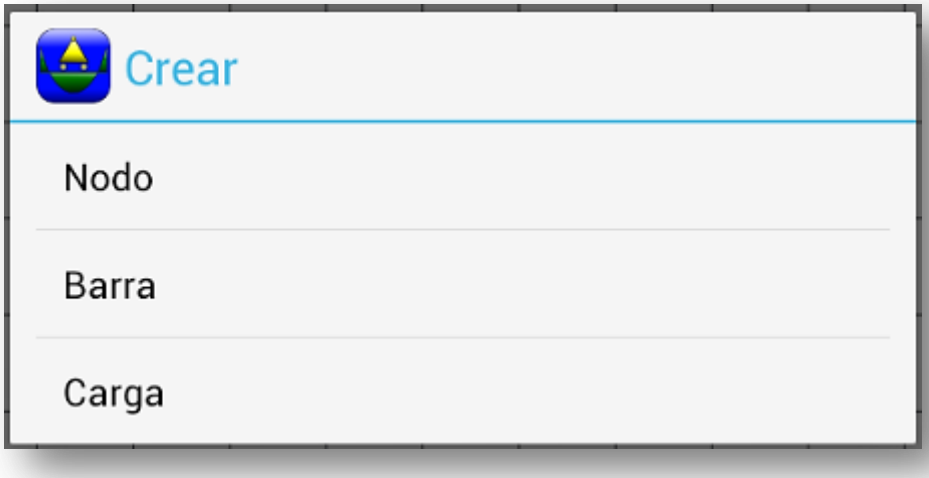
El segundo icono hace que se restaure el zoom de la estructura hasta colocarse al máximo tamaño posible en el que se ve toda la estructura.

El tercer y cuarto icono son comúnmente conocidos por ser los iconos típicos de zoom+ y zoom- respectivamente. Aparte de con ellos, el usuario puede hacer zoom en la estructura acercando o alejando dos dedos cuando están situados sobre la pantalla, esto se explicará más delante en el punto pantalla gráfica.

Se debe señalar que además de estos cuatro iconos, si expandimos este menú vertical cuando la aplicación ha realizado ya el cálculo, se mostrarán las opciones gráficas asociadas a diagramas y a deformada, estos puntos se explicaran más adelante también.

5.3.4 Icono de creación

Es el segundo icono de la fila horizontal de iconos que aparecen en la parte superior de la pantalla, este icono es el que nos va a permitir crear todos los componentes de la estructura. Al pulsarlo aparece un menú emergente o de diálogo en el que tenemos que seleccionar aquel tipo de elemento que queremos crear.



Captura 12: Menú emergente creación

Se debe tener en cuenta que la aplicación no nos va a permitir crear una barra si no hemos creado al menos dos nodos primero, ya que una barra en esta aplicación se define por dos nodos. Del mismo modo tampoco podemos crear cargas sobre nodos si no hemos creado antes un nodo y no podemos crear carga sobre barra si no hemos creado antes una barra.

5.3.4.1 Crear nodo

Al pulsar crear nodo se nos abrirá un diálogo con todas las opciones y características que podemos asociar a un nodo, es quizá uno de los diálogos más amplios de la aplicación, pero es necesario que tenga este tamaño para que el nodo pueda quedar completamente definido.

Captura 13: Diálogo Crear Nodo

En cualquier caso este diálogo solo tiene tres campos obligatorios. El primero de ellos es el número de identificación de nodo, que se autocompleta asignando el numero entero más pequeño que se encuentra no asociado a ningún otro nodo empezando por el 1, pero también podemos asignar el numero entero que deseemos a cada nodo. Los otros dos campos obligatorios se corresponden con los campos asociados a las coordenadas del nodo.

En el caso de que existan condiciones de contorno asociadas el nodo que estamos generando se deben revisar el resto de campos.

Si es un apoyo, se debe activar el check box de apoyo y dejar pulsado el tipo de apoyo que corresponda. Si el nodo es un apoyo, podremos asociarle desplazamientos prescritos activando su check box, solo se podrán introducir desplazamientos prescritos para aquellos desplazamientos que el nodo restringe. De igual manera, aquellos nodos para los que una variación de su ángulo hace que varíe también su funcionamiento, se activa la opción de introducir el ángulo automáticamente en la parte inferior del diálogo.

También se pueden generar apoyos elásticos activando su check box independientemente de que se haya definido el nodo como apoyo o no, eso sí, si el nodo se ha definido como apoyo, solo se podrán introducir apoyos elásticos en aquellas direcciones en las que el apoyo no restringe el movimiento.

Si se han elegido opciones de apoyo, apoyo elástico o desplazamientos prescritos de forma equivocada, se eliminarán todos los datos si se desactiva el check box correspondiente.

Para guardar el nodo se debe pulsar el botón de guardar nodo. Tras guardar el nodo la aplicación nos preguntará si queremos generar otro, si se confirma, la aplicación vuelve a abrir el diálogo de crear nodo.

5.3.4.2 Crear barra

Una vez que hemos creado al menos dos nodos la aplicación nos permitirá crear una barra. En este caso también se abrirá un diálogo bastante amplio.

Captura 14: Diálogo Crear Barra

Lo explicamos de arriba hacia abajo.

El primer campo, al igual que en crear nodo es el número de identificación de barra. Vale la explicación dada para este campo en crear nodo.

Después deberemos escoger tanto el nodo inicial como el final de las listas desplegables, estas listas contienen todos los nodos generados y guardados hasta el momento de abrir este diálogo.

Los siguientes campos sirven para definir las características de la barra, serán obligatorios todos los campos marcados con un *.

En cuanto al apartado de libertades, sirve para asociar libertades en los extremos de barras, se puede introducir un máximo de dos libertades en total. Los números hacen referencia al extremo de la barra en el que se crea la libertad (1=nodo inicial, 2=nodo final), y las letras x, y, z hacen referencia a la dirección en la que se genera la libertad, z en este caso será libertad de giro. Aunque todas las libertades añadirán información a la estructura, la única libertad que se representará gráficamente en caso de crearla es la de giro.

En el siguiente apartado, llamado “librería perfiles acero”, la aplicación incluye una pequeña librería de perfiles de acero, los cuales tienen ya definidas todas sus características, el usuario podrá seleccionar cualquiera de ellos en la lista desplegable correspondiente, en el momento en el que se selecciona un perfil, los campos se rellenaran automáticamente con los valores asociados a dicho perfil, y no se podrán modificar, a no ser que se coloque la lista desplegable de nuevo en su posición inicial de “Seleccionar perfil”.

Para facilitar aún más la creación de barras se ha añadido a la aplicación la opción de crear una librería personalizada, esta librería en un principio estará vacía, y para rellenarla el usuario deberá pulsar el icono de creación que aparece en el diálogo.

Una vez pulsado se abre el siguiente diálogo para introducir un nuevo tipo de perfil a la biblioteca:

Captura 15: Diálogo Nuevo Elemento

En este diálogo serán obligatorios aquellos campos marcados con un *. El primero de ellos es el nombre con el que se identifica la sección, en este caso habrá que completarlo ya que la aplicación no lo hará automáticamente. El resto de ellos son datos típicos que se asocian a cualquier tipo de perfil y creemos que no son necesarios explicar. Una vez que hemos definido el elemento, pulsando el botón guardar elemento este se guardara en la biblioteca personalizada, al pulsar el botón, además de guardarlo la aplicación vuelve a la ventana anterior.

En este momento si miramos el desplegable de la biblioteca personalizada, veremos que el perfil que acabamos de crear aparece en él, seleccionándolo, los campos se autocompletarán igual que ocurría al seleccionar uno de la librería de perfiles predeterminada. Al igual que antes, para poder modificar los datos debemos colocar el desplegable en su primera posición de “Seleccionar perfil”.

Podemos eliminar perfiles de esta biblioteca personalizada pulsando el icono de eliminar que aparece en el diálogo de crear barra. Al pulsarlo aparece un diálogo con una lista desplegable que contiene todos los perfiles creados hasta ese momento, seleccionaos el que queremos eliminar y pulsamos el botón eliminar elemento, la aplicación eliminara ese elemento y volverá al diálogo de crear barra.

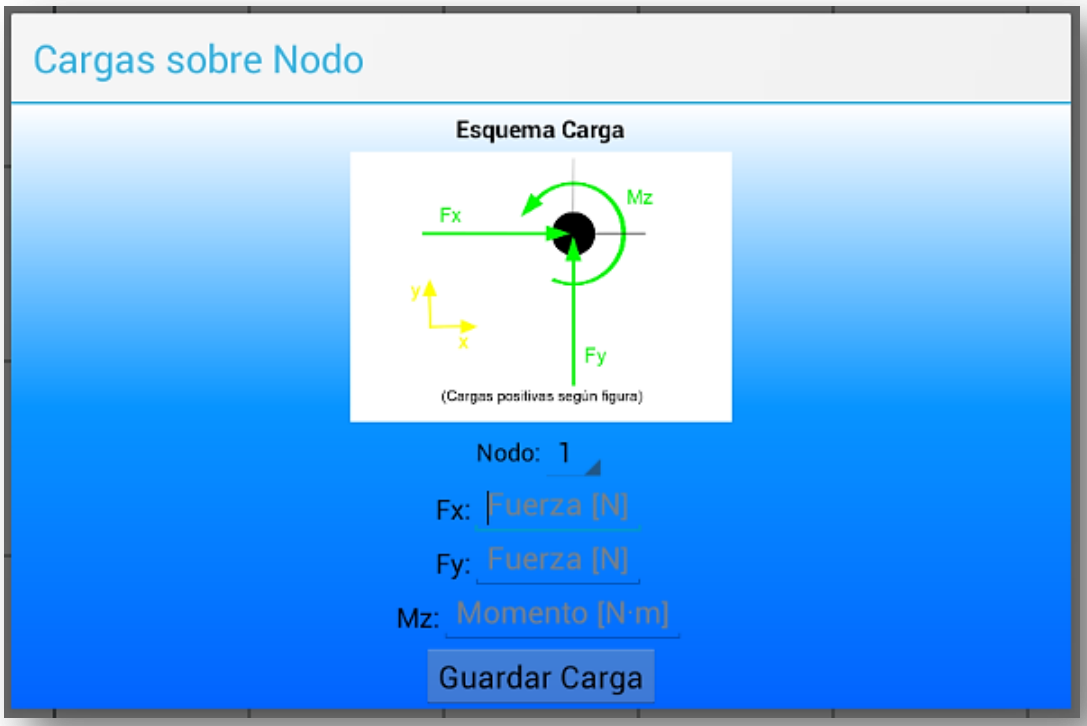
Una vez que hemos dado las características que deseamos a la barra, pulsando el botón de guardar barra, la barra se guardará y la aplicación nos preguntará si queremos crear otra, si confirmamos, se volverá a abrir el diálogo de crear barra.

5.3.4.3 Crear carga

Cuando seleccionamos esta opción en el menú emergente del icono crear, se abre otro menú emergente que nos pregunta donde queremos crear esa carga, sobre un nodo o sobre una barra, ni que decir tiene que para crear una carga sobre uno de estos elementos previamente se deben haber creado.

5.3.4.3.1 Cargas sobre nodo

Al pulsar esta opción se abre el siguiente diálogo:



Captura 16: Diálogo añadir Carga sobre Nodo

Al igual que en el resto de diálogos de cargas, en primer lugar se muestra un esquema de la carga a la cual se asocian los campos que se deben rellenar para completar la información de la misma. Las cargas introducidas con signo positivo coincidirán con el sentido de las cagas observadas en el esquema.

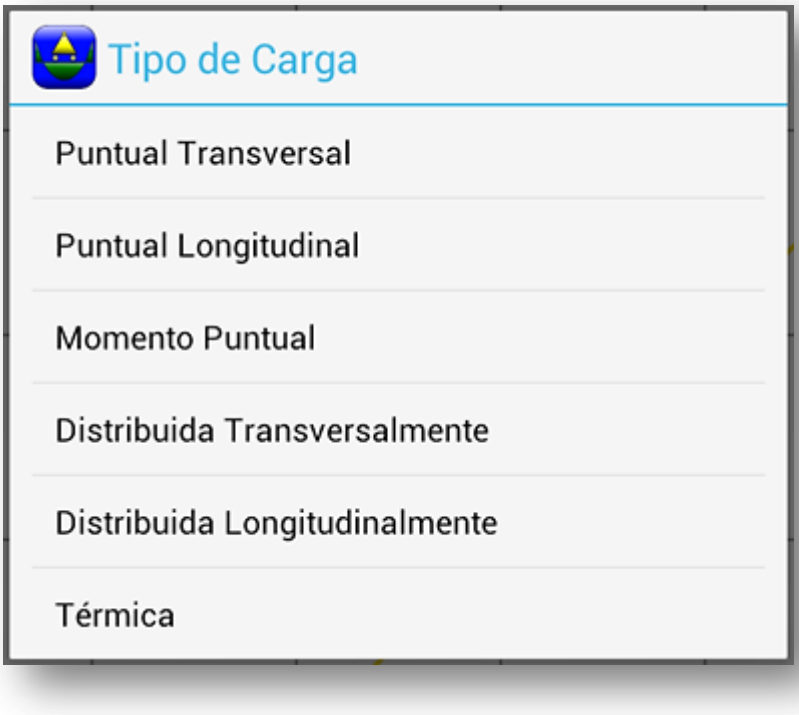
Lo primero que debemos seleccionar es el nodo en la lista desplegable al cual queremos asociar las cargas que vamos a introducir. Este desplegable contiene todos los nodos que se han creado hasta el momento de abrir el diálogo.

Después deberemos introducir el valor de cada una de las cargas, si alguna de ellas no existe o su valor es cero se puede poner cero en su campo o dejarlo sin rellenar, la aplicación actuará de igual modo en ambos casos.

Una vez que hemos creado las cargas tal y como deseábamos, la carga se guardará pulsando el botón guardar carga.

5.3.4.3.2 Cargas sobre barra

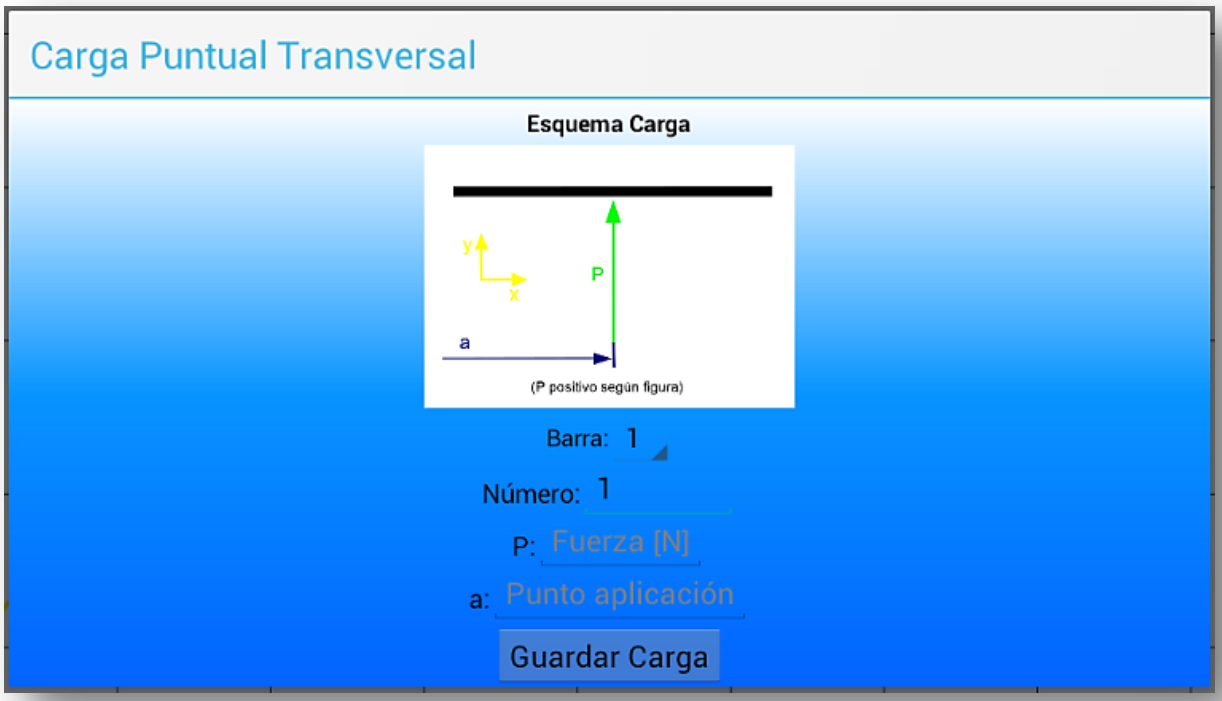
Serán muchos los tipos de carga que podemos crear sobre una barra, por ello y para poder seleccionar cada uno al pulsar esta opción se abre un nuevo menú emergente con todas las opciones. El menú es el siguiente:



Captura 17: Menú emergente Tipo de Carga

Explicamos cada una de ellas a continuación.

- Carga Puntual Transversal. Al pulsar la opción de carga puntual transversal se abre el siguiente diálogo:



Captura 18: Diálogo Crear Carga Puntual Transversal

En primer lugar y al igual que todas las pantallas de creación de cargas nos aparece un esquema de la carga que estamos creando.

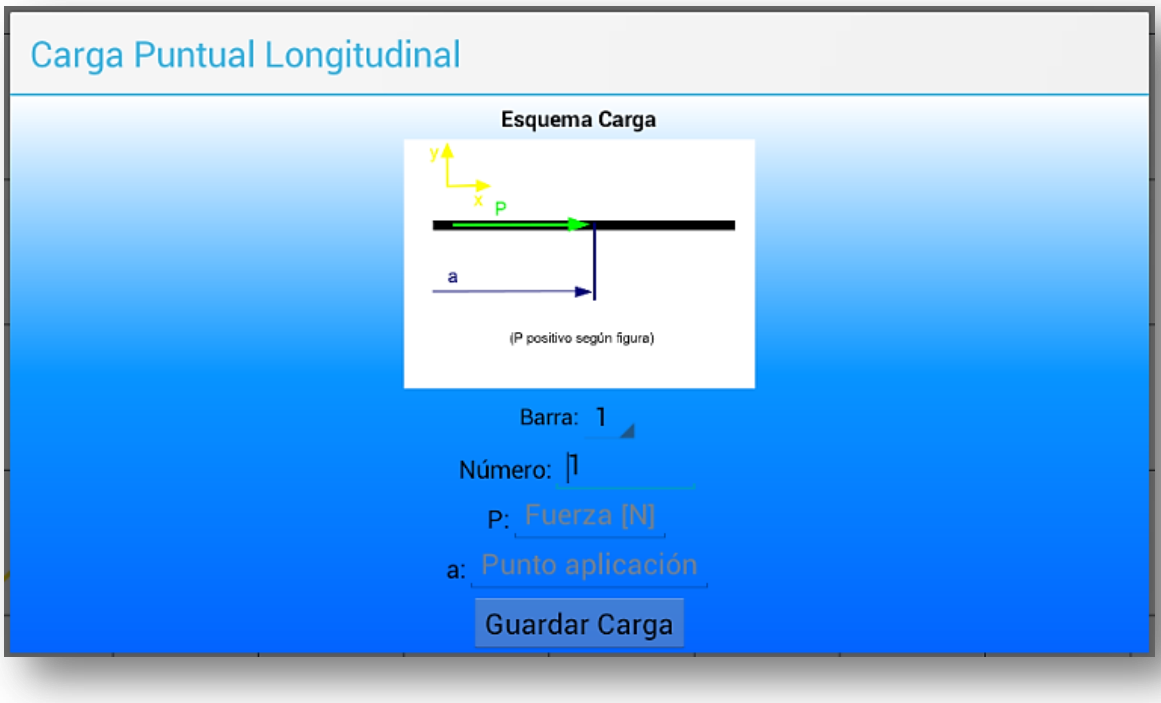
Tendremos que seleccionar la barra sobre la que queremos introducir la carga.

El primer campo a rellenar es el número de identificación de la carga, que se rellenará automáticamente al igual que ocurría en el caso de creación de nodos y barras. Si no nos gusta el número asignado automáticamente podremos asignarle cualquier otro que no esté ocupado.

El valor de P será el valor cuantitativo de la carga en N.

El valor de a será el punto de aplicación de la misma en la barra, este valor puede valer entre cero y uno, ambos inclusive, siendo cero el nodo inicial de la misma y uno el final, habrá que tener en cuenta por tanto qué se asignó como nodo inicial y final al crear la barra.

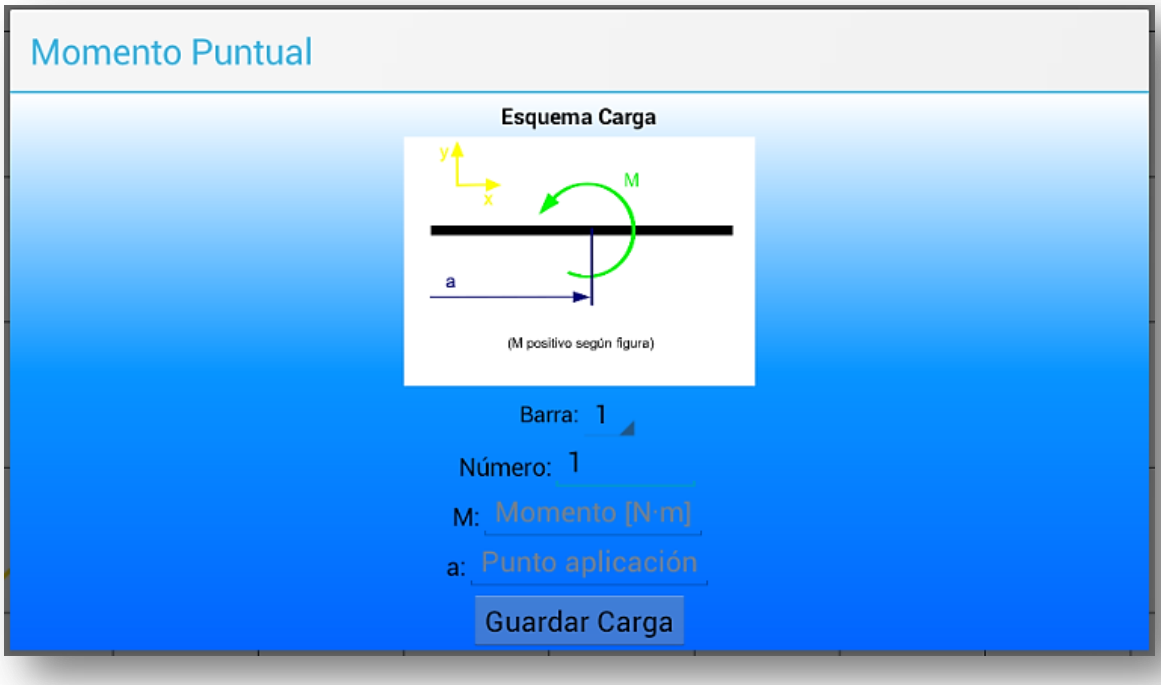
- Carga Puntual Longitudinal. El diálogo es el siguiente:



Captura 19: Diálogo Crear Carga Puntual Longitudinal

La explicación de este diálogo es similar a la del anterior.

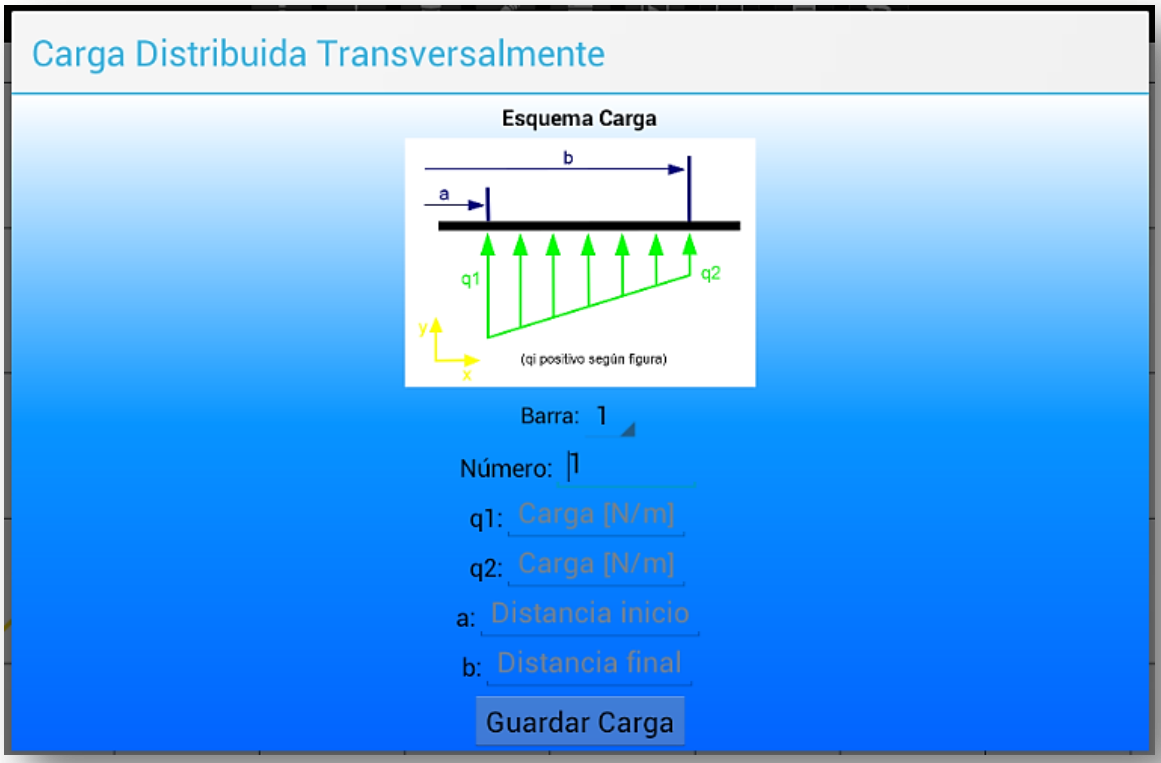
- Momento Puntual.



Captura 20: Diálogo Crear Momento Puntual

Diálogo similar a los dos anteriores.

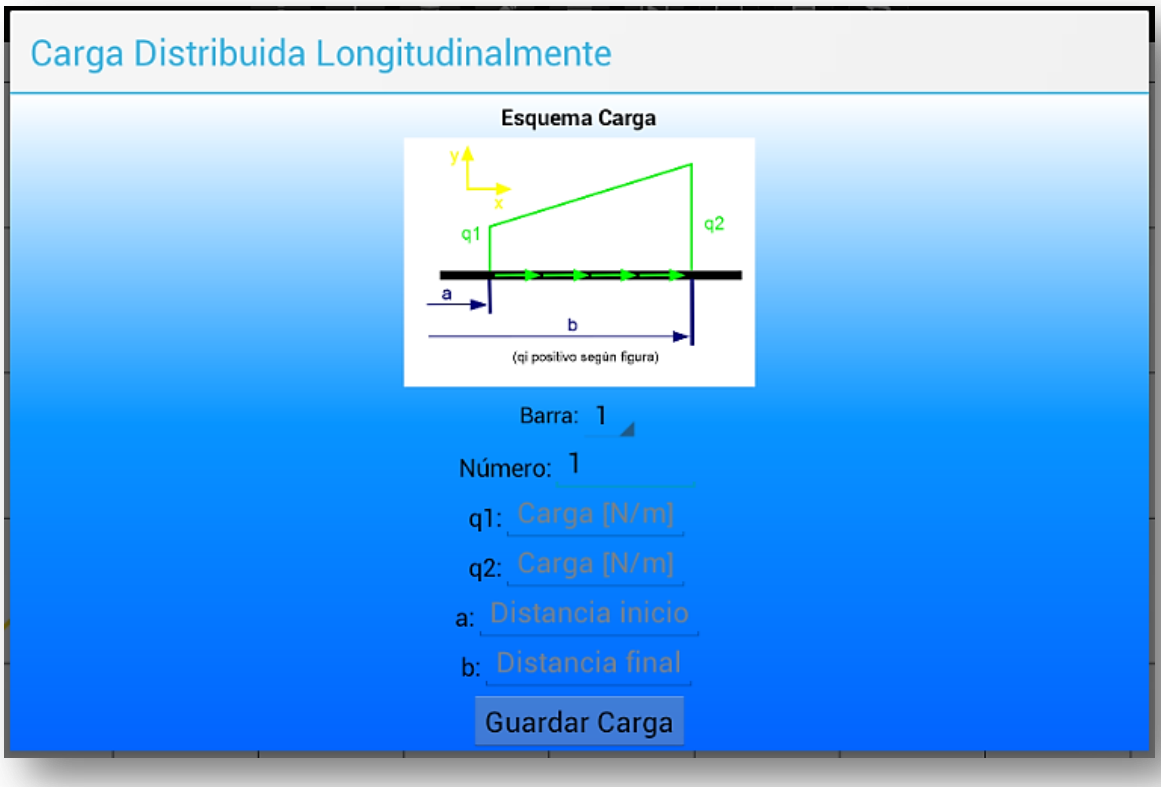
- Carga Distribuida Transversalmente. El diálogo de la carga es el siguiente:



Captura 21: Diálogo Crear Carga Distribuida Transversalmente

Comentamos las novedades de este diálogo frente a los anteriores. En este caso al tratarse de una carga distribuida, en lugar de un valor de carga debemos meter dos, el inicial y el final (q1 y q2 respectivamente), y también deberemos meter dos valores de aplicación, el punto donde comienza la carga, y en el que termina (a y b respectivamente). Al igual que ocurría en los anteriores, a y b deben estar comprendidos entre cero y uno, y además en este caso b no puede ser menor que a.

- Carga Distribuida Longitudinalmente.



Captura 22: Diálogo Crear Carga Distribuida Longitudinalmente

La explicación de esta carga es similar a la anterior.

- Carga Térmica. En este caso se pueden meter también dos tipos de gradiente térmico, en sentido transversal y longitudinal. Advertir que en este diálogo solo se podrán escoger las barras para las que se ha definido coeficiente de dilatación del material y canto de la viga, la aplicación anuncia esto al abrir el diálogo.

Captura 23: Diálogo Crear Carga Térmica

El esquema de la carga solamente se mostrara en el momento en el que seleccionamos el tipo de gradiente en el desplegable correspondiente.

Como en los anteriores, se debe escoger en el desplegable la barra sobre la que vamos a introducir la carga.

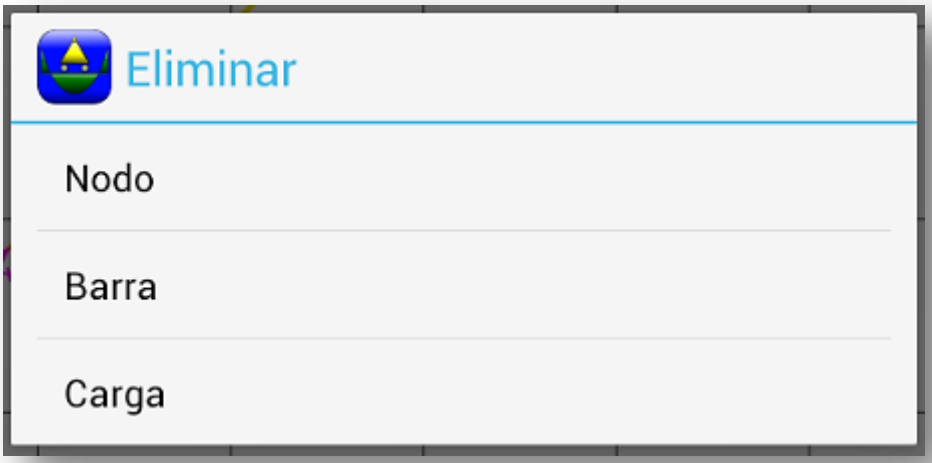
Después escogemos el tipo de gradiente y asignamos el número de identificación de carga.

Tras esto solamente hay que introducir el valor de T1 y T2 que se corresponden con el esquema mostrado.

Cuando pulsemos el botón guardar carga en cualquiera de los diálogos anteriores se guardará la carga y la aplicación nos preguntara si queremos crear una carga del mismo tipo que la que acabamos de generar, si confirmamos se abrirá de nuevo el diálogo correspondiente.

5.3.5 Icono de eliminación

Es el tercer icono de la fila horizontal de iconos que aparecen en la parte superior de la pantalla, a partir de este icono podremos borrar aquellos elementos que hemos guardado y no debieran existir. Al pulsarlo aparece un menú emergente o de diálogo en el que tenemos que seleccionar aquel tipo de elemento que queremos eliminar. El menú emergente será:

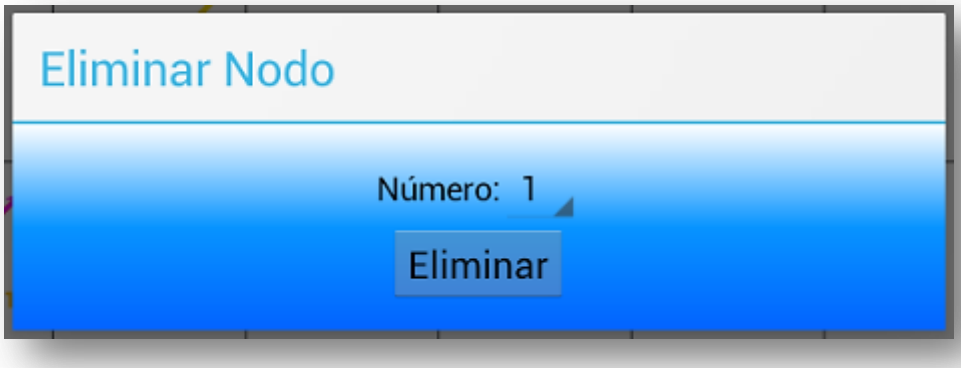


Captura 24: Menú Emergente Eliminar

Explicamos las tres opciones.

5.3.5.1 Eliminar nodo

Cuando pulsamos esta opción en el menú anterior se nos abre el diálogo de eliminación de nodo.



Captura 25: Diálogo Eliminar Nodo

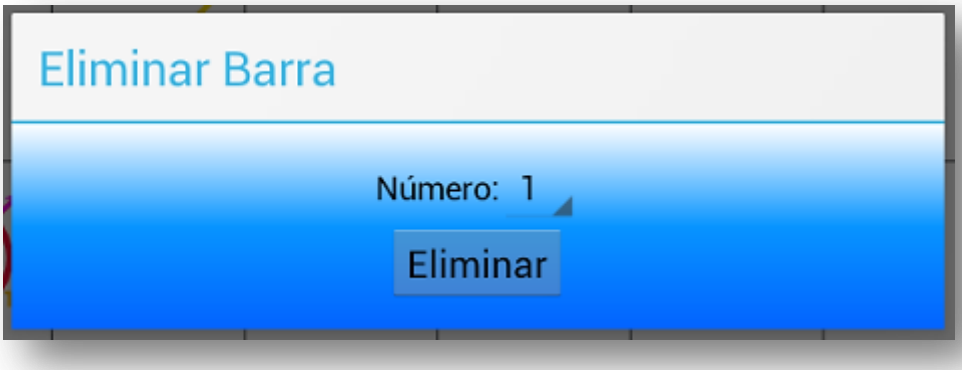
Eliminar un nodo es muy simple. Sólo hay que escoger el número de nodo que se desea eliminar en el desplegable, que contendrá todos los nodos creados hasta entonces, y pulsar el botón de eliminar. Antes de eliminar el nodo, la aplicación nos

preguntará si estamos seguros de que deseamos eliminar el nodo, si confirmamos, la aplicación lo borrará.

Hay que señalar que si eliminamos un nodo se eliminarán todas las barras que empiecen o terminen en él, es decir, se borrarán todas las barras que estén asociadas a ese nodo.

5.3.5.2 Eliminar barra

Es una acción muy parecida a la anterior. La pantalla es prácticamente igual.

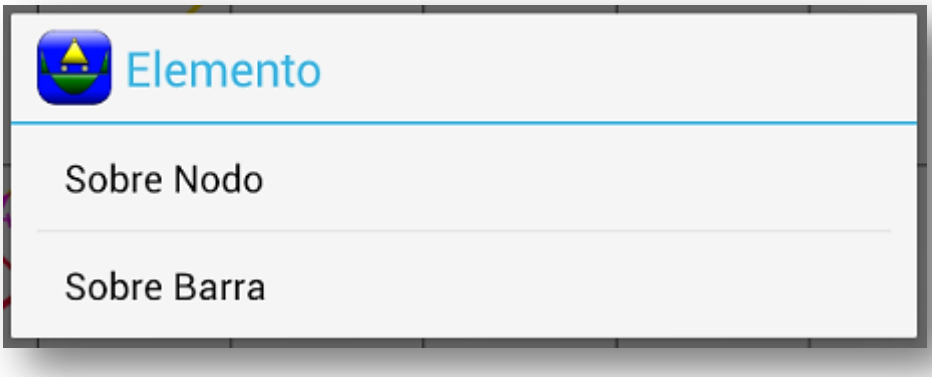


Captura 26: Diálogo Eliminar Barra

Al igual que antes únicamente debemos seleccionar el número de barra en el desplegable y pulsar eliminar. Al igual que antes, la aplicación nos pedirá que confirmemos antes de realizar la acción.

5.3.5.3 Eliminar carga

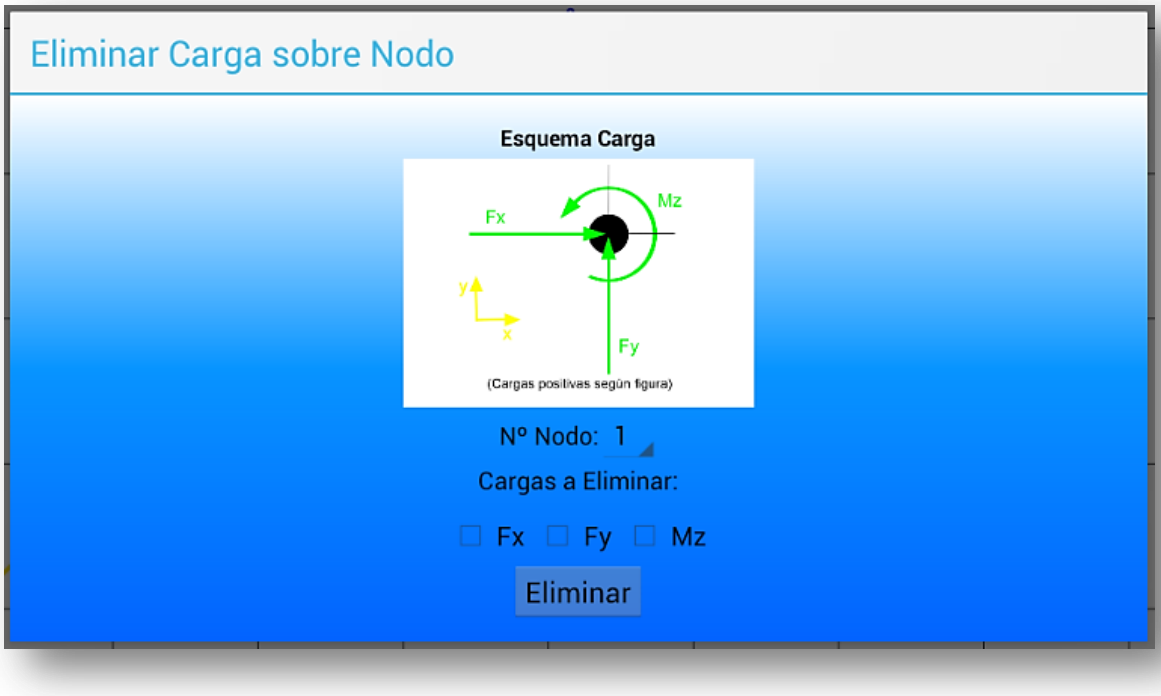
Como existen diferentes tipos de carga, al igual que ocurría en el icono de creación, en este caso se vuelve a abrir un nuevo y pequeño menú emergente con la opción de eliminar una carga que este asociada a un nodo o una que este asociada a una barra.



Captura 27: Menú Emergente Elemento

5.3.5.3.1 Eliminar carga sobre nodo

Al pulsar la opción eliminar carga sobre nodo aparece directamente la siguiente pantalla:



Captura 28: Diálogo Eliminar Carga sobre Nodo

Lo primero que vemos es de nuevo el esquema de la carga. Debemos escoger en el desplegable el nodo al que queremos quitar la carga.

Después marcaremos el check box de aquellas cargas que queremos eliminar.

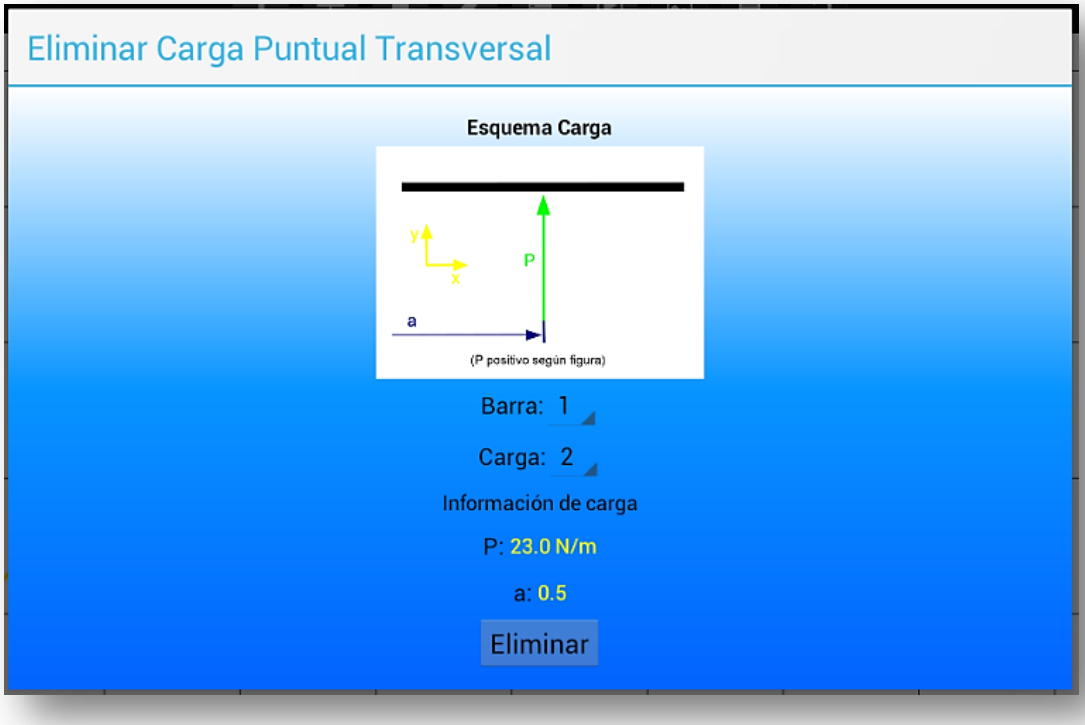
Para eliminarlas definitivamente pulsamos eliminar. La aplicación nos mostrara un mensaje de confirmación, si aceptamos, las cargas marcadas se eliminarán.

5.3.5.3.2 Eliminar carga sobre barra

Si pulsamos la opción sobre barra, se nos abrirá un nuevo menú emergente con todas las opciones de cargas en barra que pudieran existir, será el siguiente:

Captura 29: Menú Emergente Tipo de Carga

- Carga Puntual Transversal. Al pulsar la opción de carga puntual transversal se abre el siguiente diálogo:



Captura 30: Diálogo Eliminar Carga Puntual Transversal

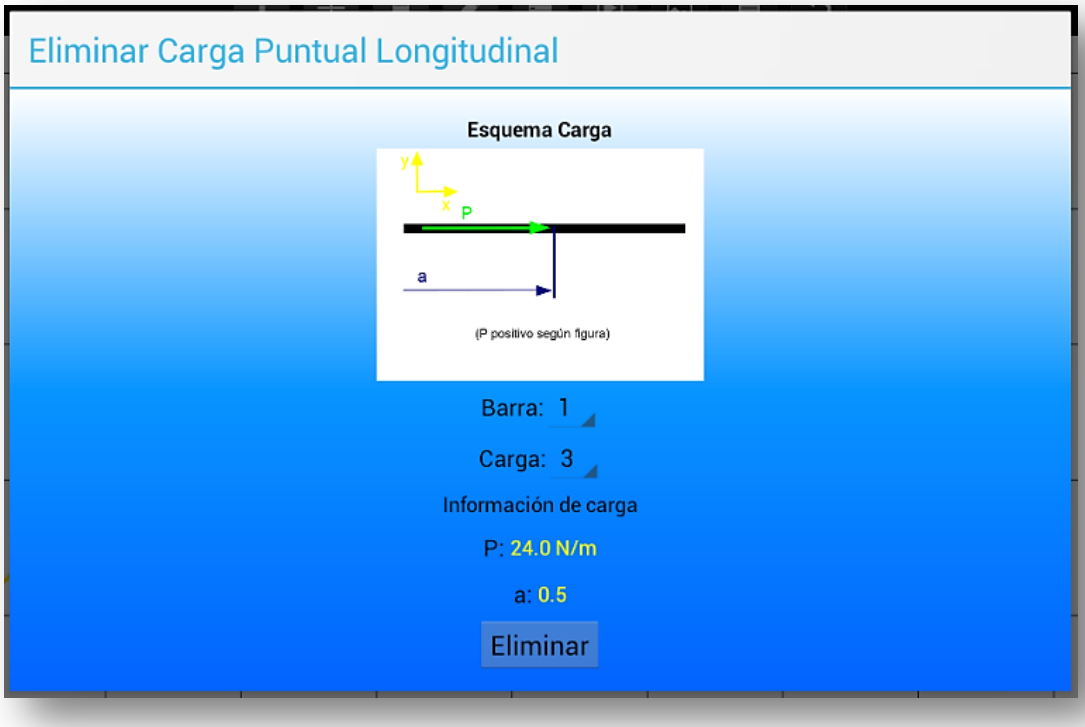
En primer lugar y al igual que todos los diálogos de eliminación de cargas nos aparece un esquema de la carga en cuestión.

Tendremos que seleccionar la barra a la que pertenece la carga que queremos eliminar.

Después seleccionamos el número de la carga a eliminar, automáticamente la aplicación nos muestra los valores de P y de a asociados a esa carga, para que nos sea más sencillo identificarla y saber si realmente es esa la carga que queremos eliminar.

Una vez que estemos seguros de haber seleccionado la carga adecuada, pulsamos el botón eliminar, se nos muestra un mensaje de confirmación y una vez que aceptemos, la carga quedará borrada.

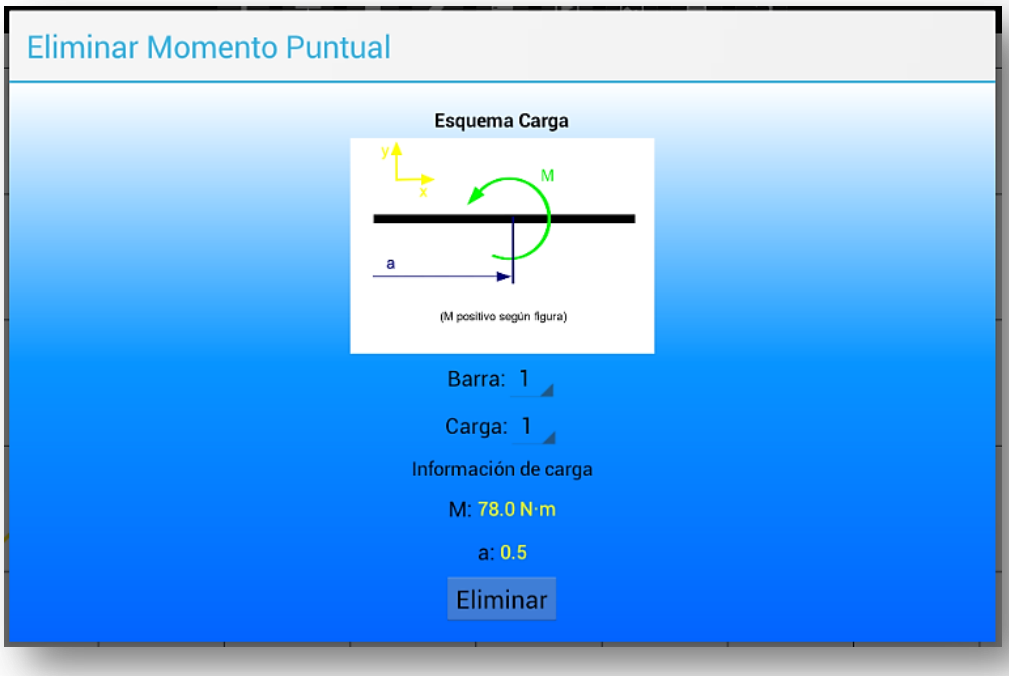
- Carga Puntual Longitudinal. El diálogo que aparece para eta carga es el siguiente:



Captura 31: Diálogo Eliminar Carga Puntual Longitudinal

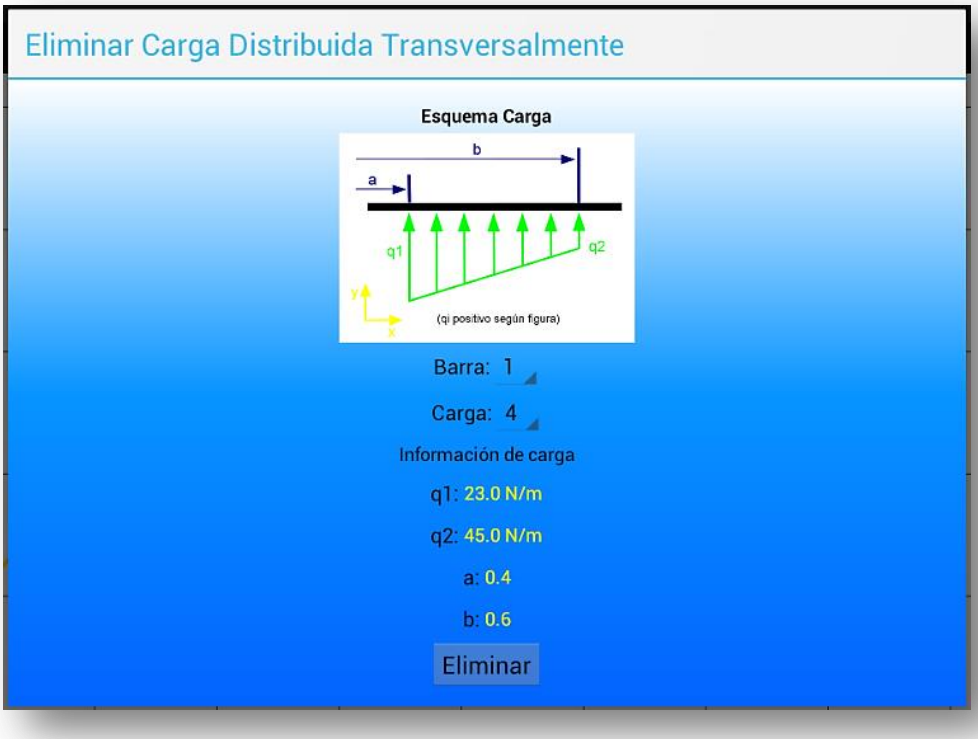
La explicación de este diálogo es similar a la del anterior, teniendo en cuenta que la dirección de la Fuerza en este caso será longitudinal a la barra y no transversal.

- Momento Puntual.



Captura 32: Diálogo Eliminar Momento Puntual

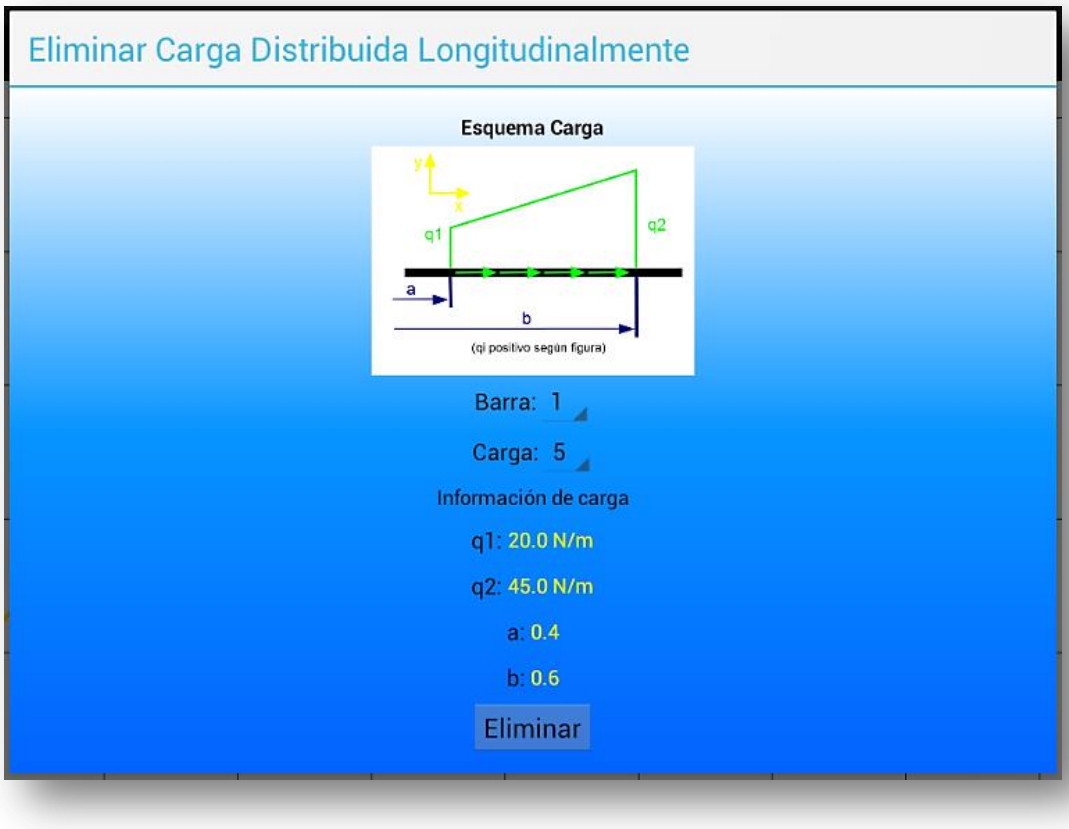
- Carga Distribuida Transversalmente. El diálogo será:



Captura 33: Diálogo Eliminar Carga Distribuida Transversalmente

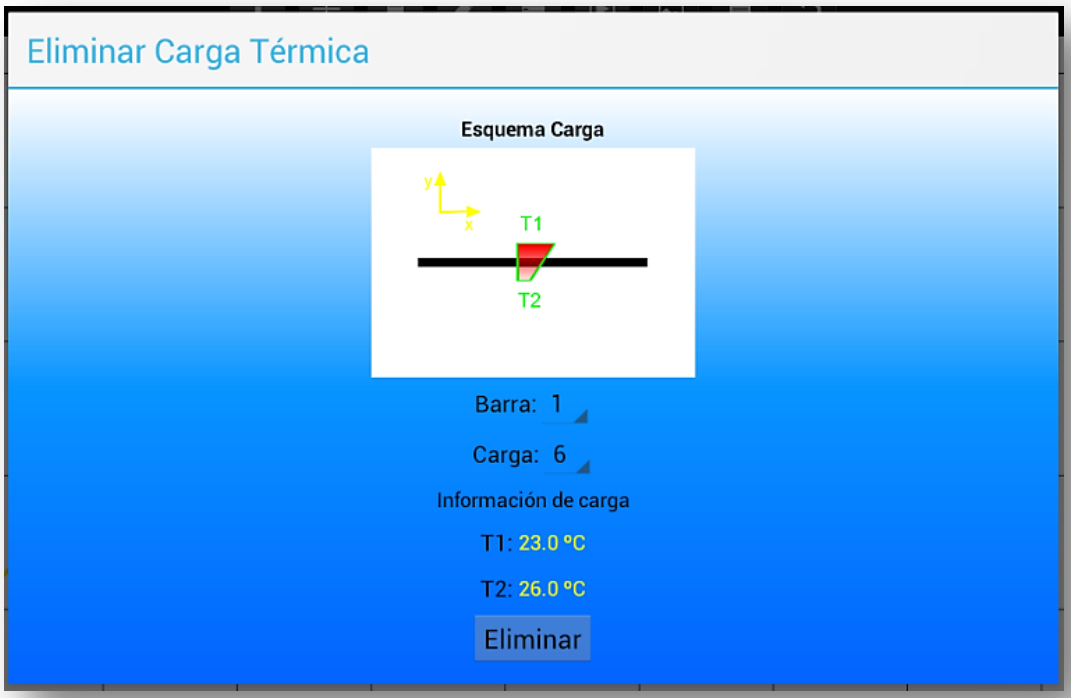
En esta pantalla el funcionamiento es el mismo que en las anteriores, lo que ocurre es que se muestra la información correspondiente a la carga en cuestión, y tiene un par de campos más que las anteriores.

- Carga Distribuida Longitudinalmente.



Captura 34: Diálogo Eliminar Carga Puntual Distribuida Longitudinalmente

- Carga Térmica. En este caso no hay que escoger entre gradientes ya que al seleccionar la barra y el número de carga aparece directamente el esquema y la información de la carga en cuestión.



Captura 35: Diálogo Eliminar Carga Térmica

Por lo demás funciona igual que los casos anteriores.

5.3.6 Icono de edición

Es el cuarto icono de la fila horizontal de iconos que aparecen en la parte superior de la pantalla, a partir de este icono podremos editar aquellos elementos que hemos guardado pero algunas de sus características fueron introducidas de forma incorrecta.

Para no extendernos en explicaciones, en este punto únicamente vamos a señalar que el funcionamiento de este icono es muy similar al del icono de creación.

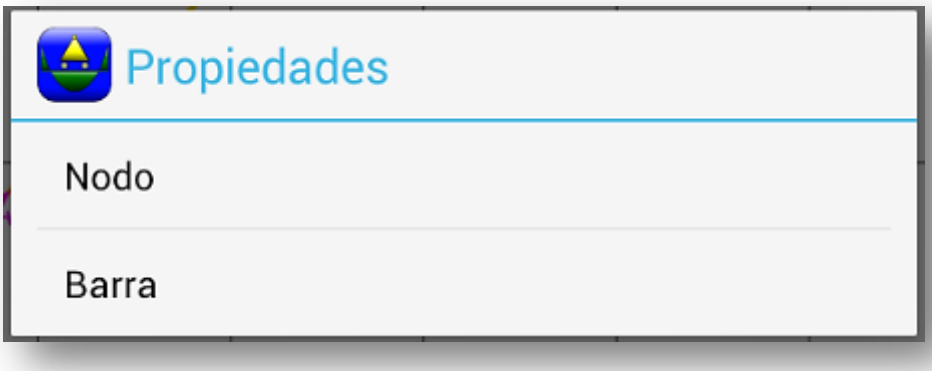
La ventaja de este icono es que nos permite modificar elementos sin tener que borrarlos y volver a crearlos. Además cuando entramos en el diálogo de cada uno de los elementos, que será muy similar al de creación, con la diferencia de que ahora debemos escoger en un desplegable el elemento a modificar y antes le teníamos que asignar el número de identificación, dicho diálogo aparecerá completado con la información que le habíamos asignado a dicho elemento en su creación, y únicamente deberemos borrar y volver a rellenar aquellos campos que estuvieran mal.

Una vez hecho esto pulsamos modificar y el elemento se guardara con las modificaciones realizadas, antes de eso, la aplicación nos muestra un mensaje de confirmación.

5.3.7 Icono de visualización de propiedades

Es el quinto icono de la fila horizontal de iconos que aparecen en la parte superior de la pantalla, a partir de este icono podremos ver las propiedades que hemos asignado a nodos y a barras, dentro de las propiedades de barra también podemos ver las cargas asignadas a dicha barra.

Lo que nos aparece al pulsar el icono es el siguiente menú emergente:



Captura 36: Menú Emergente Propiedades

Al pulsar cualquiera de ellos nos aparece directamente el diálogo correspondiente.

5.3.7.1 Propiedades de nodo

En esta pantalla podemos ver de manera detallada las características que hemos asociado a cada nodo.

Captura 37: Propiedades de Nodo

En primer lugar se debe escoger el nodo en el desplegable del cual se quieren observar las propiedades.

Una vez seleccionado, la primera información que se nos ofrece es la de las coordenadas en las que se encuentra situado dicho nodo.

En adelante la información que se muestra solo será transcendente si el nodo tiene características especiales, como puedan ser: apoyo, apoyo elástico, desplazamientos prescritos o fuerzas en el nodo.

Como información de apoyo la aplicación ofrece los desplazamientos que restringe el apoyo asignado y también el ángulo de apoyo si lo tuviere.

Si el nodo tiene asignado apoyo elástico en alguna dirección en esta pantalla se marcará el check box correspondiente a esa dirección y el valor de la constante elástica en dicho apoyo elástico.

De igual manera que con los apoyos elásticos la aplicación ofrece la información asociada a los desplazamientos prescritos y a las fuerzas en el nodo que se está estudiando.

Una vez comprobados los valores que se querían estudiar, pulsando *back* la aplicación vuelve a la ventana principal.

5.3.7.2 Propiedades de barra

Al pulsar la opción barra en el menú emergente se abre el siguiente diálogo:

Captura 38: Propiedades de Barra

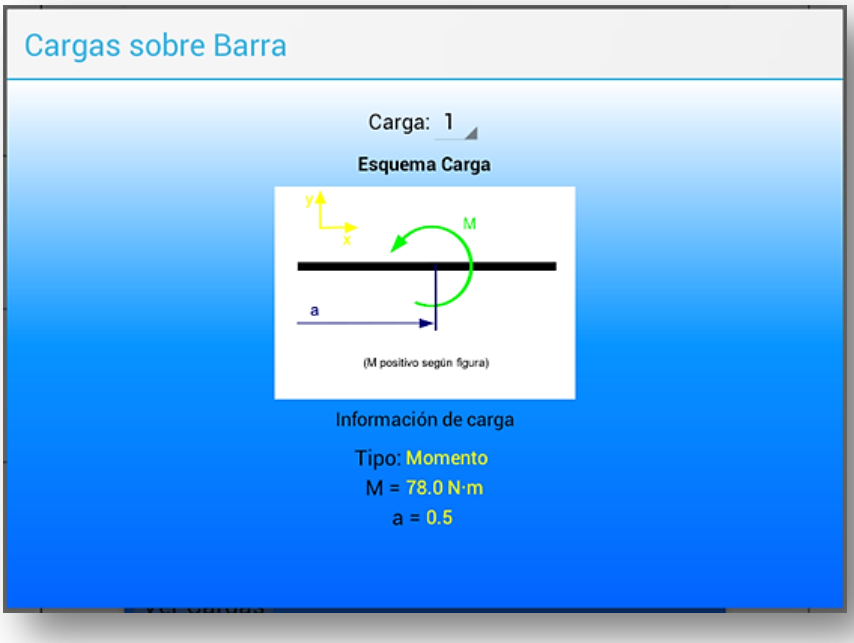
Al igual que en el caso de nodos se debe seleccionar en el desplegable la barra que se quiere estudiar, y en ese momento se mostrará toda la información que este asociada a esa barra.

Se muestra en primer lugar el número que identifica tanto al nodo inicial como al final.

Después, en el apartado de características se muestra toda la información asociada a la sección y al material utilizado.

En el apartado de libertades, en el caso de haberse definido alguna en la creación de dicha barra, tendrán su check box en este punto marcado. El numero 1 identifica al nodo inicial y el 2 al nodo final de la barra, y las letras identifican la dirección sobre la cual se introdujo la libertad.

Después nos indica el número de cargas asociadas a dicha barra. En el caso de que queramos ver más información sobre las mismas solo tendremos que pulsar sobre el botón de *ver cargas*, en ese caso se abrirá la siguiente ventana:



Captura 39: Cargas sobre Barra

Aquí seleccionamos en la lista desplegable el número de la carga de la que queremos obtener más información y nos mostrará su esquema y también su tipo y valores asociados.

Como se trata únicamente de pantallas de información, para volver a la pantalla inicial bastará con pulsar el botón *back* del dispositivo.

5.3.8 Icono de cálculo

Es el sexto icono de la fila horizontal de iconos que aparecen en la parte superior de la pantalla. Al pulsarlo, la aplicación realiza el cálculo si la estructura ha sido definida correctamente.

No se abre ninguna pantalla nueva al pulsarlo.

Si el cálculo se ha realizado con éxito, se mostrará un mensaje que lo indica durante un corto espacio de tiempo. Si no, se muestra un mensaje de error.

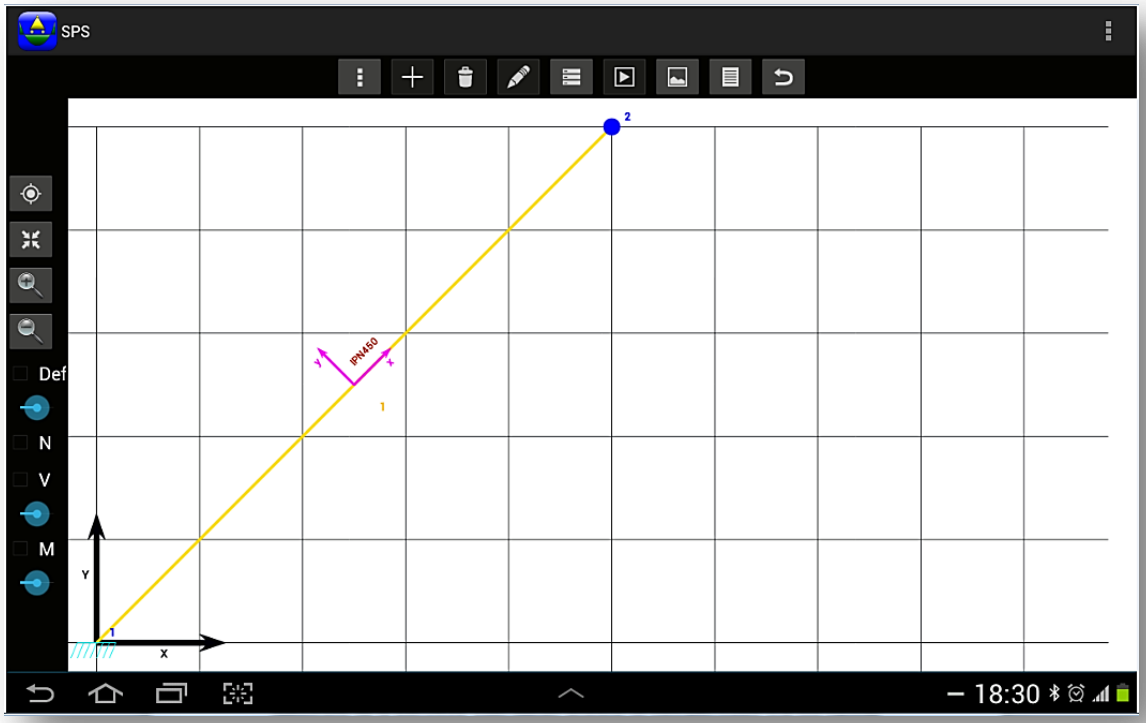
Se puede observar como los iconos que se pueden pulsar ahora en la fila horizontal son sólo los de muestra de resultados y no ya los de recogida de datos. Para poder modificar los datos del problema se debe pulsar el botón de redefinir, que es el último icono de la fila.

5.3.9 Icono de resultados gráficos

Es el séptimo icono de la fila horizontal de iconos que aparecen en la parte superior de la pantalla. Al pulsarlo, la aplicación genera los datos necesarios para poder mostrar los diferentes resultados gráficos.

En principio no se aprecian grandes cambios. Si el menú vertical estaba oculto, al pulsar el icono de opciones gráficas este menú se mostrará y se observará que tiene nuevas opciones que explicaremos a continuación. En el caso de que el menú vertical estuviera visible, se verá como aumenta de tamaño ya que se añadirán las opciones.

La pantalla principal de la aplicación con los dos menús desplegados, será:



Captura 40: Pantalla inicial con Menú de Resultados Gráficos

Las nuevas opciones gráficas, que son las que están justo debajo del icono de zoom- en el menú vertical, son:

5.3.9.1 Visualizar deformada

Se debe marcar la casilla **Def.** Cuando marcamos esta casilla se pintará la deformada de la estructura, justo debajo aparece un pequeño cursor que nos permite modificar la escala a la que se pinta esta deformada, pudiendo reducirla o exagerarla a nuestro antojo.

5.3.9.2 Visualizar diagrama de esfuerzos axiles

Al marcar la casilla **N** se pinta el diagrama de axiles de cada una de las barras de la estructura, se pinta en color azul en principio, pero como en casi todos los elementos, se podrá modificar.

5.3.9.3 Visualizar diagrama de esfuerzos cortantes

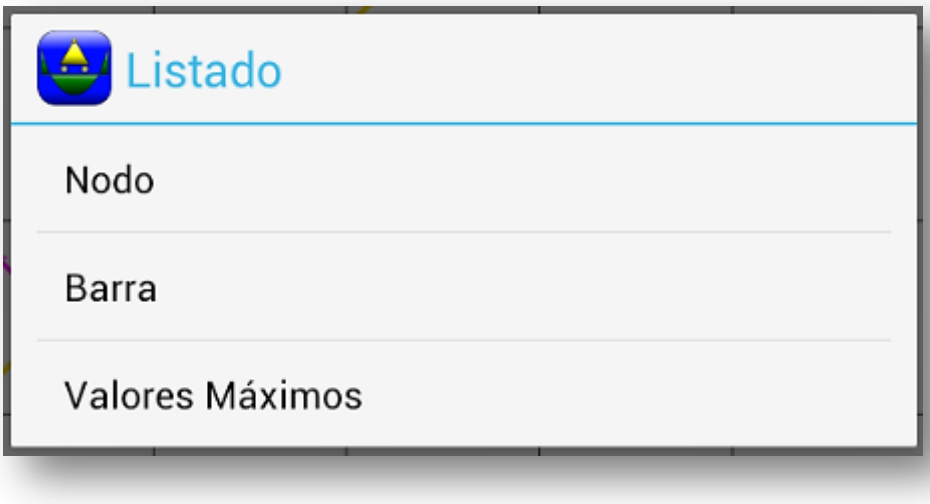
Es la casilla V. Si la marcamos se pintará el diagrama de cortantes de cada una de las barras de la estructura. En principio este diagrama será de color rojo. Debajo aparece un cursor que al igual que antes define la escala con la que se pintan tanto los diagramas de axiles como de cortantes, podremos regular dicha escala a partir de él.

5.3.9.4 Visualizar diagrama de momentos flectores

M. Es la casilla que hace que se pinten los diagramas de momentos flectores de cada barra que conforma la estructura. Serán de color verde. Se puede regular su escala al igual que en los casos anteriores con el cursor que se encuentra inmediatamente debajo de su check box.

5.3.10 Icono de resultados numéricos

Es el penúltimo icono de la fila horizontal de iconos que aparecen en la parte superior de la pantalla. Al pulsarlo, se muestra el siguiente menú emergente para que escojamos aquellos resultados que queremos que la aplicación nos muestre.



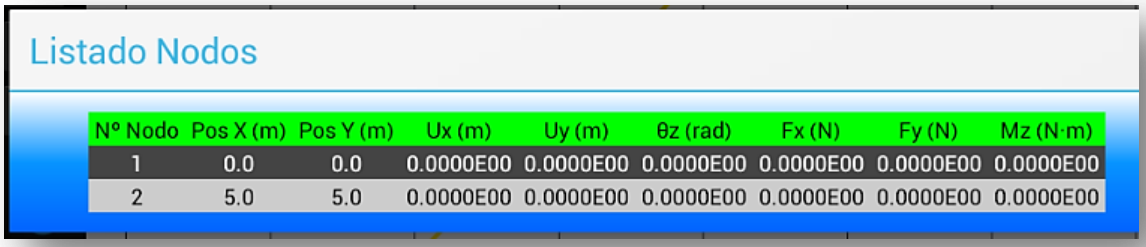
Captura 41: Menú Emergente Listado

Como los listados únicamente nos aportan información, para salir de ellos habrá que pulsar el botón *back* de nuestro dispositivo.

Vemos a continuación las diferentes opciones.

5.3.10.1 Listados de nodo

En el momento en el que pulsamos esa opción se abre el siguiente diálogo:



Captura 42: Listados de Nodo

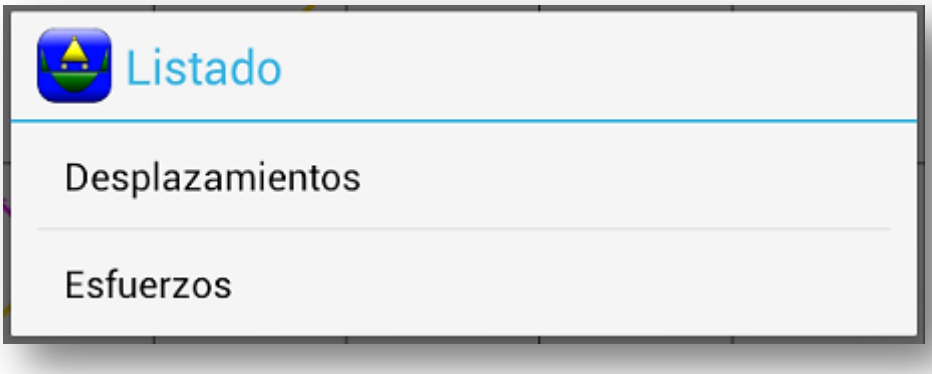
En él se muestra un listado de todos los nodos que componen la estructura y las características más importantes de los mismos.

Puede ser que los nodos no aparezcan ordenados por su número de identificación, esto no quiere decir que los resultados asociados a cada uno de ellos estén mal.

En este listado se muestra la posición inicial de cada nodo, lo que se mueve en coordenadas globales y las fuerzas que aparecen en cada uno de ellos en las diferentes direcciones.

5.3.10.2 Listados de barra

En este caso se nos abrirá un segundo menú emergente ya que tendremos dos opciones, o bien visualizar los listados de desplazamientos en barras, o bien los listados de esfuerzos.

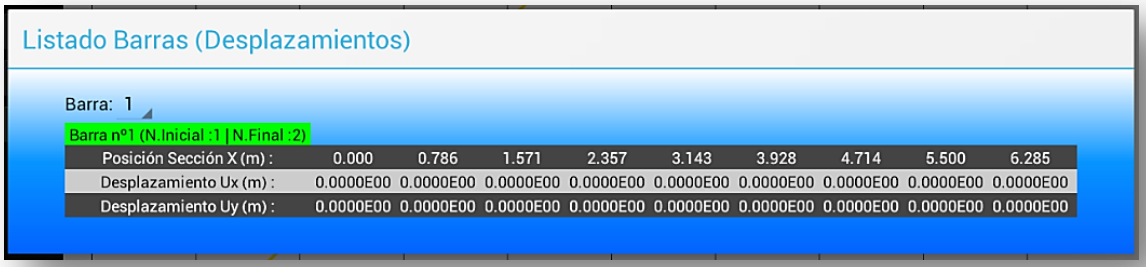


Captura 43: Menú Emergente Tipo de Listado en Barra

Vemos las pantallas que aparecerán al pulsar cada uno de ellos.

5.3.10.2.1 Desplazamientos

Se abre el siguiente diálogo:



Captura 44: Listado en Barra (Desplazamientos)

Deberemos escoger en el desplegable la barra de la cual queremos obtener los listados.

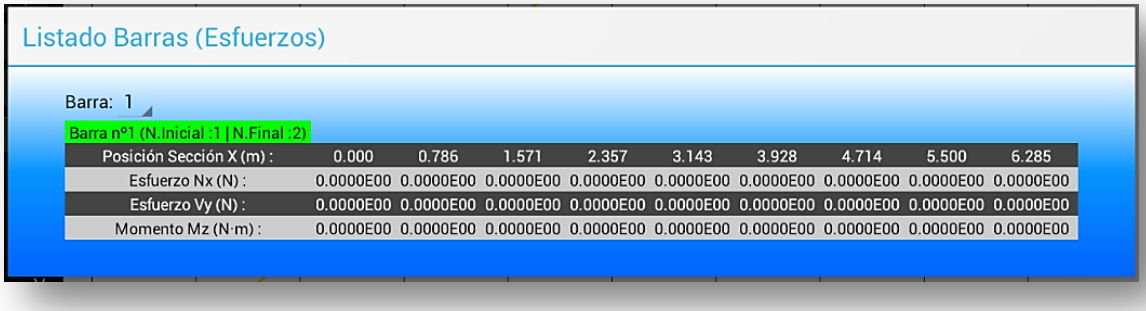
El listado nos informa de cuál es el número de identificación de barra, y también de cuales son el nodo inicial y final de la barra escogida.

Después nos indica, para diferentes secciones a lo largo de la barra, lo que se ha movido esa sección en X y en Y en coordenadas globales.

Hay que señalar, aunque ya se dijo en la explicación del menú de la aplicación, que el número de puntos a evaluar en cada barra es modificable ente 5 y 300, se debe tener en cuenta que cuanto mayor sea el número de puntos a evaluar, el tiempo en mostrar el listado será más largo, pero no excederá de 5 segundos en ningún caso.

5.3.10.2.2 Esfuerzos

Se muestra la siguiente pantalla:



Captura 45: Listado en Barra (Esfuerzos)

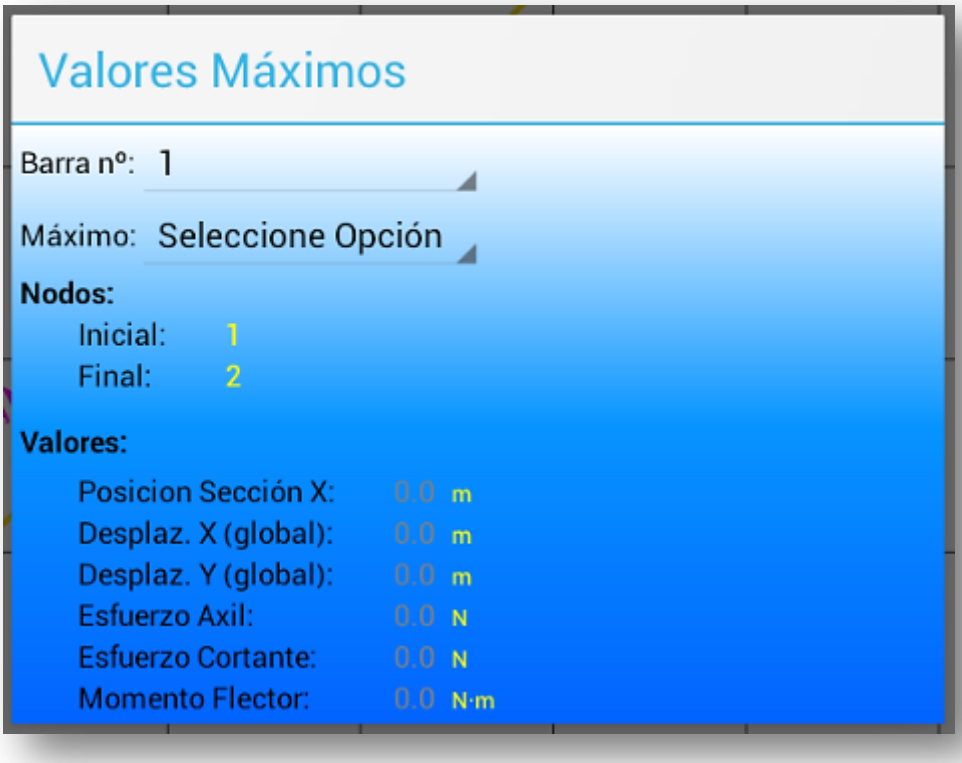
Es muy similar al anterior. Debemos escoger la barra en el desplegable. Se muestra de nuevo el número de identificación de barra, y también el nodo inicial y final que la definen.

En este caso para diferentes secciones lo que se puede visualizar es el valor del esfuerzo axil, del cortante y del momento flector.

Cuando modificamos el número de puntos de evaluación, será tanto para el listado de desplazamientos como para el de esfuerzos.

5.3.10.2.3 Valores máximos

Es el tercer tipo de listado que puede mostrar la aplicación, su apariencia será así:



Captura 46: Listado Valores Máximos

En este menú tenemos que seleccionar la opción deseada en dos listas desplegables diferentes. En la primera de ellas seleccionamos la barra sobre la que queremos obtener los valores máximos. En la segunda se escoge el tipo de máximo que se quiere buscar, por ejemplo el valor del axil máximo, del desplazamiento en X máximo, etc.

Una vez que hemos seleccionado ambas opciones este listado nos informa de cuál es el nodo inicial y final de la barra escogida para que podamos comprobar que es la barra que queremos. Después nos muestra los diferentes valores que se dan en la sección en la que se ha encontrado el máximo. El valor buscado aparecerá en rojo y el resto en color amarillo.

Aunque hayamos escogido ya un valor y una barra, el listado se adapta de forma inmediata a cambios en cualquiera de las listas desplegables.

Hay que advertir que en el listado únicamente aparecen los valores de la primera sección que se encuentra con ese valor máximo, esto quiere decir que si el valor máximo se repitiera en diferentes secciones el listado solo mostrará la primera de ellas.

5.3.11 Icono de redefinición

Es el último icono de la fila horizontal de iconos que aparecen en la parte superior de la pantalla. Ya se ha comentado anteriormente la utilidad de este icono. Al pulsarlo se activan de nuevo las opciones de recogida de datos y se desactivan las de muestra de resultados, lo podemos utilizar cuando nos damos cuenta de haber cometido algún error en la fase de introducción de datos y ya hemos pulsado el botón de calcular o cuando simplemente queremos modificar los datos de partida para comparar.

5.3.12 Pantalla gráfica

En esta parte explicaremos como la aplicación gestiona la representación gráfica de la estructura.

El dispositivo tiene unas determinadas limitaciones de tamaño de pantalla, y para evitarle al usuario la incomodidad de tener que calcular un escalado correcto para que pueda ver la estructura completa en la pantalla, la aplicación realiza este escalado automáticamente.

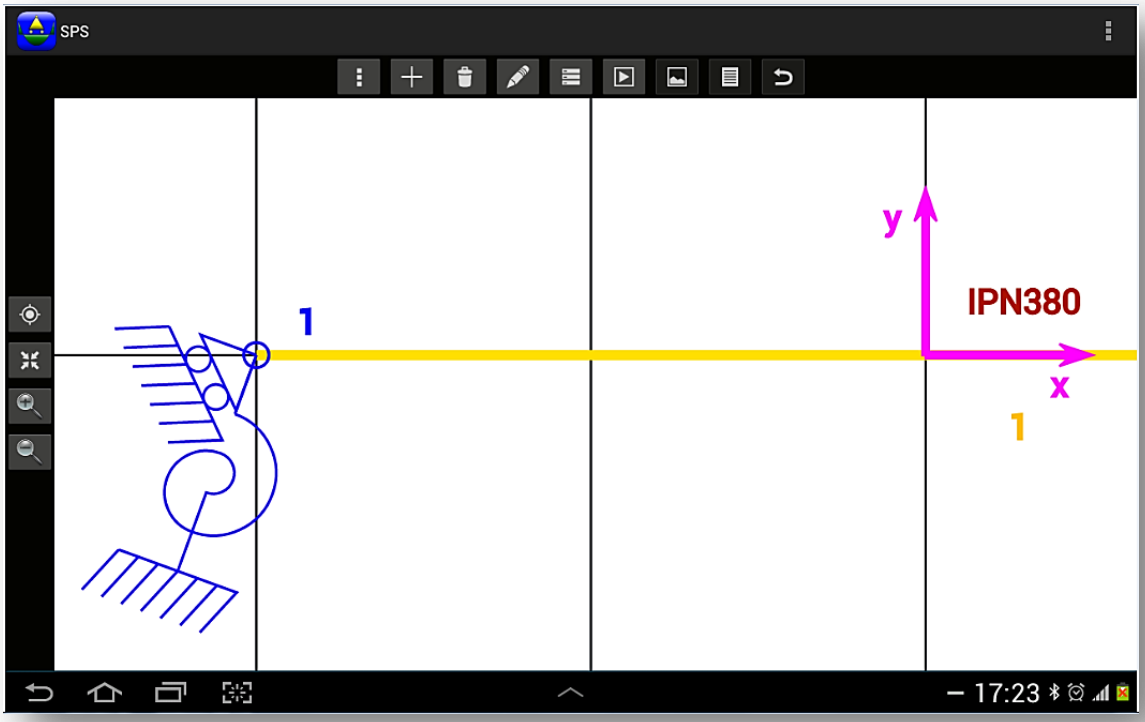
El funcionamiento es el siguiente: la estructura estará siempre comprendida en el rectángulo de la pantalla que el menú vertical y horizontal deje libre, para ello ajustará sus medidas.

También puede surgir que la estructura necesite en una determinada zona una definición muy detallada de barras o apoyos, por lo que se requiere una mejor visualización de ese área mediante un aumento o reducción. Esto se realiza mediante las opciones de Zoom.

Las opciones de Zoom están duplicadas. Es decir, el usuario puede realizar la acción deseada, manualmente pulsando el botón correspondiente o también lo puede realizar tocando con dos dedos sobre la zona deseada y separarlos para aumentar o acercarlos para reducir. Además existe la opción de desplazar la estructura en su conjunto según el deseo del usuario.

Esto se ha implementado así para poder tener un mayor control sobre el Zoom y también para facilitar el uso del Zoom en dispositivos con pantallas de pequeño tamaño. Por último comentar la existencia de un botón que restaura el zoom inicial.

Se debe señalar que la malla no es infinita, su tamaño depende del de la estructura y se ajustará a ella automáticamente en todo momento, a medida que vamos generando la estructura.



Captura 47: Ejemplo de zona aumentada y trasladada

5.4 Ejemplo práctico

El ejemplo que mostramos a continuación es el ejemplo número 6 de los apuntes de la asignatura "Teoría de Estructuras" del plan de Ingeniero Industrial. Se ha elegido este ejemplo ya que es un ejemplo sencillo pero bastante completo y a que debido al conocimiento de los resultados del problema podremos contrastarlos con los resultados calculados por la aplicación.

El esquema del ejemplo es el siguiente:

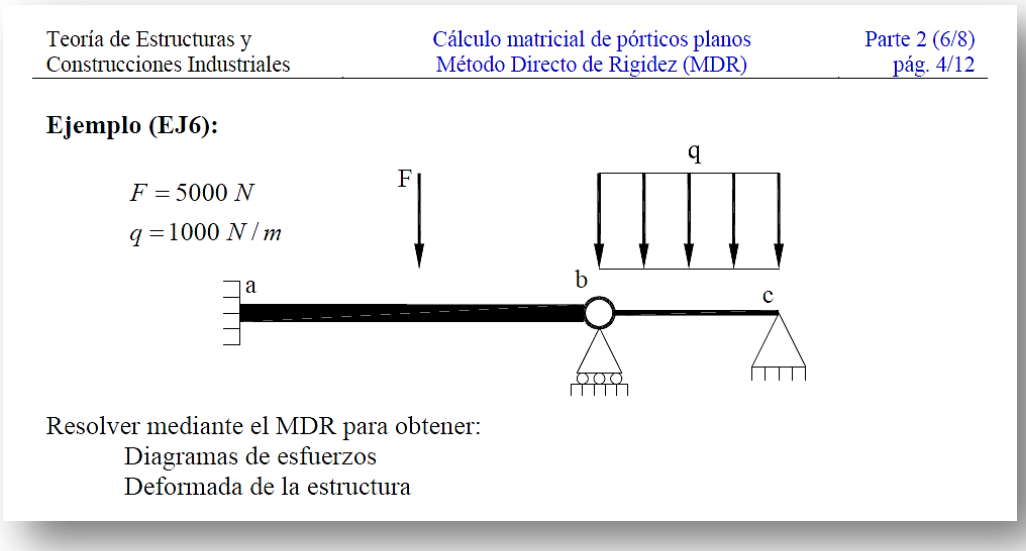


Imagen 8: Esquema Estructura obtenido de Apuntes

Los datos restantes son:

E = 2,1 x 10^{11} Pa.

Barra a-b:

L= 8 m; A = 6 x10^{-3} m^2; I = 200 x 10^{-6} m^4; h = 0.4 m.

Barra b-c:

L= 5 m; A = 4 x10^{-3} m^2; I = 50 x 10^{-6} m^4; h = 0.2 m.

5.4.1 Creación de nodos

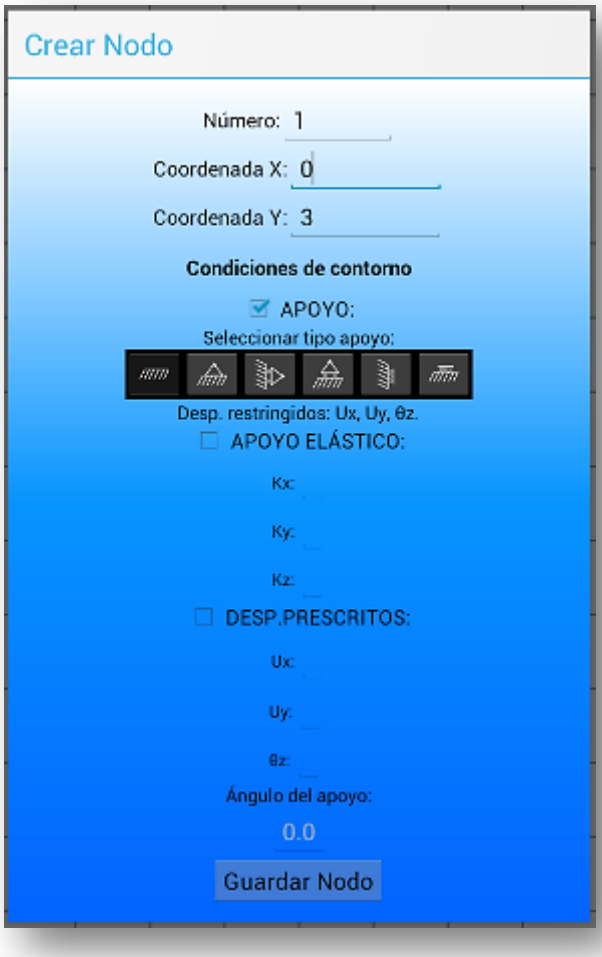
Comenzamos introduciendo los tres nodos necesarios para definir la estructura, para ello pulsamos el *Icono de creación* y más tarde la opción *nodo*.

Las características más importantes del primer nodo son:

Nº identificación	Coord.X (m)	Coord.Y (m)	Tipo apoyo
1	0	3	Empotrado

Tabla 2: Característica Nodo 1

Con lo cual el diálogo de creación de nodo debe quedar como sigue:



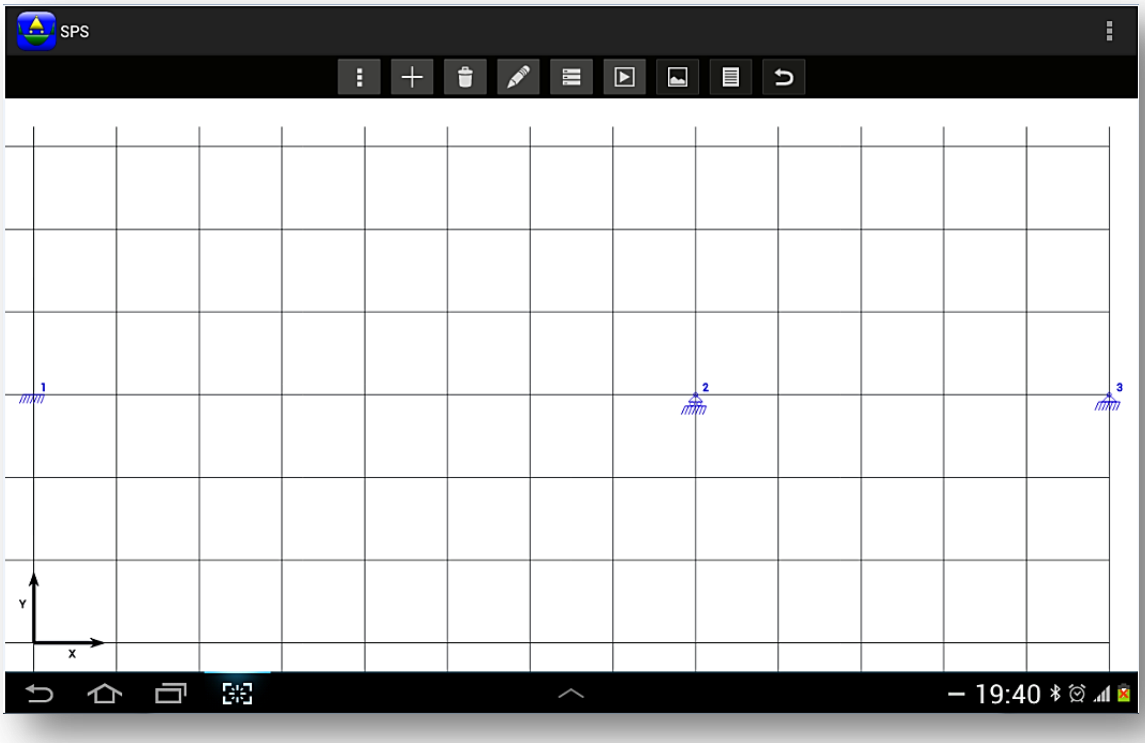
Captura 48: Creación de Nodo 1

De igual modo creamos los otros dos nodos pero con las siguientes características:

Nº identificación	Coord.X (m)	Coord.Y (m)	Tipo apoyo
2	8	3	Móvil (x)
3	13	3	Fijo

Tabla 3: Características Nodo 2 y Nodo 3

Las coordenadas de los nodos no tienen por qué ser exactamente esas, valen cualquiera que cumplan con la geometría del esquema. Lo que visualizamos hasta el momento es:



Captura 49: Visualización de los 3 Nodos

5.4.2 Creación de barras

Icono de creación > Barra

Como se trata de dos barras que no comparten características las crearemos directamente introduciendo sus datos, sin utilizar ningún tipo de biblioteca, ya que no sería útil para este ejemplo en concreto.

Lo único que debemos tener en cuenta es meter los datos en sus campos correspondientes, las características de las barras son:

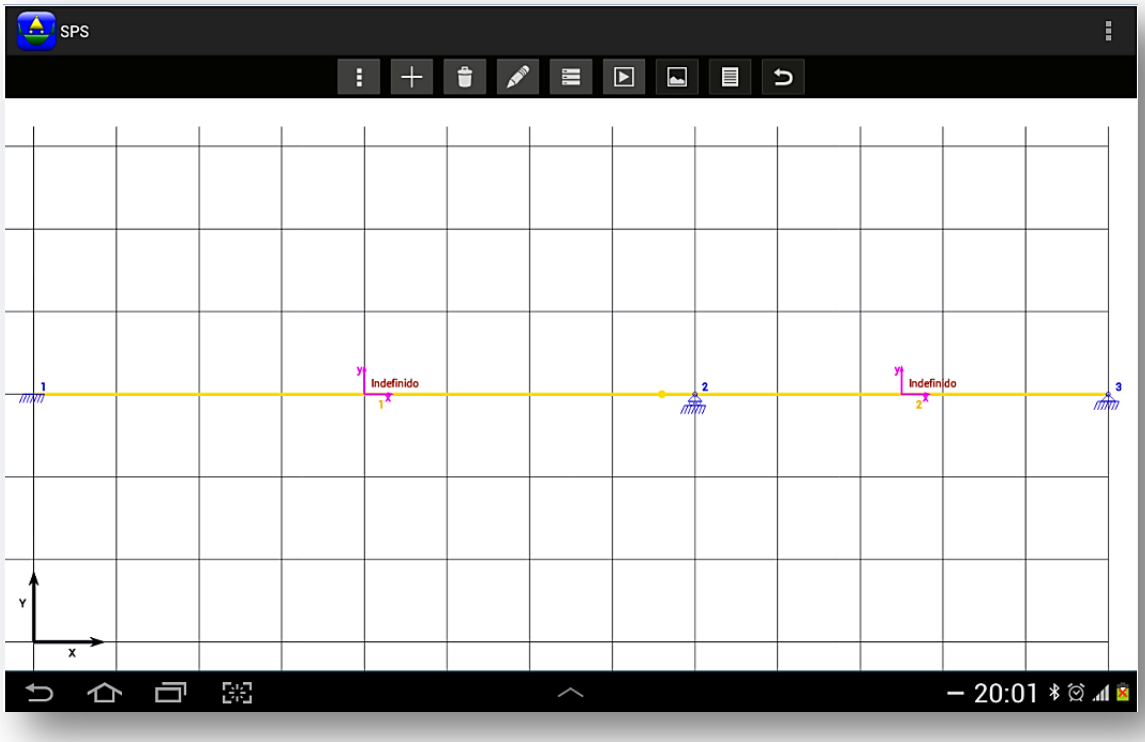
Nº identificación	Nodo inicial	Nodo final	E (Pa)	A (m^2)	L (m)	I (m^4)	h (m)
1	1	2	$2.1x10^{11}$	$6x10^{-3}$	8	$200x10^{-6}$	0.4
2	2	3		$4x10^{-3}$	5	$50x10^{-6}$	0.2

Tabla 4: Características Barra 1 y 2

Mostramos únicamente el diálogo de creación de la barra 1:

Captura 50: Diálogo de Creación de Barra 1

Señalar que se ha marcado la libertad de rotación en el extremo final de esta barra para introducir la rótula que permite el giro relativo en la unión de ambas barras, de igual manera que se introduce esta libertad en el nodo final de la primera barra se podría haber metido en el nodo inicial de la segunda, cumpliendo la misma función. Hasta ahora podemos visualizar por pantalla:



Captura 51: Visualización de Nodos Y Barras

El círculo en el extremo final de la barra 1 representa la rótula introducida.

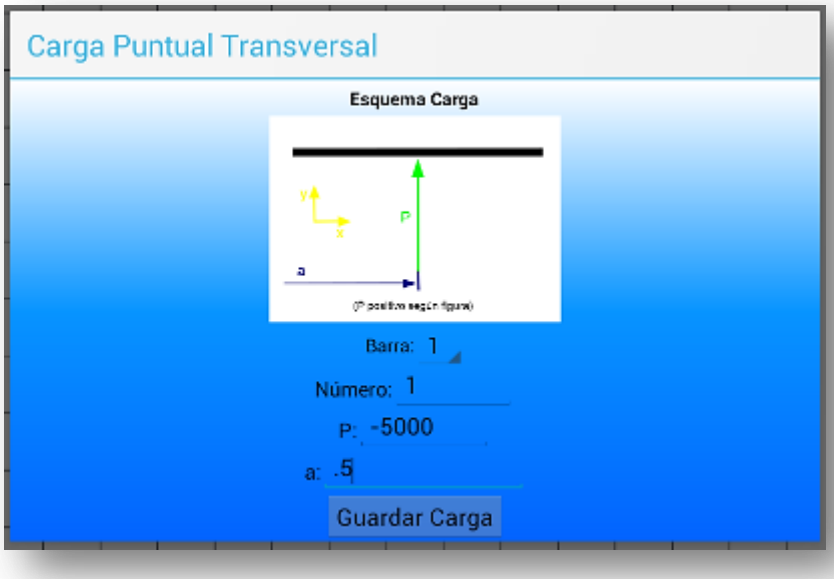
Como no hemos utilizado ningún perfil de librería, la identificación de la sección aparece como indefinida, para que no nos moleste, lo ocultaremos a partir del *menú ver* en el resto del ejemplo.

5.4.3 Creación de cargas

Como se ve en el esquema la estructura únicamente soporta dos cargas, una en cada barra.

En primer lugar generamos la carga puntual de la primera barra:

Icono de creación > Carga> Sobre Barra > Puntual transversal



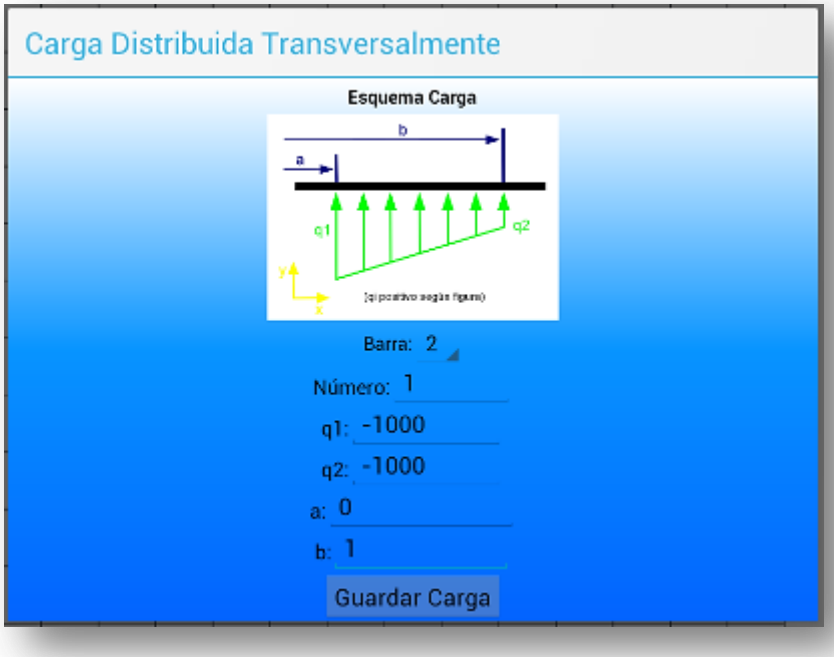
Captura 52: Diálogo de Creación de Carga Puntual Transversal

Como la carga es vertical y hacia abajo, es decir tiene el sentido negativo del eje Y, debemos poner en el valor de P = -5000.

Como está situada justo en el centro de la barra 1 el valor de a es 0.5.

De igual modo creamos la carga distribuida sobre la barra 2:

Icono de creación > Carga> Sobre Barra > Distribuida transversalmente

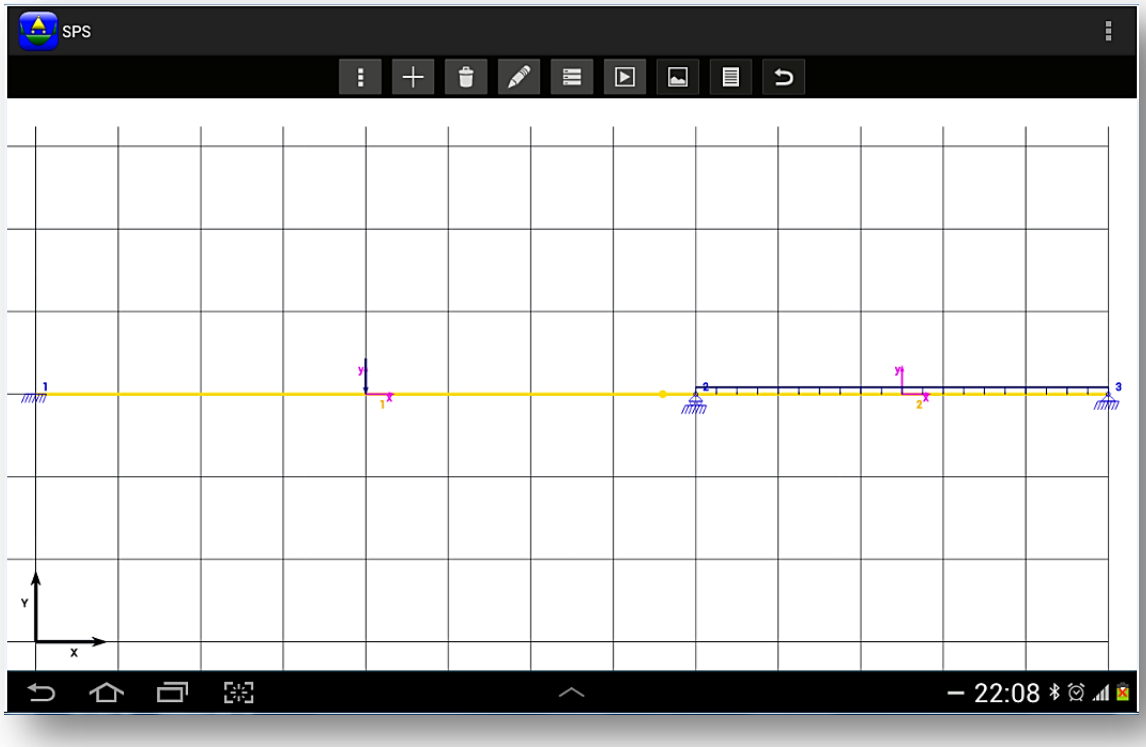


Captura 53: Diálogo de Creación de Carga Distribuida sobre Barra 2

Tenemos que cambiar el desplegable para seleccionar la barra 2. Como es una carga constante el valor de q1 y q2 será el mismo e igual a 1000, por el mismo motivo que en la carga anterior estos valores llevarán signo negativo.

Como está distribuida a lo largo de toda la barra, el valor de a es 0 y el de b, 1.

En este momento podemos ver dibujadas las cargas sobre la estructura.



Captura 54: Visualización General con cargas incluidas

Como la aplicación las dibuja a escala, vemos como la carga distribuida se pinta sensiblemente más pequeña que la puntual.

5.4.4 Comprobación de datos introducidos

Para comprobar si hemos generado correctamente toda la estructura comprobaremos la misma a partir del icono de propiedades.

Para comprobar los nodos:

Icono propiedades > Nodo

Captura 55: Propiedades del Nodo 2

Cambiando en el desplegable el número del nodo visualizaremos las propiedades correspondientes a cada uno, únicamente mostramos las del nodo 2.

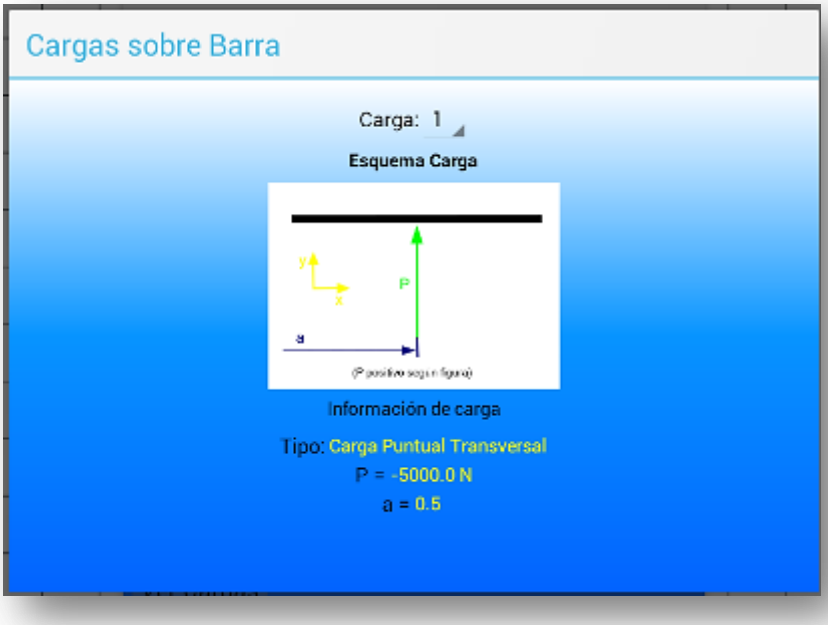
Para comprobar las barras:

Icono propiedades > Barra

Captura 56: Propiedades de la Barra 1

Solo mostramos las propiedades de barra de la barra 1.

Para ver las cargas asociadas a dicha barra, pulsamos *Ver Cargas*.



Captura 57: Cargas sobre Barra 1

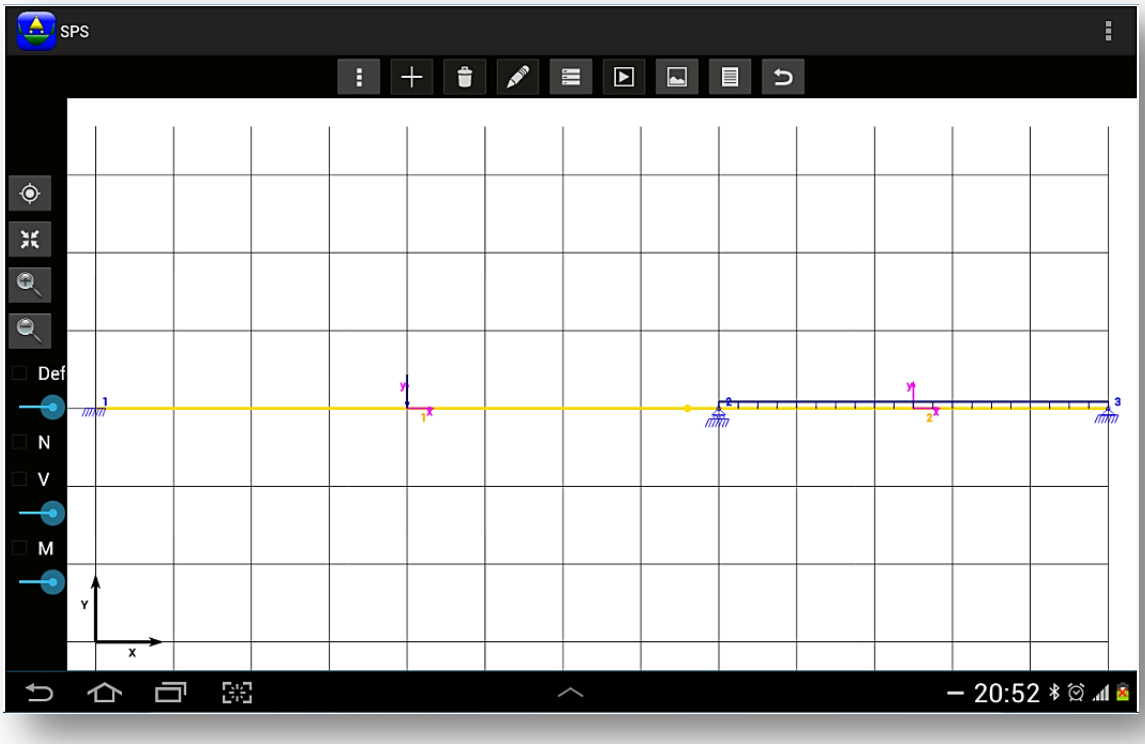
Como todos los datos introducidos son correctos no es necesario utilizar el *icono de Eliminar* ni el de *Editar*, si hiciera falta sería sencillo realizar los cambios oportunos a partir de estos dos iconos.

5.4.5 Cálculo

Para realizarlo únicamente pulsamos el *Icono de Calcular.*

5.4.6 Resultados gráficos

Para que se despliegue el menú de resultados gráficos pulsamos el icono correspondiente. En ese momento podemos seleccionar la opción que queremos visualizar y escalarla, para que se vea mejor, en este ejemplo ponemos la escala al valor máximo en todos los resultados.



Captura 58: Menú Resultados Gráficos desplegado

5.4.6.1 Deformada

La deformada se muestra marcando su check box en la pantalla inicial. Se muestra muy exagerada debido a que la escala está en su valor máximo.

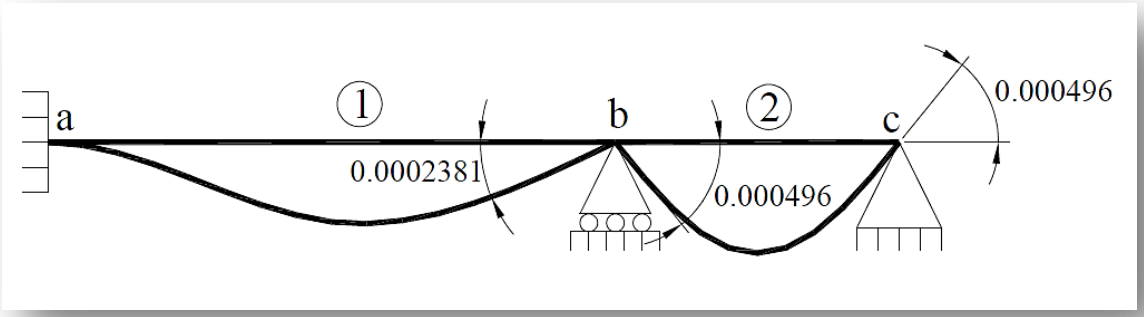
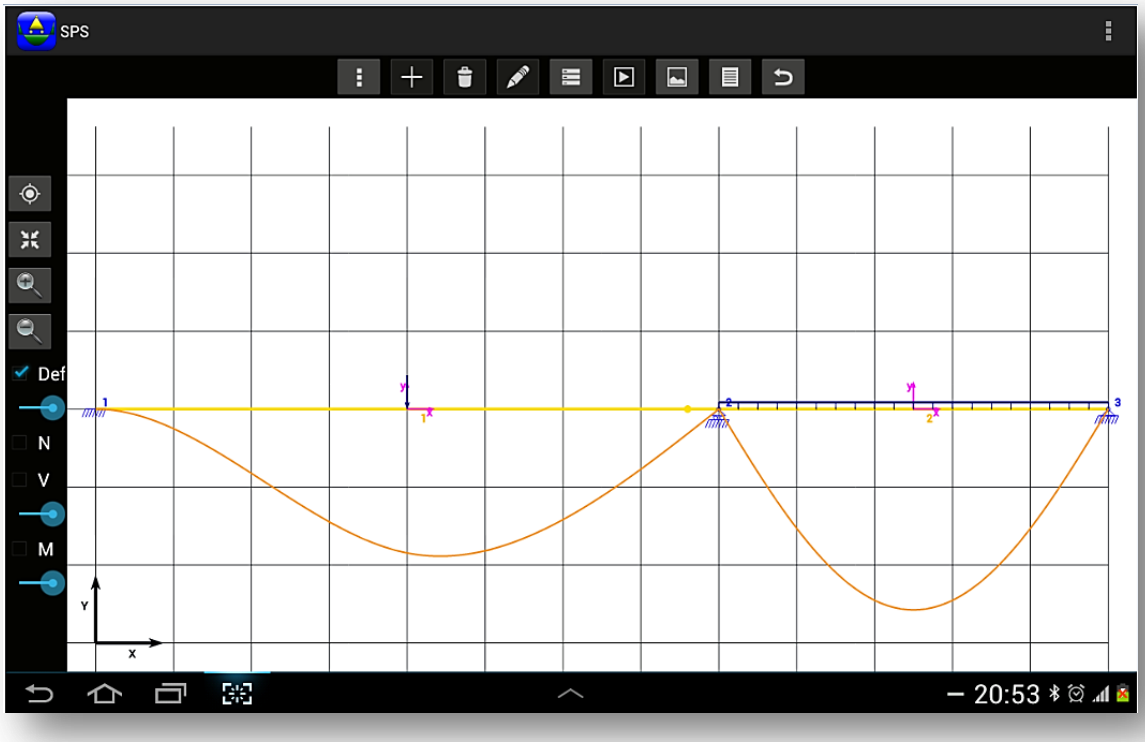


Imagen 9: Deformada Teórica según apuntes



Captura 59: Deformada obtenida en la Aplicación

5.4.6.2 Esfuerzo Axil

El valor del axil en este ejemplo a lo largo de ambas barras es cero, con lo que no se pinta su gráfico aunque marquemos la casilla correspondiente al esfuerzo axil.

5.4.6.3 Esfuerzo Cortante

Podemos comparar los gráficos que se mostraban en los apuntes con los obtenidos por la aplicación, vemos que son muy parecidos. Los valores numéricos los compararemos más tarde al comparar los listados.

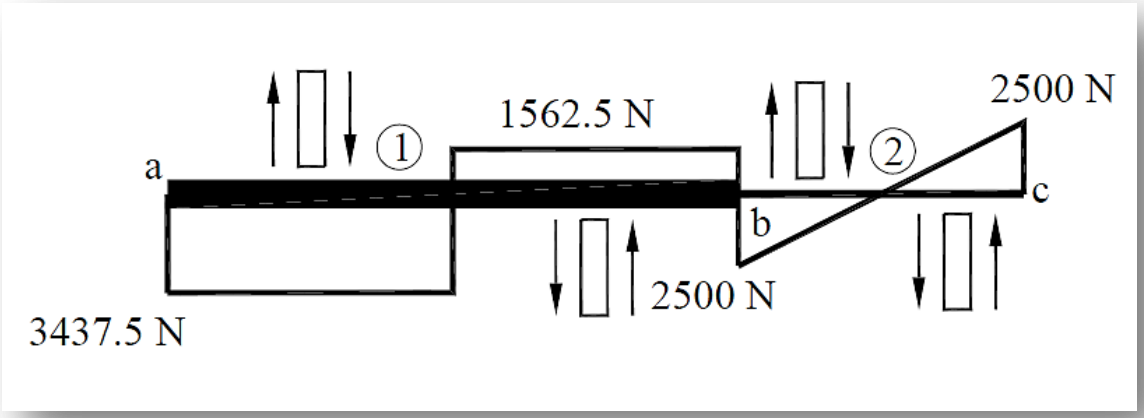
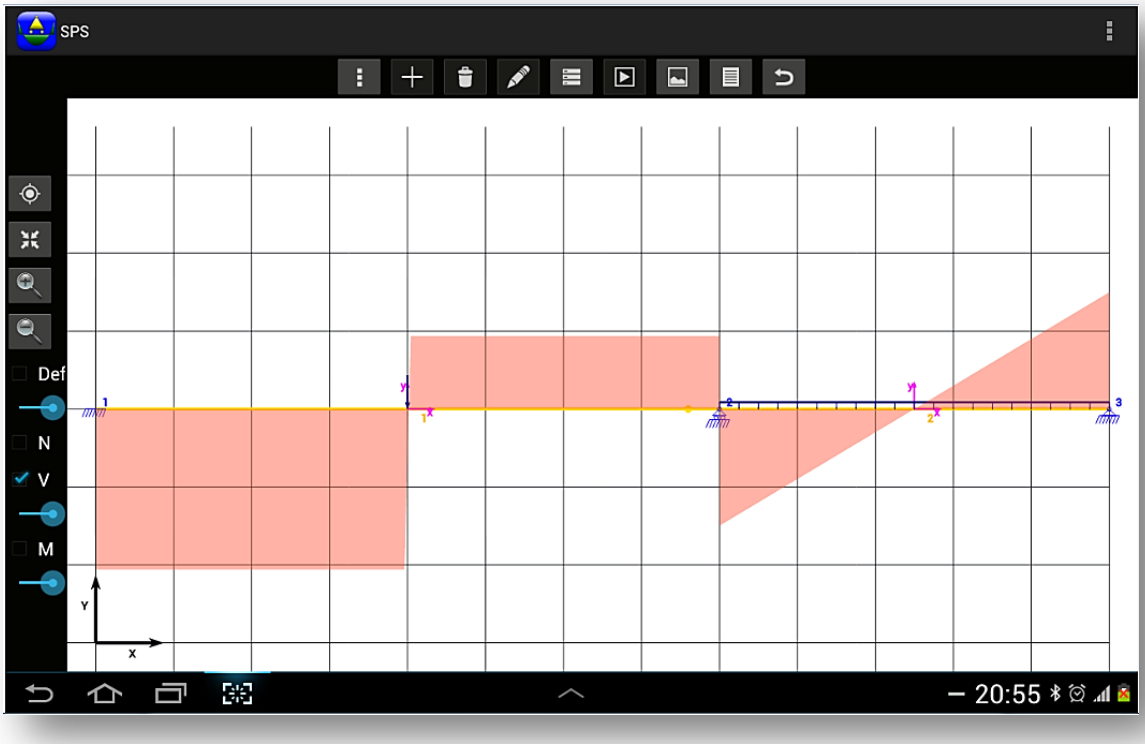


Imagen 10: Diagrama de esfuerzos Cortantes según apuntes



Captura 60: Diagrama de esfuerzos Cortantes ofrecido por la Aplicación

5.4.6.4 Momento Flector

De igual forma comparamos los gráficos de momentos flectores, en primer lugar vemos el de los apuntes y después el de la aplicación.

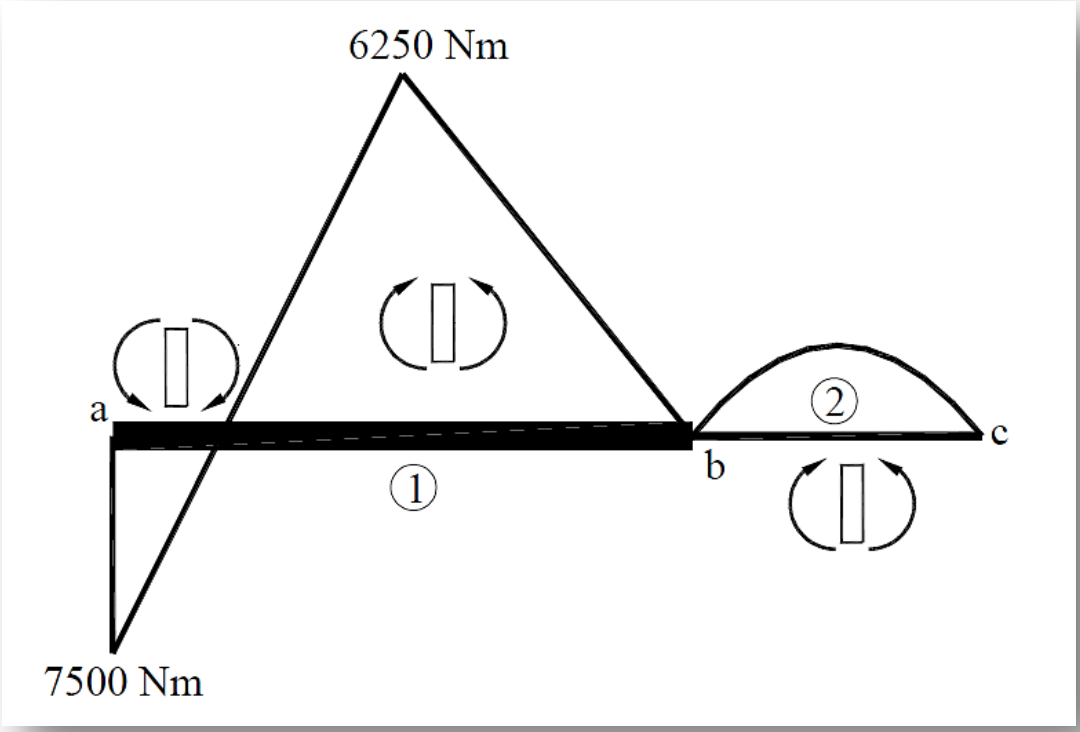
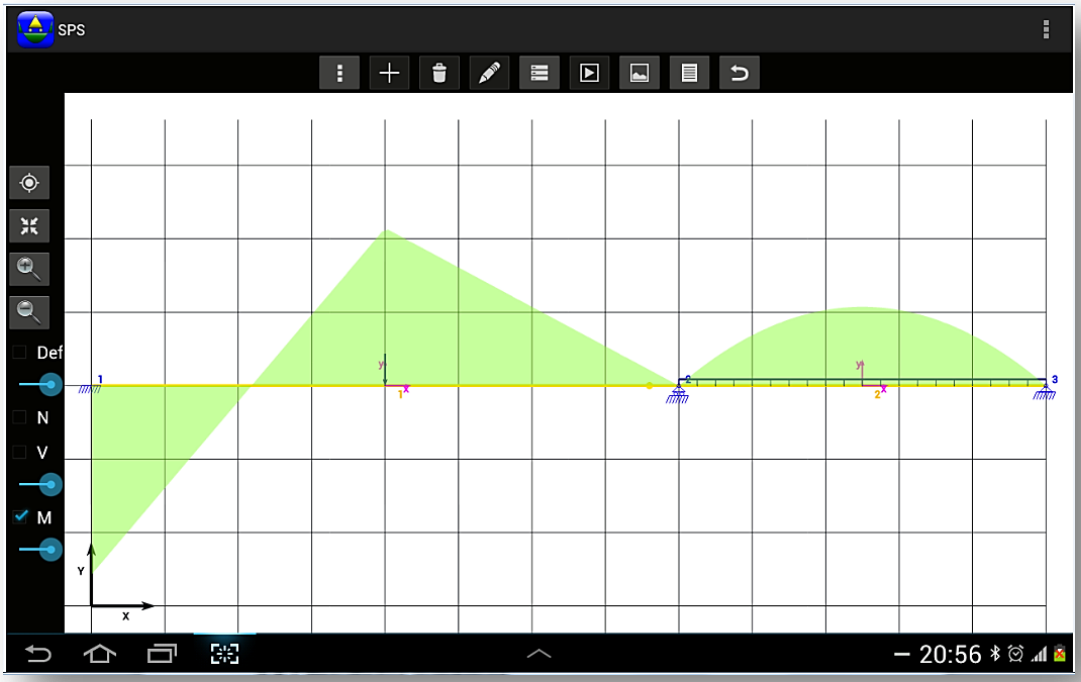


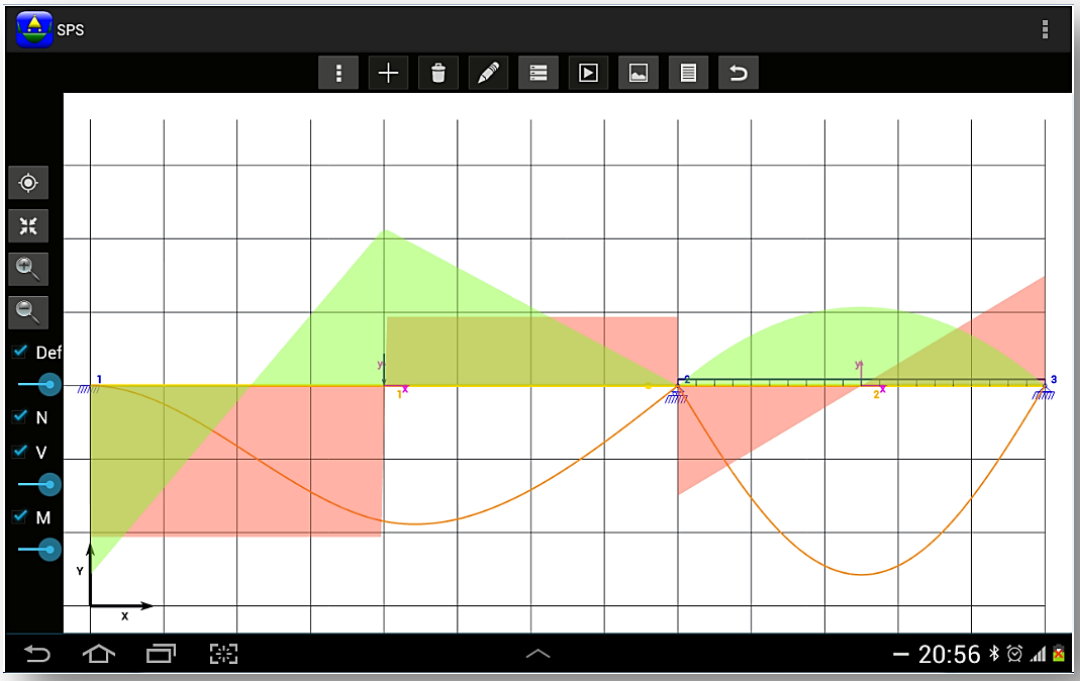
Imagen 11: Diagrama de Momentos Flectores teórico (Apuntes)



Captura 61: Diagrama de Momentos Flectores ofrecido por la Aplicación

5.4.6.5 Conjunto de gráficos

Mostramos aquí todos los resultados gráficos pintándolos simultáneamente.



Captura 62: Visualización de Resultados Gráficos

5.4.7 Resultados numéricos

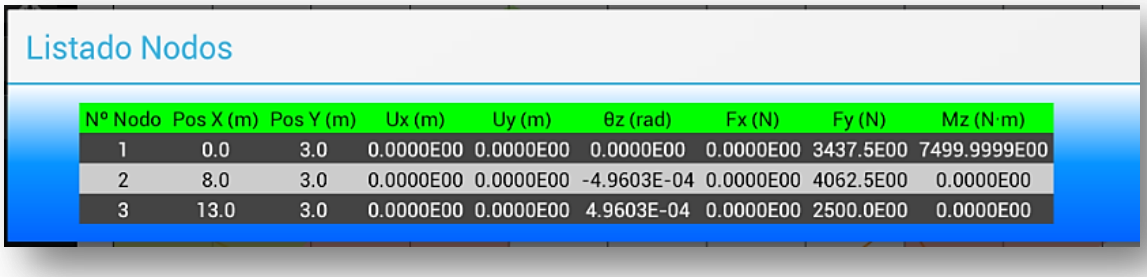
En este punto vemos los listados ofrecidos por la aplicación.

5.4.7.1 Listado Nodos

Comparamos el listado obtenido con los valores de los apuntes.

$Nº Nodo$	$Ux(m)$	$Uy(m)$	$\theta z(rad)$	$Fx(N)$	$Fy(N)$	$Mz(N \cdot m)$
1	0	0	0	0	3437.5	7500
2	0	0	-0.000496	0	4062.5	0
3	0	0	0.000496	0	2500	0

Tabla 5: Resultados teóricos

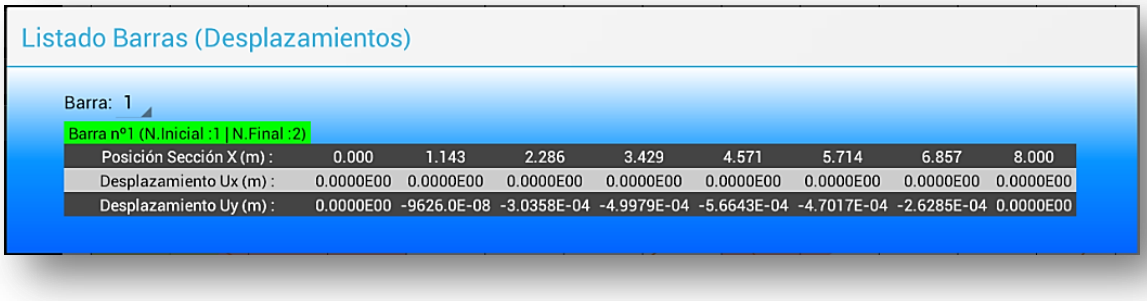


Captura 63: Listado de Nodos

Vemos que los resultados coinciden perfectamente, destacar, que si la rótula la hubiéramos colocado en la segunda barra en lugar de en la primera obtendríamos el valor del giro en el nodo 2 correspondiente a la barra 1, que, al tener la rótula, es distinto al del nodo 2 en la barra 2 que es el que se muestra en el listado.

5.4.7.2 Listado Barras (Desplazamientos)

Estos listados nos sirven para obtener valores en diferentes secciones de las barras, no podemos comparar con valores de los apuntes, ya que en estos no se ofrecen, pero si podemos ver su correspondencia con lo que se muestra en los diagramas que vimos anteriormente, hemos puesto para el ejemplo pocos puntos de evaluación para que se vieran todos en la captura de la pantalla, pero como se dijo anteriormente, podemos evaluar hasta un máximo de 300 puntos por barra.

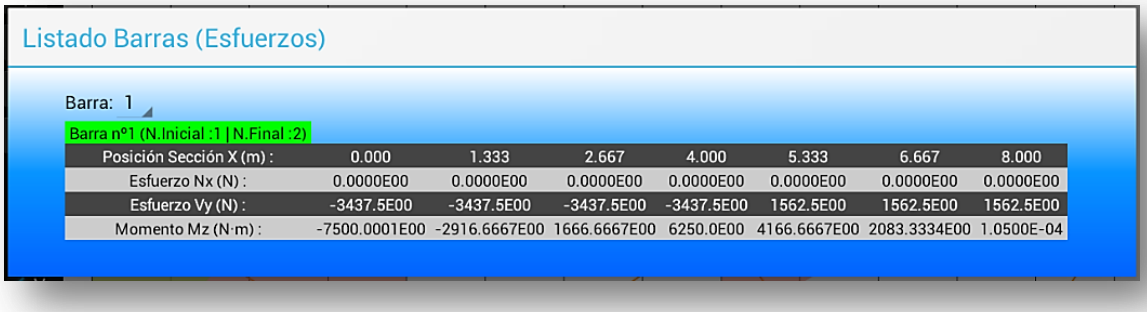


Posición Sección X (m) :	0.000	1.143	2.286	3.429	4.571	5.714	6.857	8.000
Desplazamiento Ux (m) :	0.0000E00	0.0000E00	0.0000E00	0.0000E00	0.0000E00	0.0000E00	0.0000E00	0.0000E00
Desplazamiento Uy (m) :	0.0000E00	-9626.0E-08	-3.0358E-04	-4.9979E-04	-5.6643E-04	-4.7017E-04	-2.6285E-04	0.0000E00

Captura 64: Listado de Barras (Desplazamientos)

Lo mismo ocurrirá con el resto de listados, únicamente mostraremos los listados ofrecidos por la aplicación.

5.4.7.3 Listado Barras (Esfuerzos)



Posición Sección X (m) :	0.000	1.333	2.667	4.000	5.333	6.667	8.000
Esfuerzo Nx (N) :	0.0000E00	0.0000E00	0.0000E00	0.0000E00	0.0000E00	0.0000E00	0.0000E00
Esfuerzo Vy (N) :	-3437.5E00	-3437.5E00	-3437.5E00	-3437.5E00	1562.5E00	1562.5E00	1562.5E00
Momento Mz (N·m) :	-7500.0001E00	-2916.6667E00	1666.6667E00	6250.0E00	4166.6667E00	2083.3334E00	1.0500E-04

Captura 65: Listado de Barras (Esfuerzos)

5.4.7.4 Listado Valores Máximos

Para este caso, podemos obtener diferentes listados en función de la posición en que coloquemos los diferentes desplegables, como ejemplo se muestra el valor máximo del flector en la barra 1.

Captura 66: Listado de Valores Máximos

6 GOOGLE PLAY

En todos los dispositivos que cumplen los requisitos impuestos por Google, existe una aplicación llamada Google Play. Esta aplicación nos permite acceder directamente a la tienda de Google. En ella podemos encontrar películas y/o aplicaciones de manera gratuita o pago de los desarrolladores de cualquier parte del mundo.

Esta aplicación permite clasificar y buscar las aplicaciones por categorías y tipo de dispositivo.

La búsqueda de una aplicación concreta es muy sencilla, y responde a la filosofía creada por Google con su buscador. Una búsqueda por título de aplicación mediante un motor de búsqueda o por el contrario si se desconoce el mismo, una búsqueda por categoría, las más descargadas tanto gratuitas como de pago, o las seleccionas por el personal de Android. En definitiva una navegación amigable.

Pero nuestro interés no reside en como descargar o comprar aplicaciones sino que desde el punto de vista de desarrolladores de aplicaciones queremos dar a conocer nuestro trabajo, para ello debemos dar una serie de pasos antes de que nuestra aplicación esté disponible para el resto del planeta.

6.1 Aplicación en Google Play

El procedimiento para subir una aplicación a Google Play requiere de varios pasos, pero antes de nada deberemos realizar una serie de acciones en nuestra aplicación para asegurarnos de que nuestro producto es de calidad.

6.1.1 Pasos previos

6.1.1.1 Registros de logs

Más conocido por los desarrolladores como logging, básicamente se trata de instrucciones superfluas que pueden estar en todo el código fuente y sirven al desarrollador para verificar tareas, conexiones con servidores remotos, cadenas de texto con información sobre rutinas de cifrado. Aunque todas estas entradas y sus correspondientes salidas por el log resultan útiles durante la fase de depuración, se pueden tornar en fallos importantes en la versión de comercialización.

6.1.1.2 Notificaciones del depurador

Existe una determinada notificación instantánea temporal conocida como *Toast*, que se suele utilizar por los desarrolladores para verificar los ítems seleccionados en un menú, o para obtener información cuando se producen errores. Por lo tanto, al igual que lo anterior en la versión para clientes, estos elementos no deben aparecer, o deben estar muy controlados

6.1.1.3 Datos de ejemplo

.Podremos comercializar la aplicación con datos de ejemplo. De ser así, estos deben estar realizados de forma clara e intuitiva para que el usuario pueda hacer un buen uso de ellos sin necesitar excesivo tiempo para su comprensión. Deberemos evitar filtrar datos personales o bancarios.

6.1.1.4 AndroidManifest.xml

Mención aparte requiere el archivo AndroidManifest.xml, antes de publicar nuestra aplicación debemos verificar los siguientes puntos:

Eliminar la etiqueta *android: debuggable* o, al menos, fijar su valor en false. Esto se fija, para evitar que la versión para el cliente pueda depurarse (hacer debug).

Definir valores apropiados para los atributos *label* e *icon* de la etiqueta *application*. El texto debe ser lo más corto posible, facilitará su búsqueda y comercialización.

Especificar los atributos *android: versionCode* y *android: versionName* en el elemento <application> del archivo AndroidManifest.xml. El atributo *versionCode* es un valor entero que se puede comprobar mediante código y que se suele incrementar tras cada nueva versión. Por su parte, *versionName* se muestra a los usuarios.

Especificar el nivel mínimo del SDK aceptable para nuestra aplicación. En caso de que nuestra aplicación no esté limitada por requerimiento alguno, lo más aconsejable es fijar el valor más bajo posible, de forma que sea accesible para el mayor número posible de dispositivos.

6.1.1.5 Acuerdo de licencia del usuario final

Es aconsejable que proporcionemos un acuerdo de licencia del usuario final (EULA o End User License Agreement) con nuestras propias condiciones, aun cuando la mayoría de los usuarios lo ignorará completamente. Debemos, aun así, tratar de defender la propiedad intelectual de nuestra creación, así como los intereses de posibles socios e inversores por todos los medios legales posibles.

A la hora de mostrar esta EULA, lo más común es mostrarla por primera vez al iniciar la aplicación y exigir una confirmación por parte del usuario de que está de acuerdo con sus términos y condiciones. Pasado esta verificación no deberíamos mostrarlo más veces, salvo en el caso de que el usuario así lo requiera en la correspondiente opción del menú.

6.1.1.6 Realización de prueba

Todos los desarrolladores su3elen testear su aplicación en dispositivos reales antes de ponerla en el mercado. Hay desarrolladores que prefieren hacerlo una vez concluido todo el desarrollo, pero no tienen sentido a no ser que seas un desarrollador muy experimentado, pues la cascada de errores que pueden aparecer puede

sobrepasarte fácilmente. Por otro lado, el proceso más aconsejable, es ir testeando por bloques o tareas de la aplicación y así poder localizar con mayor facilidad los errores.

Esta etapa es obligatoria para cualquier aplicación sea del tipo que sea, ya que se debe someter a cualquier situación posible de casuísticas, verificando que la aplicación se degrada adecuadamente cuando algún servicio requerido no esté disponible. Los mensajes en estos casos deben ser claros y orientativos para los usuarios. Hay que tener especial cuidado en los cierres y reinicios de la aplicación, así como cuando esté disponible el cambio de orientación del dispositivo.

Una vez hemos comprobado toda esta serie de puntos, el siguiente paso es firmar digitalmente la aplicación.

6.2 Firma digital de la aplicación

En primer lugar vamos a explicar las principales razones por las que realizamos la firma digital de nuestra aplicación:

- Como medida de seguridad y como requisito de garantía.
- Para que de esta forma sólo nosotros podamos modificar y actualizar nuestra aplicación.
- Para poder distribuir e instalar nuestra aplicación sin problemas.
- Porque es un requisito que nos pide Google para subir nuestras aplicaciones a su plataforma de venta.

Como dice el último de los puntos anteriores, la plataforma Android requiere que el archivo de cualquier aplicación, es decir, el archivo nombredeaplicacion.apk, incorpore una firma digital para que se pueda ejecutar tanto en un dispositivo como en un emulador. Sin esta firma, la aplicación sencillamente no se abrirá.

La mayoría de entornos de desarrollo se encargan de firmarla con una clave de depuración obtenida automáticamente. Por ello este proceso de firma es transparente para la mayoría de desarrolladores hasta el momento de publicarla.

Cuando se publica una aplicación para su distribución, debe ir firmada con una clave única, diferente de la de depuración que se utilizó en el desarrollo. Por fortuna, las aplicaciones se pueden "auto-firmar", es decir, que no es necesario contar con la participación de una autoridad certificadora. Esto disminuye en forma considerable la complejidad y los costes asociados, especialmente cuando los comparamos con los de otras plataformas móviles. Además, en este proceso también nos suele ayudar el entorno de desarrollo.

El IDE nos ayudará a realizar este firmado utilizando las siguientes herramientas:

- Keystores. Almacenamiento de claves, se encuentra en una carpeta con extensión .android dentro del directorio de inicio del desarrollador. A la hora de distribuir las aplicaciones en Google Play, o donde fuese, se

hace necesario crear y almacenar una nueva clave en un keystore separado que el desarrollador podrá denominar a su gusto.

- Keytool. Sirve para crear una clave privada autofirmada en el keystore. Todo esto se hace a través de ventanas de diálogo del propio software de desarrollo, lo que facilita mucho la tarea. En este proceso se pide un alias, una contraseña, información de tipo organizativo y el periodo de validez, esté debe ser superior a 25 años, que es lo mínimo que recomienda Android.

- Jarsigner. Esta herramienta lleva a cabo la firma digital propiamente dicha. Para firmar una aplicación, debemos exportarla como un archivo .APK sin firmar e introducir la clave creada con Keytool y que se encuentra almacenada en Keystores.

Tras esto la aplicación se encuentra correctamente firmada y lista para su instalación en un dispositivo, esto se puede hacer instalando el archivo .apk directamente o difundirla a través de Google Play.

6.3 Publicación en Google Play

Es una de las mejores opciones para hacer llegar nuestra aplicación al mayor número posible de usuarios, para ello debemos realizar tres pasos:

- Crear una cuenta en Google Play registrándonos como desarrolladores. Esta cuenta de usuario será similar a cualquier cuenta con la salvedad de que debemos pagar una cuota inicial única de 25$. Este hecho, no es único en Android, Apple también requiere un firmado de aplicaciones más estricto y además una cuota anual para los desarrolladores de $100 (79,41 €) anuales, estas cuotas se deben a un requisito para evitar spam y garantizar una cierta calidad en las aplicaciones.

- Crear una cuenta de Google Wallet para poder realizar las transacciones monetarias pertinentes, por ejemplo el pago de la cuota inicial antes mencionada.

- Iniciar sesión e introducir la información necesaria asociada a la aplicación y a nosotros mismos a través de la consola de desarrolladores de Google. Entre la información solicitada se encuentran nombre de desarrollador, empresa, sitio web, capturas de pantalla de la aplicación, tipo de aplicación (pago o no), resumen y descripción de la aplicación, icono en alta definición de la aplicación, etc.

Cumplidos todos los requisitos, el mundo entero podrá disfrutar de la aplicación.

6.4 Estudio de competencia

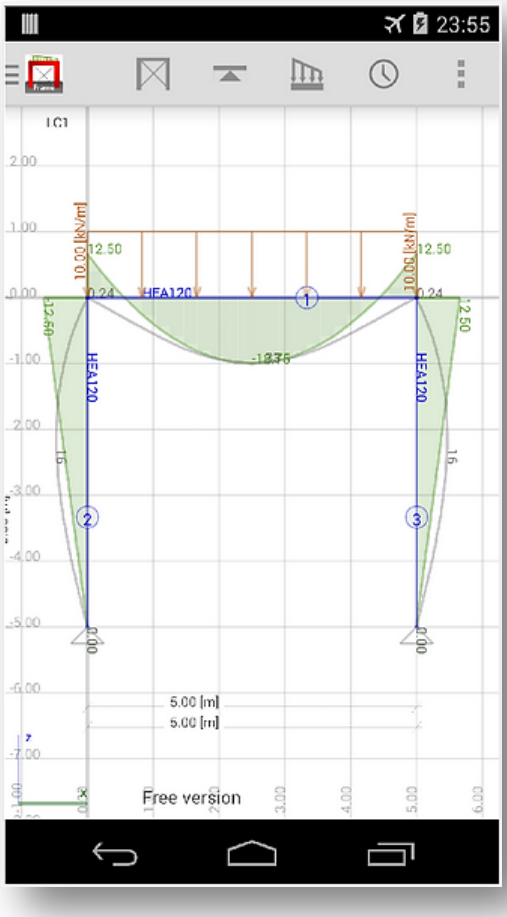
Tras realizar una búsqueda en Google Play y hacer un estudio de las diferentes aplicaciones que pueden plantear una competencia directa a nuestra aplicación Android, aparecen como destacadas las dos siguientes:

6.4.1 Frame Design 2D

Está disponible gratuitamente. Es una app de análisis estático y lineal de estructuras en 2D, permite analizar reacciones, esfuerzos y deformaciones. Está basada en el método de los elementos.

Seguramente se trate de la aplicación más completa existente ahora mismo en este campo. Las principales características son:

- Momentos, cortantes, axiles, deformaciones.
- Cargas F, M y q (rectangular y triangular).
- Uniones empotradas y articuladas en barras.
- M, N, V, ux, uz, utot, phi, sigma y en varias unidades.
- Apoyos deslizantes, fijos y empotrados en cualquier dirección.
- Hipótesis, combinaciones de cargas y coeficientes de seguridad.
- Grupos de perfiles; cada grupo puede llevar un color diferente.
- Importa perfil de acero o sección desde librería.
- Exportar a nuestras Concrete Design y Steel Design apps.
- Exportar a imagen o documento Word .doc.
- Función deshacer.
- Multicopia.
- Escalar la estructura completa.
- Acotaciones automáticas.
- Numeración de barras y nudos.
- Edición modo gráfico y modo texto.
- Unidades métricas e imperiales.
- Zoom táctil.
- Muchas opciones para mostrar y ocultar elementos.
- Opciones para definir casi todos los colores en la app.
- Fondo claro u oscuro.
- Disponible en 10 idiomas.
- Guardar las estructuras en la nube.



Captura 67: Captura de aplicación Frame Design 2D

6.4.2 EpicFEM

También es una aplicación basada en el método de los elementos finitos. Destaca la elaboración de memorias de cálculo de manera rápida y simple para compartir en clouds, compañeros, alumnos, etc.

Características:

- Apoyos simples y fijos.
- Muelles lineales y rotacionales.
- Cargas: Nodales (Fuerzas y momentos), distribuidas, triangulares.
- Biblioteca de secciones, también se puede personalizar la sección.
- Diagramas: Desplazamiento, giro, cortante, flector.
- Reacciones.
- Paso a PDF y guardado en mismo formato en SD-card.
- SI y también sistema imperial de unidades.

Captura 68: Captura de aplicación EpicFEM

7 ESTUDIO ECONÓMICO

Para poder cuantificar de alguna manera el trabajo realizado en el presente proyecto se realiza un breve estudio económico en el que se tiene en cuenta tanto las herramientas utilizadas para el desarrollo de la aplicación como la mano de obra depositada en la misma.

Como el propósito de la aplicación no es venderla a terceros, si no hacer que llegue al máximo número de usuarios, y las aplicaciones que harán la competencia a la nuestra son gratuitas, no tiene sentido interpretar este estudio como un objetivo a superar para obtener un beneficio económico de la aplicación.

El estudio llevado a cabo se resume en la siguiente tabla:

Denominación	**Ud**	**Cantidad**	**Precio (€)**	**Subtotal (€)**
Ordenador	U	1	739	739
Tableta	U	1	252	252
Desarrollo aplicación	h	319	30	9570
Consumo energía	U	1	200	200
Material Oficina	U	1	137	137
Registro Google Play	U	1	18.06	18.06
			Agregado	10916.06
			5% gastos generales	545.80
			Total	**11461.86**

Tabla 6: Estudio Económico

Por lo tanto el coste que nos ha supuesto el desarrollo de nuestra aplicación es de *ONCE MIL CUATROCIENTOS SESENTA Y UN EUROS CON OCHENTA Y SEIS CÉNTIMOS.*

El porcentaje añadido en gastos generales hace referencia a diferentes consumos que no se han tenido en cuenta en el estudio económico general de la aplicación, como por ejemplo la conexión a internet.

También se ha incluido en el estudio la adquisición de una Tablet con sistema operativo Android, si bien no es un elemento indispensable para el desarrollo de la aplicación ya que el entorno de desarrollo cuenta con variedad de emuladores en los que podemos ir probando nuestro trabajo, si es un elemento muy útil ya que reduce tiempos de prueba de la aplicación de una manera considerable (los emuladores son sensiblemente más lentos), y también es muy importante poder probar la aplicación en dispositivos reales antes de lanzarla al mercado.

8 OBJETIVOS ALCANZADOS Y VÍAS DE FUTURO

Al inicio del presente proyecto nos propusimos una serie de objetivos que en este momento analizaremos.

8.1 Objetivos alcanzados

- Conocer las principales características del lenguaje Java. Sin duda este objetivo ha sido alcanzado, pues sin tener un previo conocimiento básico del lenguaje Java nos habría sido imposible programar la aplicación en Android.

- Conocer las principales características de Android. Al igual que en el punto anterior, si no se hubiera estudiado y entendido las principales características de Android, así como su funcionamiento, llevar a cabo este proyecto con el desarrollo de la aplicación habría sido imposible, con lo que este objetivo también se ha alcanzado.

- Estudiar el entorno de desarrollo de Android. Como se ha descrito en puntos anteriores, en un principio se pudo elegir entre Eclipse y Android Studio como entorno de desarrollo para Android, ya explicamos por qué finalmente nos decantamos por Android Studio. Tras su elección, fue necesario estudiarlo, entenderlo y manipularlo con la suficiente soltura que nos permitiera el correcto desarrollo de la aplicación. Por lo que este objetivo también se cumplió.

- Desarrollar la aplicación para Android. Este objetivo también se ha cumplido en el presente proyecto ya que el producto del proyecto ha sido la aplicación Android, que funciona perfectamente y ha sido testeada en diferentes dispositivos para asegurar su correcto funcionamiento.

Por lo que todos los objetivos que nos propusimos al inicio del proyecto han sido alcanzados con éxito.

8.2 Vías de futuro

Aunque si se han alcanzado los objetivos propuestos, sin duda existen aspectos que pueden ser mejorados para que nuestra aplicación sea mucho más útil y atractiva para el usuario.

- Complementar la entrada de datos mediante la posibilidad de introducción táctil de los mismos, ya sea para la definición gráfica de nodos y barras, como para cargas.

- Ampliar aún más las posibilidades a la hora de introducción de datos, como por ejemplo la inclusión de errores de ejecución en la definición de la estructura.

- Incluir un módulo en la aplicación que realice el cálculo de tensiones normales y tangenciales.
- Aumentar la capacidad de la aplicación y módulo de cálculo para realizar más cálculos como pandeo, cálculo plástico, combinación de acciones en edificación, etc.
- Exportación de resultados a un formato de texto, incluyendo gráficos.
- Mejora en el diseño gráfico del interfaz de Usuario para hacer la aplicación más llamativa y por tanto más apetecible para el cliente.

9 BIBLIOGRAFÍA

- *DEITEL, HARVEY M. Y DEITEL, PAUL J.. Cómo programar en Java. 5ª Edición. México, Ed. Pearson Educación, 2004.*
- *http://www.android.com/*
- *http://developer.android.com/intl/es/sdk/installing/studio.html*
- *http://www.slideshare.net/dcastacun/manual-programacin-android*
- *http://www.slideshare.net/jimezam/introduccin-a-la-plataforma-j2me*
- *http://androidayuda.com/2013/07/31/la-fragmentacion-de-android-problema-o-ventaja/*
- *http://es.wikipedia.org/wiki/Android*
- *http://developer.android.com/intl/es/about/dashboards/index.html*
- *http://www.slideshare.net/SantiagoSolis1/fundamentos-del-desarrollo-de-aplicaciones-para-android*
- *http://www.carlosrobles.com/blog/2014/02/eclipse-o-android-studio/*
- *http://www.xatakamovil.com/mercado/desarrollo-de-aplicaciones-moviles-i-asi-esta-el-mercado*
- *http://barraespaciadora.com/2013/05/10/comparativa-de-sistemas-operativos-para-smartphones/*
- *http://es.wikipedia.org/wiki/Sistema_operativo_m%C3%B3vil#Symbian*
- *http://gs.statcounter.com/#mobile_os-ww-quarterly-201201-201201-map*
- *http://androideity.com/2011/07/04/arquitectura-de-android/*
- *http://developer.android.com/intl/es/distribute/googleplay/publish/index.html*
- *http://androideity.com/2011/08/25/%C2%BFcomo-firmar-aplicaciones-android/*