
Universidad de Valladolid

# ESCUELA DE INGENIERÍA INFORMÁTICA (SEGOVIA)

# GRADO EN INGENIERÍA INFORMÁTICA DE SERVICIOS Y APLICACIONES

# BITS DE INTELIGENCIA PARA ANDROID

ALUMNO: DAVID JIMÉNEZ BERMEJO

TUTOR: LUIS IGNACIO SEBASTIÁN MARTÍN

# Índice de contenido

# Índice de tablas

# Índice de ilustraciones

# SECCIÓN I: MEMORIA DEL PROYECTO

# 1.- Introducción

En esta sección de la memoria, mi objetivo es presentar al lector una idea global del estado en que se encuentra actualmente el mundo de los dispositivos móviles, el desarrollo de aplicaciones para dichos dispositivos y el Sistema Operativo (S.O.) Android.

Al ver estado tecnológico actual, podemos observar fácilmente que la tecnología a avanzado y progresado a pasos gigantescos en todos los aspectos que nos rodean, como pueden ser desde la clásica ley de Moore para los transistores contenidos en los microprocesadores, hasta las televisiones que han pasado de tubos de rayos catódicos a paneles LED o AMOLED de pantallas planas y ultrafinas o pantallas curvas con resoluciones 4K y sistemas operativos integrados para ofrecer multitud de funciones, pasando por los automóviles que ahora son aparatos equipados con las últimas novedades en tecnología aplicadas a la seguridad, entretenimiento, confort, ecología, etc. que dan como resultado automóviles que aparcan solos, te avisan de peligros, muestran la información en el parabrisas para que no se desvíe la mirada de la carretera, que cada vez sean más populares los automóviles tanto híbridos como plenamente eléctricos, otro ejemplo son las bombillas que han pasado de incandescentes a bajo consumo y de ahí a tecnología LED y con posibilidad de control remoto mediante Wifi y una aplicación en un dispositivo móvil, estas bombillas podrían incluirse en los avances en domótica y automatización de los hogares y así podríamos enumerar gran cantidad de avances en cualquier campo, con lo que podemos ver que el mundo avanza y nosotros avanzamos con él.

Dentro de estas evoluciones el ecosistema de los dispositivos móviles no es una excepción, de hecho es uno de los sectores con mayor avance y que más ha influenciado al ser humano y la manera de relacionarnos. Desde el inicio de la telefonía móvil hasta hoy, se han producido grandísimos cambios tanto en los dispositivos como en los usos. Inicialmente eran equipos muy grandes y de gran peso que solo podían usarse para hacer y recibir llamadas de teléfono, ahora los equipos son pequeños y fáciles de transportar y desde la aparición, evolución y éxito de los teléfonos inteligentes, donde las teclas y botones físicos prácticamente han desaparecido para dejar paso a grandes pantallas táctiles y donde más que teléfonos, los terminales y dispositivos móviles pueden ser considerados como pequeños ordenadores personales que están siempre disponibles en el bolsillo del usuario en cualquier lugar permitiendo que el abanico de usos y aplicaciones para los que se pueden usar haya aumentado y seguirá aumentando de forma increíble. El uso principal de hacer y recibir llamadas ha cambiado ya que ahora llamas a una persona no a un lugar concreto, además de este uso, se pueden enviar mensajes de texto mediante la red de telefonía móvil (SMS), usar posicionamiento GPS, tener un reproductor de música y vídeos, cámara de fotos, envío de mensajes instantáneos de texto, fotos, vídeos o sonidos... a través de Internet (mediante conexiones 3G, 4G, WIFI) mediante diversas aplicaciones, plataforma de juegos y entretenimiento móvil y uso de múltiples aplicaciones que otorgan nuevas funcionalidades de diversa índole.

Dentro de esta vorágine evolutiva, hay varios Sistemas Operativos para controlar los dispositivos, uno de ellos y quizás el más destacable es Android y es en el que se centrará este proyecto. Más adelante se entrará en detalle de los Sistemas Operativos.

## *1.1.- Estructura del documento*

En este apartado ofrezco un breve resumen de lo que se podrá ver en cada apartado de este Trabajo de Fin de Grado y la estructura que tiene.

- Sección I. Memoria del proyecto: Esta sección está dedicada al estudio y planteamiento inicial y a la fase de desarrollo de la aplicación. Incluye los apartados:
  - Introducción:
  - Identificación del Trabajo de Fin de Grado:
  - Descripción general del proyecto:
    - Motivación
    - Objetivos
    - Definición de Bits de Inteligencia
    - Descripción del sistema
    - Metodologías
    - Medios empleados
    - Planificación
    - Presupuestos
    - Ampliaciones
  - Estado del arte
    - Plataformas y Sistemas operativos
    - Tipos de desarrollos
    - Plataforma Android
    - Aplicaciones similares
    - Buenas prácticas en Android
    - Google Play Services
  - Modelo de negocio
  - Plan de negocio
  - Diferencias entre modelo y plan de negocio
- Sección II. Documentación técnica: Esta sección está dedicada a todo tipo de documentación, análisis y diagramas que definen el uso y comportamiento aplicación. Incluye los apartados:
  - Análisis de Requisitos
  - Requisitos Funcionales

  - Requisitos No Funcionales
  - Diagrama de clases
  - Diagramas de Casos de Uso
  - Diagramas de Secuencia
  - Diagramas de Actividad
  - Diseño de la Base de Datos
  - Usabilidad
  - Batería de Pruebas
- Sección III. Manual de usuario:
- Anexos: Esta sección incluye documentación adicional que aunque no es imprescindible para el proyecto, esta bien para ampliar nuestros conocimientos relativos a este proyecto.

## *1.2.- Estructura del soporte magnético (CD)*

En este apartado muestro como es la estructura de carpetas y archivos que están contenidos en el soporte magnético que acompaña a esta documentación.

En la raíz del dispositivo habrá tres carpetas:

- Documentación
- Código Fuente
- Aplicación

Dentro de la carpeta de **Documentación** estará la documentación del proyecto. Estará compuesta por las siguientes carpetas:

- Memoria: Contendrá los archivos pertenecientes a la memoria en formato *PDF*.
- Diagramas: Contendrá los diagramas usados en la memoria. Tendrán las extensiones *MDJ* para abrir dichos archivos con el programa *StarUML, DIA* para abrir los archivos con el programa *DIA Diagram Editor* y GAN para abrir los archivos con GanttProject. Estará subdivido en las siguientes carpetas:
  - Casos de Uso
  - Diagrama de Clases
  - Diagramas de Secuencia:
  - Diagramas de Actividad
  - Diagramas de la Base de Datos

- Diagrama de Gantt.

Dentro de la carpeta de **Código Fuente** estará el código fuente de la aplicación listo para ser importado por *Android Studio*.

Dentro de la carpeta Aplicación, estará el archivo instalable de la aplicación para que se pueda instalar en los dispositivos Android con versión 4.0 o superior. Tiene la extensión APK.

# 2.- Identificación del Trabajo de Fin de Grado

**Título:** Bits de Inteligencia para Android

**Descripción:** Este Trabajo de Fin de Grado consiste en el desarrollo de una aplicación para dispositivos Android basándose en el método didáctico de "Bits de Inteligencia" para la estimulación y aprendizaje de los niños, facilitando el acceso a este método de forma sencilla y aportando el plus del uso de las nuevas tecnologías.

**Autor:** David Jiménez Bermejo

**Fecha de Entrega:** Julio de 2016.

**Tutor:** Luis Ignacio Sebastián Martín.

**Departamento:** Informática.

**Universidad:** Escuela Ingeniería Informática adscrita a la Universidad de Valladolid

# 3.- Descripción general del proyecto

Esta sección está dedicada a ofrecer un análisis del proyecto, partiendo desde la motivación desde la que surgió la idea y recorriendo el camino de los objetivos a lograr, teniendo en cuenta los análisis que hay que realizar, metodologías y herramientas a utilizar, planificaciones y presupuesto.

## *3.1.- Motivación y Planteamiento*

La idea y motivación para la realización de este Trabajo de Fin de Grado es que cada día que pasa, el uso de teléfonos inteligentes y tabletas se está incrementando, y no solo en personas adultas si no que los niños también usan más estos dispositivos por que los tienen al alcance de la mano ya sea el de algún familiar, el suyo propio ya que hay dispositivos pensados y desarrollados para ellos o en los propios colegios donde se implanta cada vez más el uso de tabletas para realizar los ejercicios y explicaciones. Al estar los niños cada día más habituados a usar todos estos dispositivos electrónicos y cada vez desde edades más tempranas se hace imprescindible la adaptación y aparición de aplicaciones y contenidos que puedan usar para favorecer su aprendizaje y la adquisición de conocimientos de forma entretenida y a modo de juegos.

Con esa idea de desarrollo de contenidos y conociendo el método didáctico Bits de Inteligencia para la estimulación y el aprendizaje temprano de los niños, me surgió la idea de esta aplicación que considero de gran utilidad para y de gran valor ya que consiste fomentar el aprendizaje de los niños que son el mayor valor que tenemos.

Desde el punto de vista personal, como informático de profesión y vocación, porque ya desde niño me gustaba y sabía que quería dedicarme a la informática y di mis primeros pasos en ella de forma autodidacta con libros de la biblioteca pública y un viejo ordenador, consideró que es imprescindible adaptarse a este tipo de tecnología, porque es ampliamente demandada y en el mundo del usuario final la más usada ya que el dispositivo móvil siempre está a mano y encendido y resulta más cómodo usarlo para consultar información y entretenerse en cualquier sitio, que usar un ordenador personal, por ejemplo, estos dispositivos los puedes usar mientras esperas al autobús, metro, en la cola del supermercado... y tienes acceso a la información que necesitas al momento de forma cómoda y práctica sin tener que esperar a llegar a casa, encender el ordenador y realizar la tarea que necesites. Además de esto, debido a su sencillez de manejo hay personas que no han crecido de la mano de las tecnologías que no saben usar un ordenador, y en muchos casos ni se lo plantean, pero si tienen un dispositivo móvil que han aprendido a usar,

en algunos casos por motivación propia y en otros para comunicarse con sus familiares y conocidos y que de otra manera no abrían entrado en las nuevas tecnologías.

Otro dato personal es que considero a los niños personas maravillosas a las que hay que dedicar el máximo cariño, atención y apoyo y quiero hacer algo por ellos en general y por mis sobrinos en particular con lo que poder pasar más tiempo con ellos y colaborar en su aprendizaje y desarrollo.

## *3.2.- Objetivos*

El objetivo del Trabajo de Fin de Grado es el diseño y desarrollo de una aplicación para el sistema operativo Android que implemente el método didáctico Bits de Inteligencia donde a través de unas listas de selección, se escogerá la galería de Bits que se quiere ver y ésta se mostrará por pantalla además, se reproducirá un sonido que indique el Bit que se muestra por pantalla, tal y como especifica el método didáctico, esté sonido podrá ser en Español o Inglés, idioma que se habrá seleccionado previamente.

Toda la aplicación deberá ser muy fácil y simple de usar de forma intuitiva debido al público al que va destinada, que solo se fijan en los elementos visuales y llamativos.

Para mostrar de una forma clara y visual los objetivos marcados para este Trabajo de Fin de Grado, los muestro en la siguiente lista:

- Definición, diseño y desarrollo de una aplicación completamente funcional en el Sistema Operativo Android basada en el método didáctico de Bits de Inteligencia.
- Realización de la memoria correspondiente a la aplicación desarrollada.
- Analizar el funcionamiento de la *Play Store* de Google para la distribución de aplicaciones.
- Planteamiento de posibles líneas futuras de trabajo, para la mejora y ampliación de la aplicación.
- Definir las conclusiones y valoraciones obtenidas.
- Aprobar el Trabajo de Fin de Grado y conseguir el título de Graduado.
- Por último, pero el más importante para mí personalmente, ampliar mis conocimientos y aprender a desarrollar aplicaciones en Android.

## *3.3.- Definición de Bits de Inteligencia*

Ya que la aplicación se basa en el método didáctico de los Bits de Inteligencia y ya hemos hecho mención en varias ocasiones, antes de continuar, es importante que definamos que es el método didáctico de los Bits de Inteligencia.

Los Bits de Inteligencia son un método didáctico dirigido a niños/as de entre 0-6 años cuyo objetivo es que los niños mejoren su atención, facilitar la concentración y desarrollar y estimular el cerebro, la memoria y el aprendizaje. Con esto, desarrollan la inteligencia (facultad de actuar eficazmente ante situaciones nuevas). Este comportamiento se da gracias a que el cerebro es capaz de relacionar conocimientos nuevos con datos de experiencias pasadas (sinapsis).

El conocimiento se basa en estímulos que tomamos del medio a través de los sentidos y perduran en nuestra memoria si se presentan continuamente y sobre todo a temprana edad.

Creado por el médico estadounidense Glenn Doman, con el objetivo de estimular al niño para que aprenda.

Su metodología se basa en mostrar información visual y auditiva de forma escueta y rápida, mediante tarjetas de información. Se ha comprobado que los estímulos cortos son más eficaces que los largos, por lo que los bits se mostrarán rápidos, repetidamente en varias sesiones cortas y con gran alegría para atraer su atención y motivación.

El método se trabaja a través de categorías que agrupan un número de imágenes relacionas por áreas de conocimiento.

Los niños aprende de este modo, sin darse apenas cuenta, nuevos términos y significados, clasificados y estructurados, que servirán para desarrollar y ampliar el lenguaje, el vocabulario y la memoria.

Es un método potencial que les favorecerá de forma significativa en etapas educativas posteriores.

En el ámbito pedagógico, entendemos **BIT** como cualquier dato simple que pueda almacenar el cerebro y que llegue a través de los sentidos. Los bits de inteligencia son unidades de información que son presentadas a los niños de forma breve, con lo que se consigue captar su atención. Los bits son estímulos.

Los bits son unidades de información visuales, por medio de imágenes claras, precisas y bien definidas o una fotografía de buena calidad (estímulo visual) unidas a información auditiva, que consiste en enunciar en voz alta lo que se representa en la información visual (estímulo auditivo). Un bit de inteligencia es un bit de información.

**Características de los bits**: Un bit debe ser:

- Preciso: Lo más exacto posible.
- Concreto: Debe contener un único elemento.
- Claro: Debe ser una imagen o fotografía de calidad.
- Grande: La imagen debe ocupar casi toda la lámina/pantalla.
- Novedoso: Debe tratarse de algo que el niño no conoce.
- Exacto: Cada bit debe estar formado por una única imagen o fotografía. Por ejemplo: para hacer el bit de inteligencia del “sol”, se debe buscar una imagen con un sol, no valdría un paisaje con un sol (puesto que ahí, ya aparecen varios conceptos en una misma imagen).

**Categorías**: Una categoría es un grupo de bits que tienen entre sí una relación lo más estrecha posible. Las categorías pueden pertenecer a cualquier rama del saber humano (Astronomía, Bellas Artes, Botánica, Geografía, Historia, Música...), por ejemplo, en la categoría de "Los Planetas" cada bit de inteligencia sería un planeta diferente.

A la hora de seleccionar las categorías, hay que seguir una serie de criterios como los siguientes:

- Empezar por los elementos del entorno (árboles, plantas o flores del barrio, razas de las mascotas de los vecinos, monumentos de la localidad…).
- El interés de los niños (deportistas, animales, instrumentos musicales, señales de tráfico…).
- El interés de los profesores, padres o personas que vayan a realizar el método didáctico con los niños (obras de arte, monumentos, mapas…).

Algunos ejemplos de categorías de bits podrían ser los siguientes:

- Geografía: mapas de continentes, países, regiones provincias, accidentes geográficos, banderas, escudos, rincones del mundo...
- Historia: reyes, héroes, exploradores, descubridores, hechos históricos...
- Arte: retratos de artistas, obras, monumentos...
- Zoología: razas de perros, gatos, etc. Animales por continentes. Peces marinos y de agua dulce. Animales domésticos y salvajes. Crías de animales...
- Botánica: frutas, verduras, árboles frutales, árboles ornamentales, plantas, hojas, flores...
- Música: instrumentos y sus sonidos, genios musicales, notación musical…
- Cine, teatro, literatura, inventos, inventores, deportes, medios de transporte, figuras geométricas, astronomía…

Cada categoría de bits debe estar compuesta por cinco bits.

Al ahora de la **presentación**, hay que tener en cuenta los siguientes factores:

- Los bits se mostrarán de una forma rápida, agrupados por categorías, en tres sesiones diarias y durante cinco días.
- Cada sesión debe incluir cinco categorías (unos 25 bits) y la duración de cada categoría es de cinco días (un bit se visualiza quince veces).
- Después se retirarán sin comprobar si los niños los han aprendido.
- Solo se anunciará la categoría y se dirá el nombre de cada uno de los bits con alegría para atraer la atención de los niños.
- Cada estímulo dura un segundo y las sesiones deben ser siempre muy breves.
- El silencio de los niños y el entusiasmo de la persona que presenta los bits son imprescindibles.
- Ningún bit constituye un estímulo tan importante que haya que recibirlo obligatoriamente.

Este método no pretende enseñar directamente, sino estimular las áreas cerebrales, especialmente de la vista y el oído.

Hoy en día, recurren a él miles de centros educativos y hogares de todo el mundo.

Teniendo en cuenta todo lo dicho anteriormente y que la aplicación puede ser usada por los niños directamente, hay algunos aspectos de la presentación que pueden no seguirse fehacientemente como el número de repeticiones de cada categoría o el número de sesiones diarias.

Información recopilada de las siguientes fuentes:

- http://www.disanedu.com/index.php/metodo-glenn-doman/bits-de-inteligencia
- http://losbitsdeinteligenciaanaegido.blogspot.com.es/2012/04/introduccion-los-bits-de-inteligencia.html

## *3.4.- Descripción del sistema*

De acuerdo a los objetivos fijados y al método didáctico, es sistema tendrá las siguientes características:

- Selección de idioma en el que se mostraran los Bits de Inteligencia pudiendo escoger entre Español o Inglés.
- Mostrar una lista de las categorías y subcategorías de Bits de Inteligencia que estarán almacenadas en una base de datos.
- Una vez seleccionada la categoría y subcategoría deseada, se mostrará por pantalla la relación de Bits de Inteligencia correspondientes, mostrando la imagen del Bit, la palabra que define ese Bit y se reproducirá el sonido correspondiente a la palabra que define el Bit.
- El cambio entre Bit y Bit de la representación por pantalla de la relación de Bits seleccionada se realizará de forma automática.
- Habrá un contador que nos indique la cantidad de veces que se ha reproducido cada una de las galerías de Bits, este contador se mostrará en la lista junto con el nombre de la categoría y/o subcategoría y se podrán reinicializar a 0 (cero) los contadores de todas las galerías a la vez.
- Se podrá seleccionar que la palabra que define el Bit se muestre con las letras mayúsculas o minúsculas (todas las letras que conformen la palabra estarán en el mismo formato, es decir, todas mayúsculas o todas minúsculas).

## *3.5.- Metodologías*

Todo proyecto que se quiera realizar con éxito, debe basarse en una metodología de trabajo que guie el camino y establezca las pautas a seguir para el correcto desarrollo y estructuración del proyecto.

Hay dos corrientes de metodologías, las metodologías tradicionales y las metodologías ágiles. Vamos a ver una pequeña comparación entre ellas en la siguiente tabla.

| Metodologías Tradicionales | Metodologías Ágiles |
|---|---|
| Rigidez ante los cambios, los afronta de forma lenta o moderada | Flexibilidad ante los cambios del proyecto que se afrontan de forma moderada a rápida |
| Los clientes interactúan con el equipo de desarrollo mediante reuniones | Los clientes forman parte del equipo de desarrollo |
| Hay grupos de gran tamaño y a veces distribuidos en diferentes lugares | Grupos reducidos (con una media de 10 participantes) en el mismo lugar |
| Dependencia de la arquitectura del software mediante modelos | Menor dependencia de la arquitectura sel software |
| Poco *feedback* por lo que se aumentan los plazos de entrega | Continuo *feedback* acortando el tiempo de entrega |
| Pocos roles | Diversidad de roles |
| Están basadas en normas de estándares de desarrollo | Basadas en heurísticas* a partir de prácticas de producción de código |
| Procesos muy controlados por políticas y normas | Procesos menos controlados, pocas políticas y normas |
| Seguimiento estricto del plan inicial de desarrollo | Capacidad de respuesta antes los cambios |
| (*) Heurísticas: objetivos fundamentales, consiste en encontrar algoritmos con buenos tiempos de ejecución y buenas soluciones, usualmente las óptimas. Una **heurística** es un algoritmo que abandona uno o ambos objetivo mencionados. | |

*Tabla 1: Comparativa entre metodologías tradicionales y ágiles*

Teniendo en cuenta estas características, voy a elegir una metodología ágil, que está más enfocada en las tareas a desarrollar y su consecución que en un estricto control y generación de documentación. Se ha elegido la metodología *SCRUM* debido a sus características, que son las siguientes:

- Adopción de una estrategia de desarrollo incremental, en lugar de la planificación y ejecución completa del producto.
- Basa la calidad del resultado más en el conocimiento tácito de las personas en equipos auto organizados, que en la calidad de los procesos empleados.

- Hay un solapamiento de las diferentes fases del desarrollo, en lugar de realizar una tras otra en un ciclo secuencial o en cascada.
- Permite la creación de equipos auto-organizados impulsando la colocalización de todos los miembros del equipo, y la comunicación verbal entre todos los miembros y disciplinas involucrados en el proyecto.
- Un principio clave de *Scrum* es el reconocimiento de que durante un proyecto los clientes pueden cambiar de idea sobre lo que quieren y necesitan, y que los desafíos impredecibles no pueden ser fácilmente enfrentados de una forma predictiva y planificada. Por lo tanto, *Scrum* adopta una aproximación pragmática, aceptando que el problema no puede ser completamente entendido o definido, y centrándose en maximizar la capacidad del equipo de entregar rápidamente y responder a requisitos emergentes.

Las características más marcadas que se logran en *Scrum* serían: gestión regular de las expectativas del cliente, resultados anticipados, flexibilidad y adaptación, retorno de inversión, mitigación de riesgos, productividad y calidad, alineamiento entre cliente y equipo, por último equipo motivado. Cada uno de estos puntos mencionados hacen que el *Scrum* sea utilizado de manera regular en un conjunto de buenas prácticas para el trabajo en equipo y de esa manera obtener resultados posibles.

Existen varias implementaciones de sistemas para gestionar el proceso de *Scrum*, que van desde notas amarillas *post-it* y pizarras (*Scrumboards*) hasta paquetes de software. Una de las mayores ventajas de *Scrum* es que es muy fácil de aprender, y requiere muy poco esfuerzo para comenzarse a utilizar. Así, si se utiliza una pizarra con notas autoadhesivas cualquier miembro del equipo podrá ver tres columnas: trabajo pendiente, tareas en proceso y hecho. De esta forma, de un solo vistazo, una persona puede ver en qué están trabajando los demás en un momento determinado. Esta división de tareas hace que las notas con las tareas deban ir moviéndose de una sección a otra. Este movimiento físico de las notas sirve además para crear una sensación de compromiso con el proyecto, una percepción de continuidad y una motivación extra cuando se van sacando las tareas adelante.


*Ilustración 1: Ejemplo de Scrumboard de la metodología Scrum*

El funcionamiento es que durante cada *sprint*, un periodo o fase de desarrollo que tiene una duración de entre una y cuatro semanas (la duración es definida por el equipo y debe ser lo más corta posible), el equipo crea un incremento de software potencialmente entregable (utilizable), es decir que podría asemejarse fácilmente al producto final. El conjunto de características que forma parte de cada *sprint* viene dado por un conjunto de requisitos de alto nivel priorizados que definen el trabajo a realizar (*Product Backlog*). Los elementos del *Product Backlog* que forman parte del *sprint* se determinan durante la reunión de *Sprint Planning*. Durante esta reunión, el cliente identifica los elementos del *Product Backlog* que quiere ver

completados y los pone en conocimiento del equipo. Entonces, el equipo conversa con el cliente buscando la claridad y magnitud adecuadas para luego determinar la cantidad de ese trabajo que puede comprometerse a completar durante el siguiente *sprint*. Durante el *sprint*, nadie puede cambiar el *Sprint Backlog*, lo que significa que los requisitos están congelados durante el *sprint,* Pero pueden cambiarse o alterarse de cara al siguiente *sprint*.

Cada día de un *sprint*, se realiza la reunión sobre el estado de un proyecto. Esto se llama *daily standup* o *Stand-up meeting* o *daily Scrum*. Estas reuniones tienen que realizarse en la misma ubicación y a la misma hora todos los días y con puntualidad, tienen una duración de 15 minutos independientemente del tamaño del grupo y todos son bienvenidos pero únicamente pueden hablar los involucrados en el proyecto, que den responder a las siguientes preguntas:

- ¿Qué has hecho desde ayer?
- ¿Qué es lo que harás para mañana?
- ¿Has tenido algún problema que te haya impedido alcanzar tu objetivo?

El objetivo último de las reuniones diarias es que cada miembro del equipo sepa si se están cumpliendo los plazos marcados para el *sprint*.

Existen además otros tipos de reuniones:

- *Scrum* de *Scrum*: Estas reuniones, por lo general, se realizan cuando en la organización existan varios equipos Scrum, y les permiten discutir su trabajo, enfocándose especialmente en áreas de solapamiento e integración. Se hace normalmente cada día después del *Daily Scrum* o, como máximo, cada dos días. Asiste una persona asignada por cada equipo *Scrum*. Tienen la misma agenda que el *Daily Scrum* pero añadiendo un control de los equipos al incluir las siguientes preguntas:
  - ¿Qué ha hecho tu equipo desde nuestra última reunión?
  - ¿Qué hará tu equipo antes que nos volvamos a reunir?
  - ¿Hay algo que demora o estorba a tu equipo?
  - ¿Estás a punto de poner algo en el camino del otro equipo?
- Reunión de planificación del *sprint*: Al inicio de cada ciclo de s*print*, se lleva a cabo una reunión de planificación del *sprint* para establecer que tareas se van a realizar y los requisitos y el tiempo que llevará realizar las tareas. Tiene un tiempo límite de ocho horas.
- Reunión de revisión del *sprint*: Esta reunión se realiza al final del ciclo del s*print* y sirve para revisar las tareas que fueron completadas y no completadas, presentar el trabajo realizado a los clientes (iteración o demo), el trabajo incompleto no puede ser mostrado. Tiene una duración máxima de cuatro horas.
- Retrospectiva del *sprint*: Se realiza después de cada *sprint*, se lleva a cabo una retrospectiva del propio *sprint*, en la cual todos los miembros del equipo dejan sus impresiones sobre el *sprint* recién superado. El propósito de la retrospectiva es realizar una mejora continua del proceso. Esta reunión tiene un tiempo fijo de cuatro horas.

Con todas estas características descritas, obtenemos que los beneficios de Scrum son los siguientes:

- **Flexibilidad a cambios:** Gran capacidad de reacción ante los cambiantes requerimientos generados por las necesidades del cliente o la evolución del mercado. El marco de trabajo está diseñado para adecuarse a las nuevas exigencias que implican proyectos complejos.
- **Reducción del *Time to Market:*** El cliente puede empezar a utilizar las características más importantes del proyecto antes de que esté completamente terminado.
- **Mayor calidad del software:** El trabajo metódico y la necesidad de obtener una versión de trabajo funcional después de cada iteración, ayuda a la obtención de un software de alta calidad.
- **Mayor productividad:** Se logra, entre otras razones, debido a la eliminación de la burocracia y la motivación del equipo proporcionado por el hecho de que pueden estructurarse de manera autónoma.
- **Maximiza el retorno de la inversión:** Creación de software solamente con las prestaciones que contribuyen a un mayor valor de negocio gracias a la priorización por retorno de inversión.
- **Predicciones de tiempos:** A través de este marco de trabajo se conoce la velocidad media del equipo por *sprint*, con lo que es posible estimar de manera fácil cuando se podrá hacer uso de una determinada funcionalidad que todavía está en las tareas pendientes.
- **Reducción de riesgos:** El hecho de desarrollar, en primer lugar, las funcionalidades de mayor valor y de saber la velocidad a la que el equipo avanza en el proyecto, permite despejar riesgos efectivamente de manera anticipada.

Por todos estas características y ventajas que he descrito considero una muy buena opción esta metodología ya que tiene una muy buena flexibilidad a los cambio, que son bastante frecuentes en Android para adaptarse a nuevos dispositivos y corregir errores, tiene iteraciones para ir desarrollando la aplicación por tareas y funcionalidades y mejora la calidad y la productividad. Además de que visualmente motiva mucho con la pizarra y la división de las tareas.

## *3.6.- Medios empleados*

En esta sección voy a explicar todos los elementos que han sido necesarios para el desarrollo de este Trabajo de Fin de Grado, diferenciando entre elementos físicos (hardware), programas y herramientas de desarrollo, edición, etc. (software) y servicios utilizados.

### *3.6.1.- Herramientas de desarrollo - software*

Para el desarrollo de este Trabajo de Fin de Grado se han usado diferentes programas, para su elección se ha teniendo en cuenta que sean programas estándar y de precio comedido o gratuito. La lista de programas usados es la siguiente:

1. **Android Studio**: Entorno de desarrollo oficial para la creación de aplicaciones para dispositivos Android. Lo ofrece, soporta y mantiene Google. El desarrollo de aplicaciones se realizará mediante Java, Sqlite y XML. Se usará para el desarrollo de la aplicación. Es gratuito.
2. **OpenOffice**: Herramienta ofimática que se usará para desarrollar la documentación y la memoria necesaria. Es gratuito.
3. **Adobe Reader**: Lector de archivos PDF. Se usará para leer documentación de ayuda que esté en formato PDF y comprobar el correcto visualizado de la memoria a entregar. Es gratuito
4. **Mozilla Firefox**: Navegador web que se utilizará para todo tipo de consultas en Internet, ya sean de dudas o problemas con el desarrollo, búsquedas de información, imágenes, sonidos, etc. o para descarga de programas o manuales. Es gratuito.
5. **GIMP**: Programa de edición de imágenes que se usará para la preparación de las imágenes que se utilizarán, como pueden ser los Bits de Inteligencia, iconos, fondos, etc.
6. **Gantt Project**: Aplicación para la creación de diagramas de Gantt. Es gratuito.
7. **DIA Diagram Editor**: Aplicación para la creación de diagramas, por ejemplo diagramas de bases de datos. Es gratuito.
8. **Star UML**: Aplicación para la creación de diagramas UML. Es gratuito.
9. **Microsoft Windows®**: Sistema operativo que usa el ordenador. Usada la versión Windows 10. Tiene un coste.

### *3.6.2.- Tecnologías y Lenguajes de programación*

En esta sección voy a explicar los lenguajes usados, vienen definidos por la elección de Android como plataforma de desarrollo, ya que dicta el lenguaje de programación, los métodos de almacenamiento y configuración, etc. que hay que usar. Ahora paso a describirlos, empezado por la propia plataforma Android, aunque posteriormente en el apartado Estado de Arte, se describirá esta plataforma mucho más detalladamente.

- **Android**: Después de analizar detalladamente la plataforma Android, como se verá en la sección Estado del Arte, primeramente opté por desarrollar la aplicación sobre la versión 2.2 de Android para que fuera compatible con todos los dispositivos, ya que aunque uso *Fragments* que inicialmente no están soportados, cuando aparecieron en la versión 3.0, crearon librerías que los soportaban para versiones anteriores. Finalmente tras las primeras primeras pruebas con esa versión de Android y tener numerosos problemas con los *Fragments* y documentarme sobre estos problemas, descubrí que las librerías que los soportaban tenían varios *bugs*, por lo que decidí cambiar la versión sobre la que estaba trabajando y elegí como nueva versión de Android para la aplicación la 4.0 ya que así solo hay un 2.2% de dispositivos que no podrán instalar la aplicación y a partir de esta versión el soporte para pantallas más grandes y trabajo con *Fragments* esta optimizado. A la hora de tener en cuenta el apartado gráfico para las imágenes, no tendré en cuenta las

pantallas pequeñas y las resoluciones bajas, centrándome en las pantallas *Normal*, *Large* y *XLarge* y en las resoluciones *mdpi*, *hdpi*, *xhdpi* y *xxhdpi*, ya que son las mayoritarias.

- **Java**: Lenguaje de programación orientado a objetos que se usa para el desarrollo de las aplicaciones Android. Es independiente de la plataforma en la que se vaya a ejecutar, ya que al compilarse se crea un código como *bytecode* que se ejecuta en una máquina virtual, por lo que se puede ejecutar en cualquier dispositivo y hardware. Una característica importante es el recolector de basura que evita fugas de memoria, este recolector de basura borra los objetos y libera la memoria una vez que un objeto a perdido todas las referencias que tenía y lo hacían accesible.
- **SQLite**: Es motor de bases de datos ligero y de código abierto que sirve para almacenar la información de forma persistente de una manera sencilla. No es proceso independiente con el que se comunica el programa principal, sino que la biblioteca SQLite se enlaza con el programa principal pasando a ser parte del mismo y se utiliza a través de llamadas simples a subrutinas y funciones con lo que se reduce la latencia en el acceso a los datos. Tiene las siguientes características principales:
  - No requiere de servidor: SQLite no ejecuta un proceso para administrar la información, si no que implementa un conjunto de librerías encargadas de la gestión.
  - No necesita configuración: Libera al programador de todo tipo de configuraciones de puertos, tamaños, ubicaciones, etc.
  - Usa un único archivo: Crea un archivo para el esquema completo de una base de datos donde almacena el esquema, las definiciones, tablas, índices y los propios datos, lo que permite ahorrarse preocupaciones de seguridad, ya que los datos de las aplicaciones están en un único archivo perteneciente a la aplicación.
  - Es de código abierto: Está disponible al dominio público de los desarrolladores al igual que sus archivos de compilación e instrucciones de escalabilidad.

  Por todas estas características, SQLite es una tecnología cómoda para los dispositivos móviles y de recursos limitados. Su simplicidad, rapidez y usabilidad permiten un desarrollo muy amigable.
- **XML**: Es un lenguaje de marcas, del inglés e***X****tensible* ***M****arkup* ***L****anguage*, creado por el *World Wide Web Consortium* (W3C) utilizado para almacenar datos en forma legible. Permite definir la gramática de lenguajes específicos para estructurar documentos grandes. A diferencia de otros lenguajes, XML da soporte a bases de datos, siendo útil cuando varias aplicaciones deben comunicarse entre sí o integrar información. XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil. Se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable. En Android tiene varios cometidos, uno de ellos es diseñar las interfaces y *layouts* de las vistas y por otro sirve para almacenar datos (leer y escribir) de la aplicación, ya sean datos de configuraciones o datos guardados por el usuario.

### *3.6.3.- Medios materiales – hardware*

Los equipos físicos que se han utilizado para el desarrollo del Trabajo de Fin de Grado han sido los siguientes:

1. Ordenador Portátil: Utilizado para  para las tareas de desarrollo, consultas, pruebas, realización de documentación, etc. Los requerimientos son de un equipo de gama media con un procesador Intel i5 con 8 GB de RAM para poder mover de forma fluida *Android Studio* con el emulador de un dispositivo móvil.
2. Teléfono móvil inteligente: Utilizado para probar la aplicación desarrollada en dispositivo real y no solo en un emulador virtual. Es necesario que tenga como sistema operativo Android 4.0 o superior sin importar las características técnicas del dispositivo ya que la aplicación no es muy exigente en cuanto a recursos necesarios. Para pruebas se ha usado un *Samsung Galaxy S3*.
3. Dispositivo de almacenamiento USB: Utilizado para realizar copias de seguridad de todos los archivos del Trabajo de Fin de Grado. Se utiliza un dispositivo USB 3.0 de 8 GB de capacidad.
4. Soporte magnético (CD): Utilizados para entregarlos junto con la documentación del proyecto, contendrán los archivos necesarios de documentación, diagramas, aplicación, etc.

### *3.6.4.- Servicios*

Los servicios que han sido necesarios para la realización del Trabajo de Fin de Grado han sido los siguientes:

1. Conexión a Internet: Se necesita una conexión a Internet durante el desarrollo del proyecto para poder descargar los programas necesarios, buscar documentación y ayuda, consultar dudas, buscar imágenes. Se hace uso de una conexión ADSL.
2. Servicio de Imprenta: Se necesita un servicio de imprenta para poder imprimir y encuadernar la documentación a entregar.
3. Servicio de almacenamiento en la nube: Se ha utilizado el servicio de almacenamiento en la nube de *Dropbox* para tener copias de seguridad en la nube y accesibles desde cualquier lugar y en cualquier momento. Se ha utilizado el servicio gratuito.

## *3.7.- Planificación*

En este punto, voy a explicar la planificación del proyecto, viendo las reglas y modelos que se han utilizado para planificar y desarrollar el proyecto.

Todo esto se lleva a cabo teniendo siempre en cuenta el desconocimiento inicial de la plataforma Android y la escasez de tiempo disponible, ya que hay que para realizar este proyecto, se han tenido que compaginar, la vida laboral, el tiempo de descanso, vida familiar, etc. por lo que no se ha podido dedicar al proyecto el tiempo, dedicación y continuación que se hubiera deseado.

Hay que señalar que para todo desarrollo informático, al igual para que todo tipo de actividades diarias, es importante tener y seguir un modelo que aporte una estructura, un orden y una robustez. Para los desarrollos software, existe un modelo que está muy orientado a ellos y es el modelo en cascada.

El modelo en cascada se distingue por que el proceso de desarrollo está dividido en una serie de fases, que están rigurosamente ordenadas y donde cada fase tiene un conjunto de objetivos y tareas perfectamente definidas, de tal forma que para empiece una fase, tiene haber concluido completamente la fase anterior. De acuerdo a este funcionamiento, este modelo sigue la siguiente idea:

*"Definir antes que diseñar, diseñar antes que codificar"*

Las fases del modelo son las siguientes:

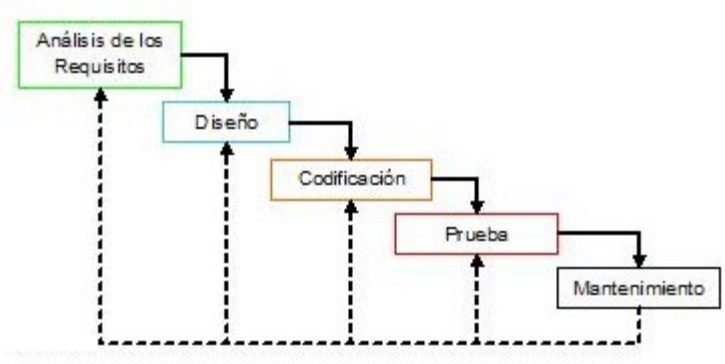


*Ilustración 2: Fases del modelo en cascada*

- **Análisis de Requisitos**: En esta fase se analizan las necesidades de los usuarios/clientes para obtener los objetivos a cumplir (rendimiento, interfaces, etc.). Lo que se determine en esta fase, condicionará las siguientes fases, no pudiéndose requerir nuevos resultados a mitad del proceso de elaboración del software de una manera.
- **Diseño**: La fase de diseño traduce los requisitos en una representación del software necesario con la calidad requerida antes de que comience la codificación. Se puede subdividir en:
    - **Diseño del sistema**: Define la estructura y organización del sistema mediante la descomposición y diferenciación de elementos que puedan elaborarse por separado, aprovechando las ventajas del desarrollo en equipo. Se defina la descripción de la estructura relacional global del sistema y la especificación de lo que debe hacer cada una de sus partes, así como la manera en que se combinan unas con otras.

    - **Diseño del programa**: Es la fase en donde se realizan los algoritmos necesarios para el cumplimiento de los requerimientos del usuario así como también los análisis necesarios para saber qué herramientas usar en la etapa de Codificación

- **Codificación**: En esta fase, el diseño se transforma en el código fuente del programa, haciendo especial hincapié en la creación de librerías y elementos reutilizables. Si el diseño se realiza de una manera detallada, la codificación puede realizarse mecánicamente.
- **Prueba**: Una vez que se ha generado el código, se ensamblan todos los elementos generados para crear el sistema completo y comenzar con las pruebas del programa. La fase de prueba se centra en la lógica interna del software y en las funciones externas, realizando pruebas que aseguren que la entrada definida produce los resultados que realmente se requieren, es decir, que cumple con los requisitos.
- **Mantenimiento**: Prácticamente todo programa o elemento software sufrirá cambios después de que se realice la entrega al cliente. Los cambios ocurrirán por que se hayan encontrado errores, o porque el software necesite alguna adaptación frente a cambios del entorno externo (sistema operativo, dispositivos periféricos, etc.) o a que el cliente requiera ampliaciones funcionales o del rendimiento. Debido a esto se contempla esta fase de mantenimiento para encargarse de dichos cambios y adaptaciones.

Pese a que tiene grandes ventajas como que es intuitivo y fácil de seguir, tiene sus contras y la parte más negativa a destacar de este modelo es cualquier error de diseño detectado en la fase de prueba conduce necesariamente a volver a la fase de diseño y rediseñar, lo que conlleva una nueva programación del código afectado, aumentando los costos del desarrollo.

### *3.7.1.- Planificación Temporal*

Este punto está dedicado a explicar la planificación temporal del proyecto, hay una planificación que se estimó inicialmente, basándose en las fases y división de tareas en la que se calculó la duración del proyecto, y posteriormente hay otra planificación ajustándome a la duración real que ha tenido el proyecto.

La planificación inicial es la que muestro a continuación, primero una visión de la descomposición en tareas y posteriormente la duración, ayudándome de un diagrama de GANTT.

| Duración | Nombre | Fecha de inicio | Fecha de fin |
|---|---|---|---|
| 61 | Bits de Inteligencia | 4/02/16 | 28/04/16 |
| 4 | Análisis | 4/02/16 | 9/02/16 |
| 1 | Estudio del problema | 4/02/16 | 4/02/16 |
| 1 | Analisis de Requisitos | 5/02/16 | 5/02/16 |
| 1 | Estudio de soluciones | 8/02/16 | 8/02/16 |
| 1 | Fijar objetivos | 9/02/16 | 9/02/16 |
| 2 | Planificación | 10/02/16 | 11/02/16 |
| 1 | Planificación temporal | 10/02/16 | 10/02/16 |
| 1 | Estudio presupuestario | 11/02/16 | 11/02/16 |
| 7 | Materiales didacticos | 12/02/16 | 22/02/16 |
| 3 | Busqueda de imagenes | 12/02/16 | 16/02/16 |
| 2 | Ajuste Imagen/Resolucion | 17/02/16 | 18/02/16 |
| 1 | Busqueda de sonidos | 19/02/16 | 19/02/16 |
| 1 | Procesar Sonidos | 22/02/16 | 22/02/16 |
| 5 | Diseño | 23/02/16 | 29/02/16 |
| 1 | Comportamiento de la App | 23/02/16 | 23/02/16 |
| 1 | Casos de Uso | 24/02/16 | 24/02/16 |
| 1 | Diagramas de secuencia | 25/02/16 | 25/02/16 |
| 1 | Base de Datos | 26/02/16 | 26/02/16 |
| 1 | Interfaz de la App | 29/02/16 | 29/02/16 |
| 20 | Implementación | 1/03/16 | 28/03/16 |
| 7 | Gestión Idiomas | 1/03/16 | 9/03/16 |
| 6 | Interfaz y Menús | 10/03/16 | 17/03/16 |
| 7 | Galeria de imagenes | 18/03/16 | 28/03/16 |
| 3 | Pruebas | 29/03/16 | 31/03/16 |
| 1 | Caja Blanca | 29/03/16 | 29/03/16 |
| 1 | Caja Negra | 30/03/16 | 30/03/16 |
| 1 | Corrección Fallos y Bugs | 31/03/16 | 31/03/16 |
| 20 | Documentación | 1/04/16 | 28/04/16 |
| 8 | Memoria Global | 1/04/16 | 12/04/16 |
| 7 | Manual Técnico | 13/04/16 | 21/04/16 |
| 5 | Manual de Usuario | 22/04/16 | 28/04/16 |

*Ilustración 3: Lista de tareas del diagrama de Gantt*

Antes de explicar la lista de tareas, quiero aclarar que la duración que aparece en la lista de tareas está medida en días, teniendo en cuenta que cada día son 8 horas de trabajo, aunque en la realidad, no pude trabajar todos los días seguidos ni dedicarle 8 horas al día.

En la planificación se ha tenido en cuenta que la aplicación tiene que ser simple e intuitiva, además de que ya se tenían los objetivos muy claros. Debido a esto, las fases de análisis, planificación y diseño no tiene una duración muy prolongada. Se ha planificado una parte considerable de tiempo a la elaboración de los materiales didácticos ya que son la parte esencial de la idea del proyecto y después se ha planificado que la fase de implementación y documentación tendrán la misma duración ya que considero que la elaboración de la presente

documentación es de vital importancia y hay que dedicarle mucho tiempo para que sea correcta y adecuada.

No se definió una tarea específica que fuera la "Adquisición de conocimientos sobre Android" ya pensé que con un breve estudio inicial y después con la consulta de dudas y problemas según surgían durante la implementación sería suficiente. Esto ha resultado ser un gran error y un problema. Entraré en más detalles más adelante al hacer la comparación entre la planificación inicial y la duración real del proyecto.

Ahora vamos a ver la sucesión de actividades que se planificó, mediante diagramas de Gantt. Para facilitar el correcto visionado, primero habrá u4n diagrama general con las fases, y después uno por cada fase. En todos los diagramas, en la parte de la izquierda aparece el nombre de la actividad y a la derecha la duración en días.

1. Diagrama de Gantt general

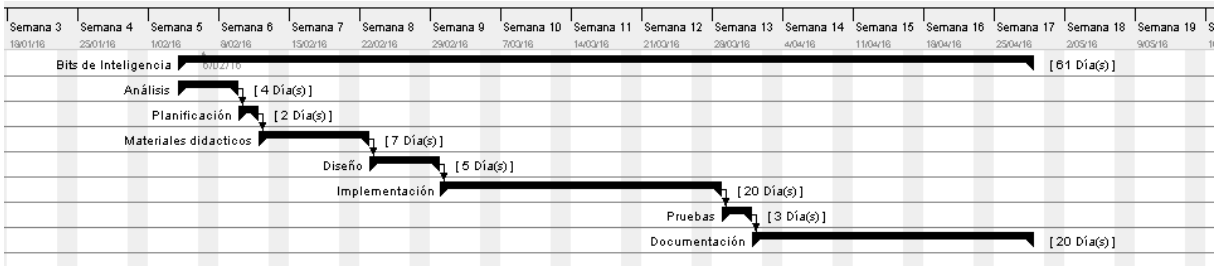


*Ilustración 4: Diagrama de Gantt General*

Como vemos, el proyecto total tiene una duración de 61 días, que a razón de 8 diarias hacen un total de 488 horas, con un intervalo de tiempo para realizarlo de 13 semanas, teniendo en cuenta que los fines de semana no se computan como tiempo laborable.

2. Diagrama de Gantt de la fase de análisis

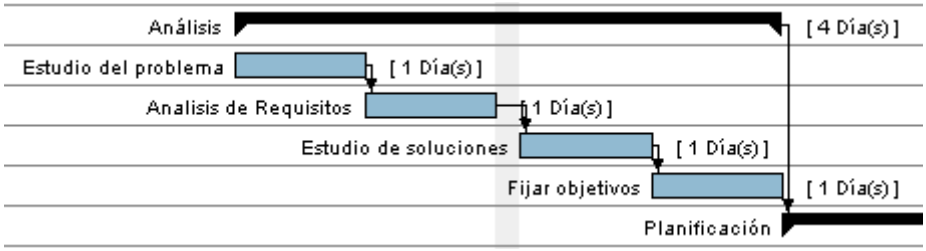


*Ilustración 5: Diagrama de Gantt - Fase de Análisis*

La fase análisis tiene una duración de 4 días, equivalente a 32 horas.

3. Diagrama de Gantt de la fase de planificación

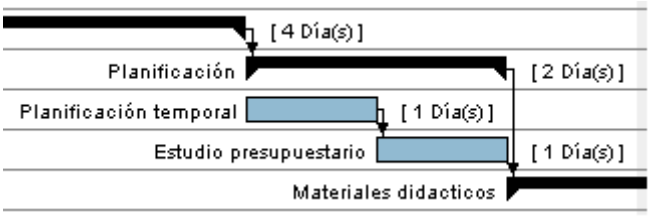


*Ilustración 6: Diagrama de Gantt - Fase de Planificación*

La fase planificación tiene una duración de 2 días, equivalente a 16 horas.

4. Diagrama de Gantt de la fase de materiales didácticos

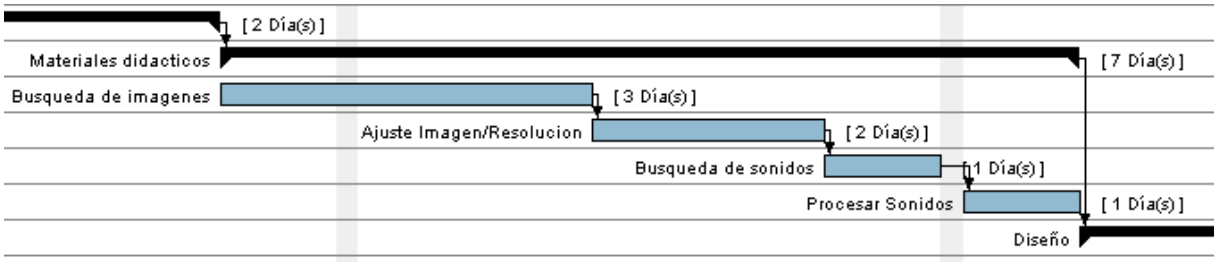


*Ilustración 7: Diagrama de Gantt - Fase de Materiales Didácticos*

La fase materiales didácticos tiene una duración de 7 días, equivalente a 56 horas. Está fase es de vital importancia para la parte gráfica del proyecto y que sea llamativo y atrayente para los usuarios.

5. Diagrama de Gantt de la fase de diseño

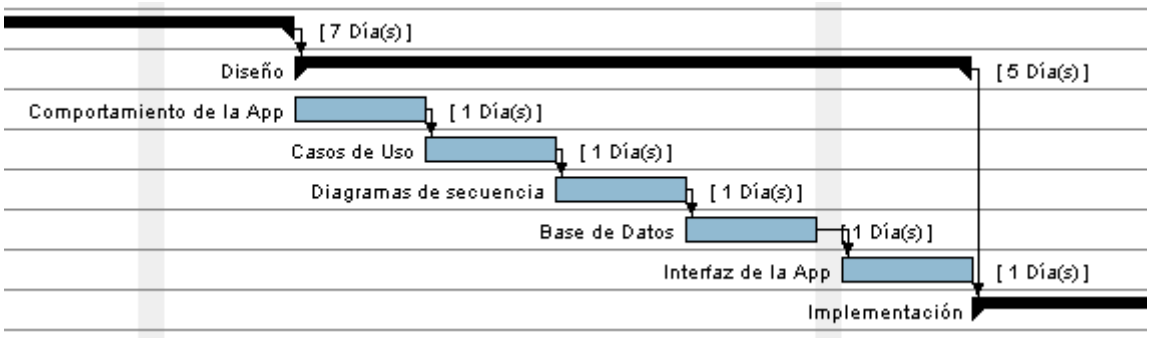


*Ilustración 8: Diagrama de Gantt - Fase de Diseño*

La fase diseño tiene una duración de 5 días, equivalente a 40 horas. Es una fase importante, ya que cuando mejor sea el diseño y mejor estén especificados los casos de uso, comportamientos, secuencias de utilización, más fácil y clara será la fase de implementación

6. Diagrama de Gantt de la fase de implementación

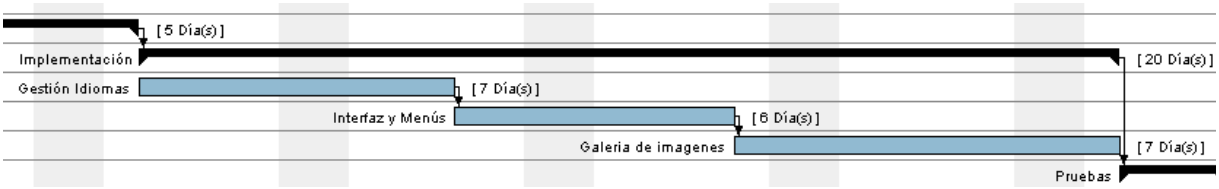


*Ilustración 9: Diagrama de Gantt - Fase de Implementación*

La fase planificación tiene una duración de 20 días, equivalente a 160 horas.

7. Diagrama de Gantt de la fase de pruebas

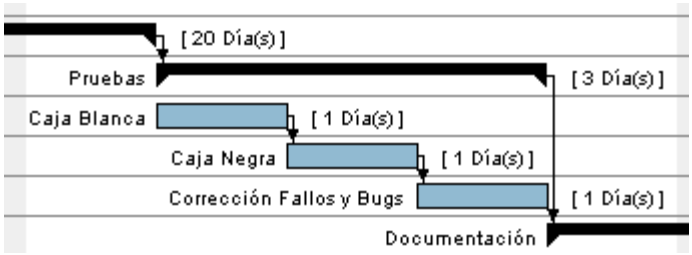


*Ilustración 10: Diagrama de Gantt - Fase de Pruebas*

La fase pruebas tiene una duración de 3 días, equivalente a 24 horas. No es una fase demasiado larga, ya que durante la implementación se van realizando pruebas parciales para comprobar el funcionamiento, por lo que las pruebas finales son fáciles y rápidas de realizar.

8. Diagrama de Gantt de la fase de documentación

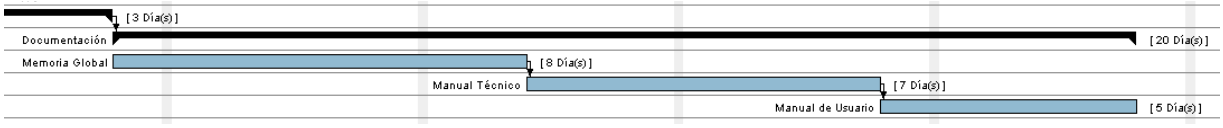


*Ilustración 11: Diagrama de Gantt - Fase de Documentación*

La fase documentación tiene una duración de 20 días, equivalente a 160 horas. Aunque esta fase está puesta al final del proyecto y una vez que se han terminado el resto de fases, realmente se realizará durante todo el proyecto de forma paralela al resto de tareas.

### *3.7.2.- Duración real del proyecto*

Esta parte está dedicada a presentar lo que ha sido la duración real que ha tenido el proyecto, ahora que ya se han finalizado la mayoría de las tareas, solo falta concluir la documentación. Voy utilizar como medida de tiempo horas, ya que al no dedicar cada día el mismo número de horas y poder trabajar todos los días, utilizar días no serviría para ajustarse a la temporización.

1. Fase de análisis:

Esta fase ha sido sencilla de realizar ya que la idea de la aplicación llevaba un tiempo rondándome por la cabeza y ya tenía bastantes claras las ideas que quería y como quería tomaran forma. Por lo que está fase tarde menos de lo planificado en realizarla, teniendo una duración real de **12 horas**.

2. Fase de planificación:

Esta fase ha tenido una duración de **18 horas**, la mayor parte de ellas han sido dedicadas a la planificación temporal, ya que se realizaron búsquedas de proyectos similares para comprobar lo que se podría tardar en realizar y la complicación que podrían tener y al estudio de los Bits de inteligencia y búsquedas superficiales de si es posible encontrar mucha materia prima de información, imágenes y sonidos.

3. Fase de materiales didácticos:

Esta ha sido una fase complicada de llevar a cabo ya que aunque hay gran variedad de imágenes aptas para ser utilizadas como bits, mi manejo de la aplicación GIMP no es muy bueno y me resultaba costoso que todas las imágenes tuvieran las mismas dimensiones y resoluciones. En lo que respecta al sonido, se han utilizado sintetizadores de texto a voz, con lo que el resultado no es tan bueno como me hubiera gustado. Se utilizó un tiempo considerable buscando y probando gran cantidad de sintetizadores.

Al final la duración de esta fase ha sido de **65 horas**.

4. Fase de diseño:

En esta fase, al igual que en la fase de análisis, al tener muy claras las ideas de antemano y haberlo pensado con mucho tiempo, junto con que la aplicación tiene que ser visual e intuitiva, se ha realizado con bastante facilidad y rapidez. Esta fase se ha realizado en **20 horas**.

5. Fase de implementación:

Esta fase ha sido sin lugar a dudas la más complicada de realizar y la que más esfuerzo ha requerido tanto temporal como mental ya que han surgido gran cantidad de problemas y dificultades, como por ejemplo tener que cambiar la versión de Android sobre la que se estaba

trabajando. En el siguiente punto en el que se comparan la planificación inicial y la duración real con los desfases que se han producido, explicaré con detalle los problemas surgidos. Está fase se completó en **300 horas**.

6. Fase de pruebas:

Esta fase después de los problemas surgidos durante la implementación fue prácticamente redundante, ya que para solucionar los problemas y comprender por que surgían, tuve que realizar muchísimas pruebas y correcciones durante la implementación, por lo que al llegar a esta fase estaban hechas prácticamente todas las pruebas posibles y por ello la duración se acorto mucho. Esta fase se completó en **6 horas**.

7. Fase de documentación:

Para completar la documentación se han invertido un total de **165 horas**.

### *3.7.3.- Comparación entre la planificación inicial y la duración final*

Para comenzar este apartado, voy a representar en forma de tabla la duración estimada de cada fase y la real, para con un simple vistazo poder ver las diferencias.

| Fases | Planificación Inicial | Duración realizando |
|---|---|---|
| Análisis | 32 | 12 |
| Planificación | 16 | 18 |
| Materiales Didácticos | 56 | 65 |
| Diseño | 40 | 20 |
| Implementación | 160 | 300 |
| Pruebas | 24 | 6 |
| Documentación | 160 | 165 |
| Total | 488 | 586 |

*Tabla 2: Comparación entre planificación inicial y duración real (horas)*

Como vemos, el proyecto ha durado más de lo que se planificó en un principio, habiendo invertido 98 horas más, que es aproximadamente un 17% más de lo que inicialmente se planificó.

La duración real ha sido de 6 meses desde que me matriculé del Trabajo de Fin de Grado en Febrero hasta la fecha en que se solicita la defensa en Julio, trabajando todo ese tiempo. Según la planificación inicial, en la que se empieza también en Febrero, se habría terminado en Abril, es decir una duración 3 meses, si se trabajara 8 horas diarias.

La **parte positiva** del proyecto en cuanto a la duración está en las fases de Análisis y Diseño, ya que han durado menos de lo planificado. Esto es debido a que la idea del proyecto está ya muy clara y se había dedicado mucho tiempo a estudiarla previamente a la ejecución del Trabajo de Fin de Grado, de lo que se deduce que cuanto más elaborada y clara se tenga la idea, más fácil es plasmar lo que se quiere.

La **parte neutral** se la llevan las fases de Planificación y Documentación ya que han durado prácticamente lo que se había estimado para ellas, ligeramente por encima.

La **parte negativa** sin lugar a dudas es la fase de Implementación que ha durado más del doble de lo que se había estimado. Esto es debido a que se han tenido gran cantidad de problemas y que cuando se estudiaron otros proyectos similares y la complejidad de Android, aunque tuve en cuenta la falta de experiencia, valoré de forma errónea la dificultad de Android pensando que sería más fácil de lo que realmente es en realidad. He tenido numerosos problemas con la creación de las vistas y la correcta visualización de la forma en que estaba planeada, problemas con actualizaciones de *Android Studio* y versiones de Android que provocaban que después de una actualización hubiera partes que dejaban de funcionar. Pese a todo el mayor problema ha estado con los *Fragments* y el cambiar el *Fragment* de forma dinámica para presentar los Bits en forma de secuencia, este problema me obligó a cambiar en mitad del desarrollo la versión de Android a utilizar pasando de la 2.2 inicial a la 4.0, ya que en la versión 2.2 los *Fragments* funcionan con una librería adaptada, ya que aparecieron en una versión posterior (3.0) y dicha librería adaptada tiene problemas de funcionamiento. La única parte positiva de esto es que para poder solucionar todos los problemas realicé gran cantidad de pruebas y correcciones, que sirvieron para después acortar la fase de pruebas que duró mucho menos de lo estimado (una cuarta parte).

Caso aparte es la fase de Materiales Didácticos, que ha durado más de lo planificado y la razón ha sido la falta de experiencia en el uso de la aplicación de manipulación de imágenes *GIMP*, si se hubiera tenido algo más de experiencia, está fase habría durado lo planificado.

## *3.8.- Presupuestos*

Este apartado sirve para describir el presupuesto y coste económico del proyecto, diferenciando entre el coste de la plantilla, el hardware y el software.

El proyecto ha sido realizado por una sola persona, por lo que el coste del hardware y del software es unitario, si hubiera habido más personal, estos costes se multiplicarían ya que harían falta más ordenadores, más software instalado en cada ordenadores (un ordenador y un programa de cada por persona).

### *3.8.1.- Coste del hardware de desarrollo*

Se ha utilizado un ordenador portátil para  para las tareas de desarrollo, consultas, pruebas, realización de documentación, etc. Los requerimientos son de un equipo de gama media con un procesador Intel i5 con 8 GB de RAM para poder mover de forma fluida *Android Studi*o

con el emulador de un dispositivo móvil y el programa de manipulación gráfica GIMP. El precio de un ordenador de estas características es de 549,95 €.

También se ha utilizado un teléfono móvil inteligente para instalar la aplicación y probar en un dispositivo real, además de en los virtuales de *Android Studio*. El único requisito que tiene que tener el dispositivo es tener una versión de Android 4.0 o superior. Se ha utilizado un teléfono *Samsung Galaxy S3* con un precio de 129,99 €.

Se ha utilizado un dispositivo de almacenamiento USB para copias de seguridad de 8 GB de capacidad con un precio de 3,95 €.

Para concluir, se han utilizado dispositivos de soporte magnético (6 unidades) para la documentación del Trabajo de Fin de Grado. Con un coste de 0,17 € por unidad, que hacen un total de 1,02 €.

El coste total de los equipos hardware utilizados, con su correspondiente porcentaje de utilización es:

| Equipo | Precio (€) | % utilización | Coste Real (€) |
|---|---|---|---|
| Ordenador portátil | 549,95 | 25 | 137,49 |
| Teléfono inteligente | 129,99 | 5 | 6,49 |
| Dispositivo de almacenamiento USB | 3,95 | 30 | 1,19 |
| Dispositivos de soporte magnético | 1,02 | 100 | 1,02 |
| **Total** | | | **146,19** |

*Tabla 3: Presupuesto Hardware*

### *3.8.2.- Coste del software de desarrollo*

En este apartado analizo el precio de los elementos software que he utilizado para el desarrollo del proyecto, tanto para la generación de la documentación, codificación, tratamiento de imágenes, creación de diagramas, etc. Al ser parte imprescindible, hay que incluirlos en el presupuesto. Se ha tratado en la medida de lo posible utilizar software gratuito, cuando el software es gratuito, al no implicar coste, defino la utilización como del 100%. Para el sistema operativo Windows 10, se define una utilización igual que la del ordenador portátil.

| Programa | Precio (€) | % utilización | Coste Real (€) |
|---|---|---|---|
| Windows 10 Home | 94,95 | 25 | 23,74 |
| Android Studio | gratuito | 100 | 0 |
| Open Office | gratuito | 100 | 0 |
| Adobe Reader | gratuito | 100 | 0 |
| Mozilla Firefox | gratuito | 100 | 0 |
| Gimp | gratuito | 100 | 0 |
| Gantt Project | gratuito | 100 | 0 |
| DIA Diagram Editor | gratuito | 100 | 0 |
| Star UML | gratuito | 100 | 0 |
| **Total** | | | **23,74** |

*Tabla 4: Presupuesto Software*

Si se hubieran utilizado otros programas, el coste hubiera sido mucho mayor ya que por ejemplo el precio de *Adobe Photoshop CS 6* es de 580,80 € que sustituiría a *GIMP*, el precio de *Microsoft Office 2016 Hogar y Estudiantes* es de 113,00 € que substituiría a *Open Office*, *Enterprise Architect* tiene un precio anual de 45,00 € que sustituiría a *Star UML*.

### *3.8.3.- Coste de los servicios*

Para la realización del proyecto son necesarios varios servicios y que hay que incluir en el presupuesto. En la siguiente tabla muestro los servicios utilizados.

Para el servicio de conexión a Internet, se ha utilizado una conexión de ADSL que tiene un precio mensual de 49,99 € y como la duración del proyecto ha sido de 6 meses, el coste precio de la conexión a Internet es de 299,94 €.

El precio del servicio de imprenta se estima en 150 €, ya que hasta que no se haga uso de ello no se sabe el precio exacto.

Se utiliza un servicio de almacenamiento gratuito en la nube para copias de seguridad. Al ser gratuito defino la utilización como del 100% ya que no supone un coste.

| Servicio | Precio (€) | % utilización | Coste Real (€) |
|---|---|---|---|
| Conexión a Internet | 299,94 | 40 | 119,98 |
| Imprenta | 150,00 | 100 | 150,00 |
| Almacenamiento en la nube | gratuito | 100 | 0 |
| Licencia de desarrollo de Google | 25,00 | 100 | 25,00 |
| **Total** | | | **294,98** |

*Tabla 5: Presupuesto Servicios*

### *3.8.4.- Coste de los trabajadores*

En esta sección presento el coste que tienen los trabajadores del proyecto. Solo ha habido un trabajador que ha realizado todas las funciones, pero se va a distinguir entre las labores de análisis que las realizaría un analista que tiene un coste de 25 €/hora (fases de análisis y planificación), las labores de diseño que realizaría un diseñador con un coste de 20 €/hora (fases de diseño y material didáctico), las labores de programación que realizaría un programador con un coste de 15 €/hora (fase de implementación y pruebas) y las labores de documentación que realizaría un responsable de documentación con un coste de 10 €/hora (fase de documentación).

Voy a tomar la duración real del proyecto para hacer el cálculo del coste del personal.

| Puesto | Nº horas | Coste hora | Coste total |
|---|---|---|---|
| Analista | 30 | 25,00 | 750,00 € |
| Diseñador | 85 | 20,00 | 1.700,00 € |
| Programador | 306 | 15,00 | 4.590,00 € |
| Responsable de documentación | 165 | 10,00 | 1.650,00 € |
| **Total** | | | **8.690,00 €** |

*Tabla 6: Presupuesto Plantilla*

### *3.8.5.- Coste total de la aplicación*

El coste total de la aplicación, sumando todos los costes definidos hasta ahora es:

| Descripción | Coste |
|---|---|
| Coste del Hardware | 146,19 € |
| Coste del Software | 23,74 € |
| Coste de los servicios | 294,98 € |
| Coste de la plantilla | 8.690,00 € |
| **Total** | **9.154,91 €** |

*Tabla 7: Presupuesto Global*

De este presupuesto global, hay unos gastos fijos que son los elementos hardware y software y que se pagan una vez, aunque pueden variar en función de la utilización y periodo de amortización de cada elemento. El servicio de imprenta también es un gasto fijo, que se paga una única vez y no es variable. Pero hay otros gastos como el servicio de la conexión a Internet y los gastos de la plantilla que son variables y están en función de la duración del proyecto, cuanto más dure más grandes son los gastos, por lo que cuanto mejor se planifique el proyecto y menos problemas haya en su ejecución, más barato será el coste final. Sobre todo porque el mayor gasto son los salarios de la plantilla.

A este coste total, podría incluirse el precio de la matrícula del Trabajo de Fin de Grado.

## *3.9.- Ampliaciones*

Como toda aplicación software, una vez que está concluida y entregada al cliente, aparte del mantenimiento que forma parte del ciclo de vida de la aplicación, siempre se pueden hacer ampliaciones y mejoras para ofrecer más servicios y cubrir nuevas necesidades.

En este caso en concreto las ampliaciones más probables y razonables serían la inclusión de nuevos idiomas en los que se pueda utilizar la aplicación o añadir nuevas categorías y galerías de Bits.

Otra posible ampliación sería la inclusión de pequeños juegos que reforzaran el aprendizaje como por ejemplo juegos de memoria con cartas dadas las vueltas que cuando las presionas se dan la vuelta mostrando una imagen y hay que hacer parejas o que aparezcan varias imágenes en la pantalla y mediante un sonido se solicite que se encuentre uno de los bits de la aplicación.

# 4.- Estado del Arte

La finalidad de este apartado es poner en conocimiento del lector el estado actual en el que se encuentra el marco de trabajo, donde podremos ver los diferentes sistemas operativos y plataformas existentes, herramientas, entornos de desarrollo, la evolución que ha habido, que ventajas e inconvenientes tienen y el motivo de la elección que he realizado.

## *4.1.- Plataformas y Sistemas Operativos*

La elección de una plataforma de desarrollo enfocada en un Sistema Operativo concreto es la primera y más crucial elección que hay que tomar, ya que en función de esa decisión está el lenguaje de programación y herramientas a usar que son excluyentes unas de otras y en caso de querer tener una aplicación en todas las plataformas habría que emplear tiempo y esfuerzo en aprender y controlar cada una de ellas, por lo que al decantarte por una u otra plataforma te estás decantado por lo más importante de la aplicación que son los usuarios que te vas a dirigir.

En la actualidad podemos hacer referencia a tres grandes plataformas que son, *Android* de *Google*, *iOS* de *Apple* y *Windows Phone* de *Microsoft* (sucesor de *Windows Mobile*), aunque no son los únicos ya que a lo largo dela historia ha habido otros como *Symbian*, *Palm*, *BlackBerry*, *Linux Mobile* (*LiMo*), *Firefox OS* entre otras. Para ver claramente las características de algunas de estas plataformas, presento la siguiente tabla:

| Plataforma | Apple iOS 8 | Android 6.0 | Wndows Phone 8 | BlackBerry 10 | Firefox OS 2.2 |
|---|---|---|---|---|---|
| **Compañía** | Apple | Open Handset Alliance | Microsoft | BlackBerry | Mozilla Foundation |
| **Núcleo del SO** | Mac OS X | Linux | Windows NT | QNX | Linux |
| **Licencia de software** | Propietaria | Libre y abierto | Propietaria | Propietaria | Libre y abierto |
| **Año de lanzamiento** | 2007 | 2008 | 2010 | 1999 | 2013 |
| **Fabricante único** | Sí | No | No | Sí | No |
| **Variedad de dispositivos** | Modelo Único | Muy alta | Media | Baja | Muy baja |
| **Soporte memoria externa** | No | Sí | Sí | Sí | Sí |
| **Motor del navegador** | WebKit | WebKit/ Chromium (Blink) | Trident | WebKit | WebKit |
| **Tienda de Aplicaciones** | App Store | Google Play | Windows Marketplace | BalckBerry World | Firefox Marketplace |
| **Número de aplicaciones** | 800.000 (marzo 2013) | 800.000 marzo 2013) | 130.000 (enero 2013) | 100.000 (enero 2013) | ¿? |
| **Coste Publicar** | $99 / año | $25 una vez | $99 / año | Sin coste | Sin coste |
| **Otras tiendas sin supervisión** | No | Sí | No | Sí | Sí |
| **Familia CPU soportada** | ARM | ARM, MIPS, x86 | ARM | ARM | ARM, x86 |
| **Soporte 64 bits** | Sí | Sí | No | No | No |
| **Máquina Virtual** | No | Dalvik / ART | .net | No | Navegador Web |
| **Lenguaje de programación** | Swift, Objective-C, C++ | Java, C++ | C#, Visual Basic, C++ | C, C++, Java | HTML5, CSS, Javascript |
| **Plataforma de desarrollo** | Mac | Windows, Mac, Linux | Windows | Windows, Mac | Windows, Mac, Linux |
| **Entorno de Desarrollo** | Xcode | Android Studio | Visual Studio | Editor de textos como Eclipse | Editor de textos como Eclipse |
| **Multiusuario** | No | Sí | No | No | No |
| **Modo Invitado** | Sí | Sí | No | No | No |

*Tabla 8: Comparativa de Plataformas*

Además de todas estas características, donde podemos ver que la plataforma más versátil es Android ya que su licencia es libre y abierta, la variedad de dispositivos en los que está presente es muy alta, soporta memoria, externa, varias familias de CPU y arquitectura de 64 bits,

tiene varias tiendas de aplicaciones tanto oficiales como no oficiales, etc. Para poder elegir correctamente que plataforma utilizar, hay que ver la cuota de mercado de cada plataforma para poder llegar al mayor número posible de potenciales usuarios. Para ver la cuota de mercado, presento a continuación una figura con el porcentaje de teléfonos vendidos en todo el mundo a lo largo del tiempo, perteneciente a un estudio realizado por la empresa *Gartner Group*.

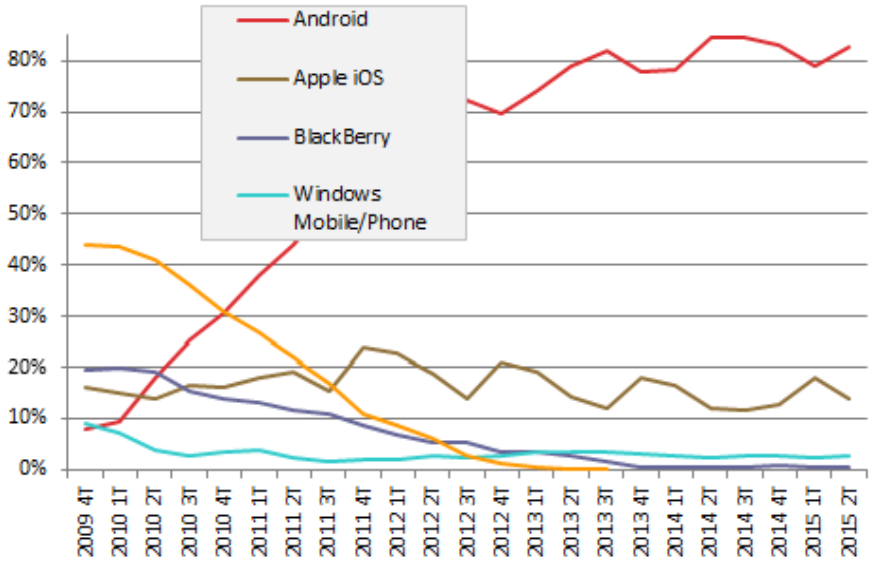


*Ilustración 12: Porcentaje de teléfonos inteligentes vendidos según su sistema operativo (fuente Gartner Group)*

Como podemos ver en la figura, solo aparecen representadas las 4 plataformas más importantes, otras como *Symbian* de Nokia (la línea Amarilla) ya ha desaparecido y *Firefox OS* al ser muy reciente, tampoco aparece en la figura. Analizando la figura vemos como *BalckBerry* está en una perdida continua de usuarios, *Windows Phone* tiene poca cuota de mercado por lo que parece que no termina de despegar y popularizarse, *iOS* tiene una cuota más o menos estable, de sus fieles consumidores, que puntualmente se incrementa con la salida de un nuevo dispositivo y finalmente *Android* que ha tenido un incremento espectacular de cuota de mercado, por encima de del 80%.

Debido a las características de cada plataforma y a las cuotas de mercado, además de por motivos puramente personales, **he elegido como plataforma de desarrollo Android**.

## *4.2.- Tipos de Desarrollos*

Ahora bien a la hora de desarrollar aplicaciones, existen varios métodos de desarrollo con los que puedes conseguir tener tu aplicación en diferentes plataformas. Los diferentes métodos de desarrollo son:

- **Desarrollo nativo**: Es el desarrollo único y exclusivo para cada tipo de plataforma, para tener una aplicación en varias plataformas habría que aprender la forma de desarrollo de cada uno de ellas y desarrollar por separado, con el aumento de esfuerzo y tiempo que conlleva.
- **Aplicación web móvil híbrida**: Este método se fundamenta en desarrollar la aplicación bajo estándares web que todas las plataformas reconocen como HTML5 y CSS 3, por lo que aplicación funcionaría en todas las plataformas y tendría un aspecto similar en todas ellas, además podremos usar las funciones del dispositivo por ser un sistema híbrido en vez de puramente web. Un ejemplo de este método es el framework *PhoneGap* (distribución de *Apache Cordova*). Tenemos dos opciones:
  - **Aplicación en local**: La aplicación web se encuentra dentro del dispositivo como una aplicación cualquiera de otro tipo, por lo que la instalación y actualizaciones, requieren el uso de la tienda de aplicaciones de la plataforma.
  - **Aplicación en Remoto**: El contenido de la aplicación web se encuentra en un servidor remoto por lo que los cambios y actualizaciones se realizan en el servidor y no en la aplicación
- **Cross-compilation**: Basa su funcionamiento en el IDE utilizado. Se desarrolla la aplicación bajo el código y los requerimientos del IDE y una vez terminada la aplicación, el IDE se encarga de transforma y exportarla a cada una de las plataformas que tenga disponibles y configuradas. Este metido de desarrollo te permite tener más funcionalidades y crear aplicaciones más potentes y exigentes ya que tienes la ayuda y soporte del IDE. Tiene como inconveniente que necesita utilizar los entornos de desarrollo nativos para compilar y ejecutar los proyectos para poder realizar pruebas y ensayos.

En función de estos métodos de desarrollo, lo ideal sería usar el de Cross-compilation o aplicación web híbrida para poder exportarlo a varias plataformas sin mucho esfuerzo, pero como uno de los objetivos fijados del Trabajo de Fin de Grado es el aprendizaje de la plataforma Android, **se realizará mediante el desarrollo nativo**.

Ahora llegaría el momento de analizar las diferentes posibles soluciones para el desarrollo nativo en Android, pero como de manera oficial solo existe una que es *Android Studio* ya que *Google* dejo de dar soporte al SDK para *Eclipse*, utilizaré *Android Studio* que se describirá más adelante.

## *4.3.- Plataforma Android*

Una vez que ya he escogido la plataforma y tipo de desarrollo que voy a realizar, voy a explicar un poco más las cualidades de Android que lo hacen especial y su funcionamiento.

- **Plataforma Abierta**: Es una plataforma de desarrollo libre basada en Linux (que es el Sistema Operativo) y de código abierto, lo que le proporciona una gran ventaja al poder usarlo y personalizarlo sin pagar *royalties*. Bueno esto es una verdad a medias, ya que

hay una parte libre compuesta por *Android Open Source Project* (*AOSP*) que desarrolla el núcleo y la parte funcional, por lo que solo con ello habría dispositivos funcionales, pero también hay otra parte cerrada y propietaria por parte de Google que son los *Google Mobile Sercvices* (*GMS*) y *Google Apps*. Esta parte cerrada y propietaria es la que da acceso a aplicaciones y servicios muy valorados por los usuarios y fabricantes como pueden ser la tienda de *Google Play Store*, los mapas de *Google Map*s, sincronización con los servicios de Google como *Gmail*, *Drive* y el resto de servicios y *APIs* de Google que cada vez son más importantes.

- **Adaptable a cualquier tipo de Hardware**: Android no ha sido diseñado exclusivamente para su uso en teléfonos y tabletas. Hoy en día lo podemos encontrar en multitud de equipos como relojes, cámaras, electrodomésticos, automóviles y gran variedad de sistemas empotrados que se basan en esta plataforma. Esto que es una gran ventaja, también supone un gran esfuerzo al desarrollador ya que la aplicación ha de funcionar correctamente en una vasta cantidad de dispositivos de características muy diferentes en cuanto a tamaño y resoluciones de pantalla, tipos de entradas, memoria, etc. Algo que contrasta con estrategias de otras compañías como *Apple*, que con *iOS* hay que desarrollar una aplicación para *iPhone* y otra para *iPad* y no hay más dispositivos posibles.
- **Portabilidad asegurada**: Las aplicaciones finales son desarrolladas en Java lo que asegura que podrán ser ejecutadas en cualquier tipo de CPU presente y futura gracias al concepto de Máquina Virtual.
- **Arquitectura basada en componentes inspirados en Internet**: Por ejemplo, el diseño de la interfaz de usuario se hace en XML, lo que permite que una misma aplicación se ejecute en un reloj con pantalla reducida o en un televisor.
- **Filosofía de dispositivo siempre conectado a Internet**: Muchas aplicaciones solo funcionan si disponemos de una conexión permanente a Internet. Por ejemplo, comunicaciones interpersonales o navegación con mapas.
- **Gran cantidad de servicios incorporados**: Disponemos de una amplia variedad de servicios como localización basada en GPS o redes, bases de datos con SQL, reconocimiento y síntesis de voz, navegador, multimedia...
- A**ceptable nivel de seguridad**: Los programas se encuentran aislados unos de otros gracias al concepto de ejecución dentro de una caja heredado de Linux. Además, cada aplicación dispones de una serie de permisos, que posibilitan y limitan su rango de acción (servicios de localización, acceso a Internet, acceso a memoria, llamadas, SMS...). Desde la versión 6.0 el usuario puede conceder o retirar permisos a las aplicaciones en cualquier momento y no solo cuando las instala o actualiza.
- **Optimizado para baja potencia y poca memoria**: En el diseño de Android se ha tenido en cuenta el Hardware especifico de los dispositivos móviles, por ejemplo, Android utiliza una Máquina Virtual ART (o Dalvik en versiones antiguas), que se trata de una implementación de Google de la máquina virtual de Java optimizada para dispositivos móviles.
- **Alta calidad de gráficos y sonido**: En el apartado de gráficos, disponemos de gráficos vectoriales suavizados, animaciones inspiradas en Flash, gráficos en 3D basados en

*OpenGL*, incorporación de los *codecs* y estándares más comunes de audio y vídeo incluyendo H.264 (AVC), MP3, AAC, etc.

### *4.3.1.- Los orígenes de Android*

Google adquiere Android Inc. en el año 2005. Se trataba de una pequeña compañía, recién creada, orientada a la producción de aplicaciones para terminales móviles. Ese mismo año empiezan a trabajar en la creación de una máquina virtual Java optimizada para móviles (Dalvik VM).

En el año 2007 se crea el consorcio Handset Alliance con el objetivo de desarrollar estándares abiertos para móviles. Está formado por Google, Intel, Texas Instruments, Motorola, T-Mobile, Samsung, Ericson, Toshiba, Vodafone, NTT DoCoMo, Sprint Nextel y otros. Una pieza clave de los objetivos de esta alianza es promover el diseño y difusión de la plataforma Android. Sus miembros se han comprometido a publicar una parte importante de su propiedad intelectual como código abierto bajo licencia Apache v2.0.

En noviembre de 2007 se lanza una primera versión del Android SDK. Al año siguiente aparece el primer móvil con Android (T-Mobile G1). En octubre, Google libera el código fuente de Android, principalmente bajo licencia de código abierto Apache (licencia GPL v2 para el núcleo). Ese mismo mes se abre *Android Market*, para la descarga de aplicaciones. En abril de 2009, Google lanza la versión 1.5 del SDK, que incorpora nuevas característica como el teclado en pantalla. A finales de 2009 se lanza la versión 2.0 y a lo largo de 2010, las versiones 2.1, 2.2 y 2.3.

Durante el año 2010, Android se consolida como uno de los sistemas operativos para móviles más utilizados, con resultados cercanos a iOS e incluso superando al sistema de Apple en EE.UU.

En el año 2011 se lanza la versión 3.x (Honeycomb), específica para tabletas, y la 4.0 (Ice Cream Sandwich), tanto para móviles como para tabletas. Durante ese año, Android se consolida como la plataforma para móviles más importante y alcanza una cuota de mercado superior al 50%.

En 2012, Google cambia su estrategia en su tienda de descargas *online*, reemplazando Android Market por Google Play Store, donde en un solo portal unifica tanto la descarga de aplicaciones como la de contenidos. Ese año aparecen las versiones 4.1 y 4.2 (Jelly Bean). Android mantiene su espectacular crecimiento y alcanza, a finales de año, una cuota de mercado del 70 %.

En 2013 se lanzan las versiones 4.3 y 4.4 (KitKat). En 2014 se lanza la versión 5.0 (Lollipop). A finales de ese año, la cuota de mercado de Android llega al 85 %. En octubre de 2015 ha aparecido la versión 6.0, con el nombre de Marshmallow.

En 2016 se lanza la siguiente versión, cuyo *preview* para programadores, Android N, está disponible desde marzo.

Cada versión de Android lleva asociado un nombre se elige en orden alfabético (en Ingles) y está asociado a un dulce que es su logotipo correspondiente.

En el **Anexo I** hay una explicación un poco más detallada de cada versión de Android.

### *4.3.2.- Arquitectura de Android*

Vamos a analizar la arquitectura en la que se basa Android. Esta arquitectura está basada en cuatro capas, que podemos ver en la siguiente figura:

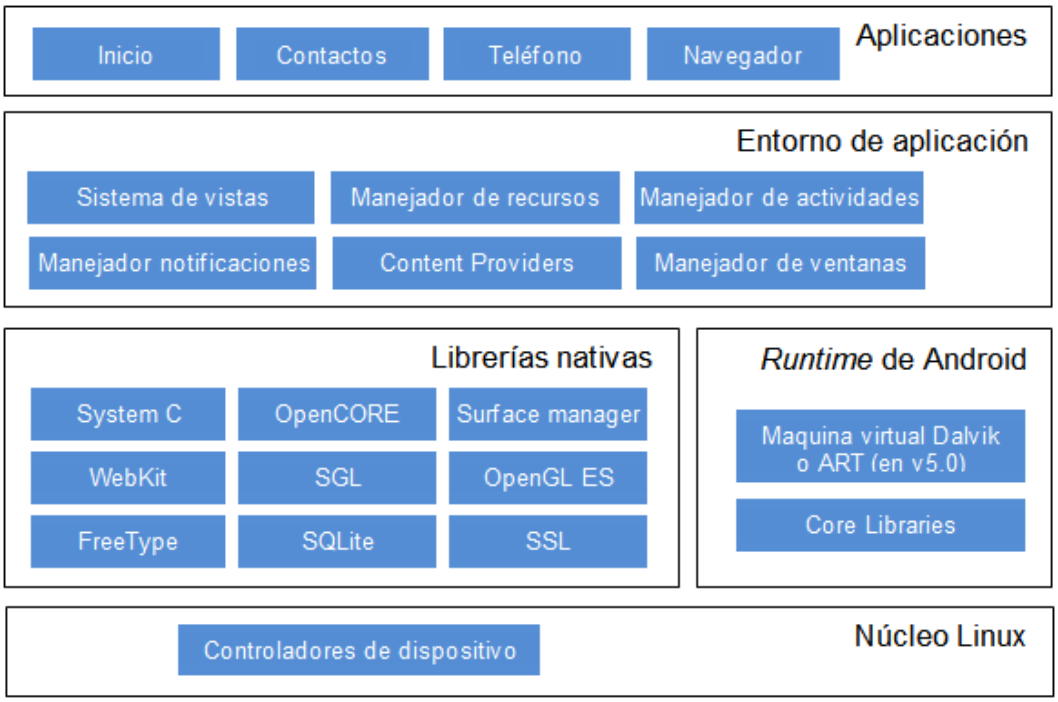


*Ilustración 13: Arquitectura de Android*

Voy a explicar cada capa:

1. **El núcleo Linux:** El núcleo de Android está formado por el sistema operativo Linux versión 2.6. Esta capa proporciona servicios como la seguridad, el manejo de la memoria, el multiproceso, la pila de protocolos y el soporte de drivers para dispositivos (cámaras de fotos, GPS, bluetooth, GMS...).

   Esta capa del modelo actúa como capa de abstracción entre el hardware y el resto de la pila. Por lo tanto, es la única que es dependiente del hardware y el desarrollador no tiene acceso a ella directamente pero ha de tenerla en cuenta.

   Se encarga entre otras cosas de:

   - Gestión de procesos: Gestiona los procesos en ejecución del dispositivo. Cada aplicación se ejecuta en un proceso que es una instancia de la máquina virtual.
   - Gestión de usuarios y permisos: La seguridad es un característica vital en un sistema donde se ejecutan multitud de aplicaciones diferentes de distintas procedencias y distribuidores, donde se manejan gran cantidad de datos de distinta índole con diferentes grados de sensibilidad y seguridad, desde puntuaciones en aplicaciones a datos bancarios pasando por fotos y contactos por ejemplo.

Mención aparte de aplicaciones maliciosas y malintencionadas. Esto se resuelve creando una cuenta de usuario para cada aplicación para gestionar los datos de la misma, otros usuarios podrán acceder a esos datos previa autorización de la aplicación.

- Gestión de memoria: Android gestiona la memoria disponible del dispositivo, de hecho fue diseñado para optimizar este recurso escaso, por lo que el desarrollador no tiene que preocuparse de la liberación de memoria de su aplicación ya que Android lo hará por él, esto no quita que el desarrollador deba preocuparse de la cantidad de memoria que utiliza su aplicación ya que si el usuario abre otra aplicación, la primera no tiene por qué cerrarse, puede pasar a segundo plano haciendo uso de memoria y cuando la memoria llegue al límite de uso óptimo, Android cerrará procesos. Esto no es predecible por el desarrollador por lo que es conveniente que tenga en cuenta el uso de memoria y estudia los ciclos de vida de las aplicaciones.

2. **Runtime de Android:** Está basado en el concepto de máquina virtual utilizado en Java. Dadas las limitaciones de los dispositivos donde ha de correr Android (poca memoria, procesador limitado y ahorro de energía), no fue posible utilizar una máquina virtual Java estándar. Google tomó la decisión de crear una nueva, la máquina virtual Dalvik, que respondiera mejor a estas limitaciones.

   Entre las características de la máquina virtual Dalvik que facilitan esta optimización de recursos se encuentra la ejecución de ficheros Dalvik ejecutables (*.dex*), formato optimizado para ahorrar memoria. Además, está basada en registros. Cada aplicación corre en su propio proceso Linux con su propia instancia de la máquina virtual Dalvik. Delega al kernel de Linux algunas funciones como *threading* y el manejo de la memoria a bajo nivel.

   A partir de Android 5.0 se reemplaza Dalvik por ART. Esta nueva máquina virtual consigue reducir el tiempo de ejecución del código Java hasta en un 33%.

   También se incluye en el *runtime* de Android el módulo *Core Libraries*, con la mayoría de las librerías disponibles en el lenguaje Java.

3. **Librerías Nativas:** Incluye un conjunto de librerías escritas en C/C++ usadas en varios componentes de Android. Están compiladas en código nativo del procesador. Muchas de las librerías utilizan proyectos de código abierto. Estas librerías pueden estar envueltas en capas superiores dela arquitectura para facilitar su uso por medio de APIs. Algunas de estas librerías son:

   - **System C Library**: Es una derivación de la librería BSD de C estándar (libc), adaptada para dispositivos embebidos basados en Linux.
   - **Media Framework**: Librería basada en OpenCORE de PacketVideo. Soporta codecs de reproducción y grabación de multitud de formatos de audio y vídeo e imágenes MPEG4, H.264, MP3, AAC, AMR, JPG y PNG.
   - S**urface Manager**: Maneja el acceso al subsistema de representación gráfica en 2D y 3D.

- **WebKit/Chromium**: Soporta el moderno navegador Web utilizado en el navegador Android y en vista Webview. En la versión 4.4, WebKit ha sido reemplazado por Chromium/Blink, que es la base del navegador Chrome de Google.
- **SGL**: Es el motor de gráficos 2D.
- **Librerías 3D**: Implementación basada en OpenGL ES 1.0 API. Las librerías utilizan el acelerador hardware 3D si está disponible, o el software altamente optimizado de proyección 3D.
- **FreeType:** Fuentes en bitmap y renderizado vectorial.
- **SQLite**: Es el motor de bases de datos relacionales disponible para todas las aplicaciones, es ligero y potente.
- **SSL**: Proporciona servicios de encriptación Secure Socket Layer (capa de conexión segura).

4. **Entorno de Aplicación:** Proporciona una plataforma de desarrollo libre para aplicaciones con gran riqueza e innovaciones (sensores, localización, servicios, barra de notificaciones, etc.).

   Esta capa ha sido diseñada para simplificar la reutilización de componentes. Las aplicaciones pueden publicar sus capacidades y otras pueden hacer uso de ellas (sujetas a las restricciones de seguridad). Este mismo mecanismo permite a los usuarios reemplazar componentes.

   Los servicios más importantes que incluye son:

   - **Views**: extenso conjunto de vistas, (parte visual de los componentes).
   - **Resource Manager**: proporciona acceso a recursos que no son en código.
   - **Activity Manager**: maneja el ciclo de vida de las aplicaciones y proporciona un sistema de navegación entre ellas.
   - **Notification Manager**: permite a las aplicaciones mostrar alertas personalizadas en la barra de estado.
   - **Content Providers**: mecanismo sencillo para acceder a datos de otras aplicaciones (como los contactos).

   Una de las mayores fortalezas del entorno de aplicación de Android es que se aprovecha el lenguaje de programación Java. El SDK de Android no acaba de ofrecer para su estándar todo lo disponible del entorno de ejecución Java (JRE), pero es compatible con una fracción muy significativa de este

5. **Aplicaciones:** Este nivel está formado por el conjunto de aplicaciones instaladas en una máquina Android. Todas las aplicaciones han de correr en la máquina virtual Dalvik para garantizar la seguridad del sistema.

   Normalmente las aplicaciones Android están escritas en Java. Para desarrollar aplicaciones en Java podemos utilizar el Android SDK. Existe otra opción consistente en

desarrollar las aplicaciones utilizando C/C++. Para esta opción podemos utilizar el Android NDK (Native Development Kit).

### *4.3.3.- El Modelo Vista Controlador de Android*

A la hora de desarrollar aplicaciones en la plataforma Android tenemos que tener en cuenta que hay que desarrollar conforme al patrón Modelo-Vista-Controlador, que es el especificado y obligado en la plataforma. El Modelo-Vista-Controlador (MVC) es patrón de arquitectura software cuya principal característica es separar la capa de datos, la lógica de negocio y la interfaz de usuario. Para ello propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador. Este patrón busca facilitar el desarrollo de aplicaciones y su posterior mantenimiento haciendo uso de la reutilización de código y la separación de conceptos.

Una representación gráfica de este modelo es la siguiente:

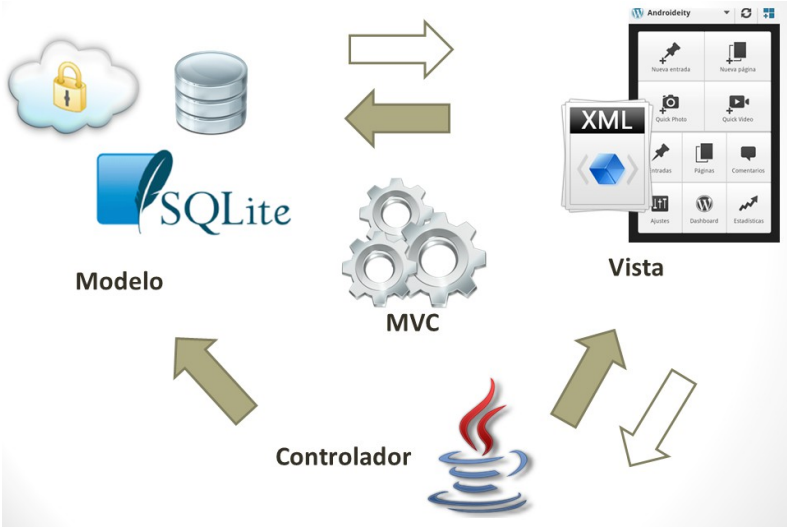


*Ilustración 14: Modelo-Vista-Controlador*

De esta forma podemos seccionar de forma más fácil nuestro equipo de trabajo y dedicarnos a desarrollar nuestros componentes de tal forma que construyamos módulos o librerías con funcionalidades específicas que incluso podríamos reutilizar en proyectos posteriores y no simplemente en el proyecto actual. Para lograr esto, el diseño de la arquitectura de nuestra aplicación juega un papel importante y la capacidad de abstracción que tengamos desarrollada.

Cada uno delos componentes son los siguientes:

- **Modelo**: Es la representación de la información con la que operará nuestra aplicación, por tanto gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio). Envía a la *vista* aquella parte de la información que en cada momento se le solicita

para que sea mostrada (típicamente a un usuario). Las peticiones de acceso o manipulación de información llegan al *modelo* a través del *controlador*. En Java, el modelo viene siendo análogo a los *beans* que tienen la particularidad de ser reutilizables y nos ayudan a cumplir con el proverbio de oro ***"Don´t Repeat Yourself" (DRY)*** haciendo a nuestras aplicaciones escalables. En esta parte del modelo también juega la decisión de qué modelo para almacenar información se utilizará. Algunos ejemplos de modelos para almacenar información son base de datos (la plataforma Android da soporte al motor SQLite) o Web Services (con los que almacenamos la información en un servidor remoto). El modelo que elijas depende obviamente de las necesidades de tu aplicación.

- **Vista**: La *vista* presenta el *modelo* (información y lógica de negocio) en un formato adecuado para interactuar (normalmente la interfaz de usuario), por lo que necesita que el *modelo* le proporcione la información que debe presentar como salida. En Android, la *vista* es interfaz con la que va a interactuar el usuario. En Android, las interfaces (*layouts*) las construimos en XML. Esta parte es parecida a lo que hacemos en el desarrollo web con los CSS.
- **Controlador**: Responde a eventos (usualmente acciones del usuario) e invoca peticiones al *modelo* cuando se hace alguna solicitud sobre la información (por ejemplo, rellenar un campo de un formulario o escoger una opción de una lista). También puede enviar comandos a su *vista* asociada si se solicita un cambio en la forma en que se presenta el *modelo* (por ejemplo, desplazamiento o scroll por una lista o por los diferentes registros de una base de datos), por tanto se podría decir que el *controlador* hace de intermediario entre la *vista* y el *modelo*. O dicho de otra forma, el *controlador* son todas esas clases (*activities*) que nos ayudarán a darle vida a las interfaces y que nos permiten mostrar información al usuario y recogerla si es preciso. Estos controladores se programan en lenguaje Java y marcan el funcionamiento de la aplicación.

El flujo de trabajo en este tipo de patrón es el siguiente:

1. El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace, etc.)
2. El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback.
3. El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario.
4. El controlador llama a la vista correcta que obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se reflejan los cambios que generó su acción del paso 1.
5. Se muestra la nueva vista y la interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

Por todas estas características y funcionamiento este modelo es muy bueno al separar los conceptos y ayuda a mantener los proyectos bien ordenados y limpios para su desarrollo, mantenimiento y ampliaciones.

### *4.3.4.- Niveles de API*

Como hemos visto, dentro de la plataforma Android existen varias versiones, desde la primera versión comercial Android 1.0 (Apple Pie) hasta las más novedosas Android 6.0 Marshmallow y Android Nougat. Cada nueva versión de Android o cada cambio importante dentro de una versión lleva un nivel de API diferente.

Cada nivel de API que se distribuye, tiene las características de los niveles anteriores (retrocompatibilidad) más nuevas características y funcionalidades. Debido a este sistema de actualizaciones, si en nuestra aplicación requerimos de alguna característica especial que solo esté disponible a partir de determinada versión, los usuarios con versiones anteriores no podrán instalar la aplicación. Por lo tanto, es recomendable seleccionar la menor versión posible que nuestra aplicación pueda soportar para que llegue al mayor número de usuarios posibles. Como explicación práctica, si nuestra aplicación usa el motor de animaciones de propiedades, tendremos que utilizar la versión de Android 3.0 (Nivel de API 11) ya que es la primera que lo incorpora, el problema estaría en que la aplicación no podría ser instalada en dispositivos que tengan una versión anterior a la 3.0.

Para facilitarnos la decisión de que versión y nivel de API usar, podemos consultar el nivel de utilización de cada versión que se nos proporciona en la web de desarrolladores de Android (https://developer.android.com/about/dashboards/index.html). En el momento de realizar esta documentación, los datos que arroja esta información son:

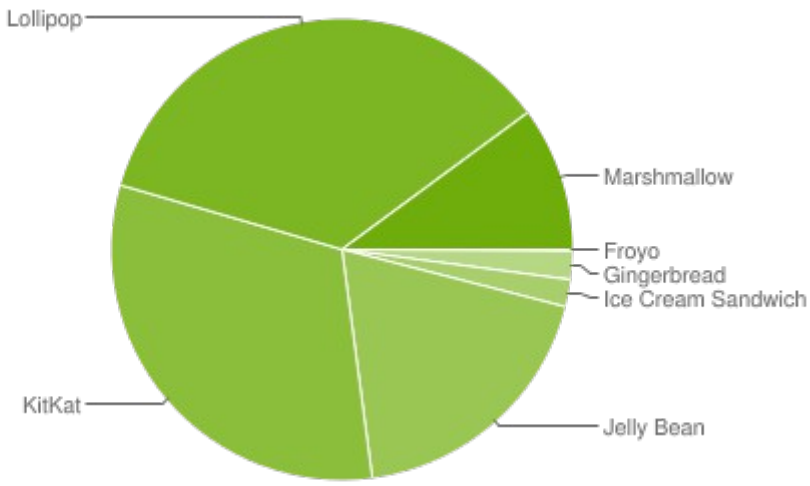


*Ilustración 15: Dispositivos Android, según la plataforma instalada, que han accedido a Google Play Store el 6 de Junio de 2016 y los 7 días anteriores. Las versiones con porcentajes inferiores al 0,1% no se muestran.*

Además del gráfico, también nos dan esta información en forma de tabla para que se pueda ver de diferentes formas:

| Versión | Nombre | Nivel de API | Utilización |
|---|---|---|---|
| v2.2 | Froyo | 8 | 0.1% |
| v2.3.3 – 2.3.7 | Gingerbread | 10 | 2.0% |
| v4.0.3 – 4.0.4 | Ice Cream Sandwich | 15 | 1.9% |
| v4.1.x | Jelly Beam | 16 | 6.8% |
| v4.2.x | | 17 | 9.4% |
| v4.3.x | | 18 | 2.7% |
| v4.4 | KitKat | 19 | 31.6% |
| v5.0 | Lollipop | 21 | 15.4% |
| v5.1 | | 22 | 20.0% |
| v6.0 | Marshmallow | 23 | 10.1% |

*Tabla 9: Dispositivos Android, según la plataforma instalada, que han accedido a Google Play Store el 6 de Junio de 2016 y los 7 días anteriores. Las versiones con porcentajes inferiores al 0,1% no se muestran.*

En función de estos datos, podemos ver que más del 75% de dispositivos utilizan la versión 4.4 o superior y que por debajo de la versión 4.0.x apenas ningún dispositivo la usa (solo el 2.1%). Como dato significativo, no hay ningún dispositivo con la versión 3.X, está versión fue diseñada específicamente para tabletas cuando surgieron, pero con la aparición de la versión 4, se eliminó la división existente entre tabletas y móviles.

Además de los datos de utilización de las versiones, en la página mencionada también aparecen datos de utilización referentes a las pantallas de os dispositivos, tanto en tamaño como en resoluciones, estos datos son:

Tamaño de las pantallas:

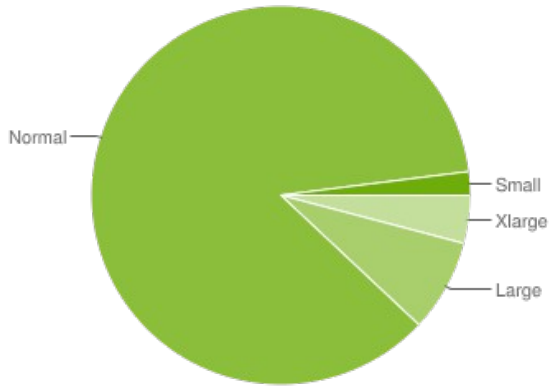


*Ilustración 16: Dispositivos Android, según el tamaño de la pantalla, que han accedido a Google Play Store el 6 de Junio de 2016 y los 7 días anteriores.*

Resoluciones de las pantallas:

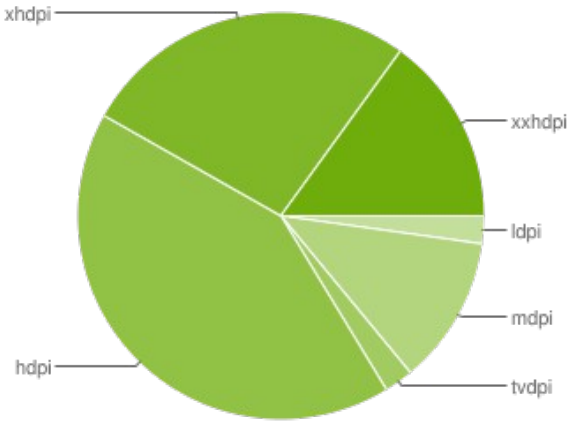


*Ilustración 17: Dispositivos Android, según la resolución de la pantalla, que han accedido a Google Play Store el 6 de Junio de 2016 y los 7 días anteriores*

Los datos en forma de tabla son:

| | **ldpi** | **mdpi** | **tvdpi** | **hdpi** | **xhdpi** | **xxhdpi** | **Total** |
|---|---|---|---|---|---|---|---|
| **Small** | 2.0% | | | | | | 2.0% |
| **Normal** | | 4.2% | 0.1% | 41.1% | 25.6% | 15.0% | 86.0% |
| **Large** | 0.2% | 4.5% | 2.2% | 0.5% | 0.5% | | 7.9% |
| **XLarge** | | 3.1% | | 0.3% | 0.7% | | 4.1% |
| **Total** | 2.2% | 11.8% | 2.3% | 41.9% | 26.8% | 15.0% | |

*Tabla 10: Dispositivos Android, según el tipo de la pantalla, que han accedido a Google Play Store el 6 de Junio de 2016 y los 7 días anteriores*

Viendo estos datos sabemos que las pantallas de tamaño normal son las predominantes con el 86% del total y que las resoluciones elevadas son mayoritarias ya que entre hdpi, xhdpi y xxhdpi suman un 83.7%

Para terminar de tener el aspecto de las pantallas claro, tenemos que ir a la sección dedicada a ello en la web de desarrolladores de Android para ver que hacen referencia con los las nomenclaturas. Dicha web https://developer.android.com/guide/practices/screens_support.html es y en resumen los datos son los siguientes:

Resoluciones (densidad):

- ldpi (low) ~ 120 dpi.
- mdpi (medium) ~ 160 dpi.
- Tvdpi (television) ~ 213 dpi.
- hdpi (high) ~ 240 dpi.
- xhdpi (extra-high) ~ 320 dpi.
- xxhdpi (extra-extra-high) ~ 480 dpi.
- xxxhdpi (extra-extra-extra-high) ~ 640 dpi.

Tamaños de pantalla:

- Small screen: Pantallas con un tamaño mínimo de 426 x 320 dp.
- Normal screen: Pantallas con un tamaño mínimo de 470 x 320 dp.
- Large screen: Pantallas con un tamaño mínimo de 640 x 480 dp.
- XLarge screen: Pantallas con un tamaño mínimo de 960 x 720 dp.

Está disponible el siguiente gráfico con las resoluciones y tamaños de pantalla para que nos hagamos una idea de a que tamaños de pantallas en pulgadas corresponden los tamaños definidos en Android y sus resoluciones:

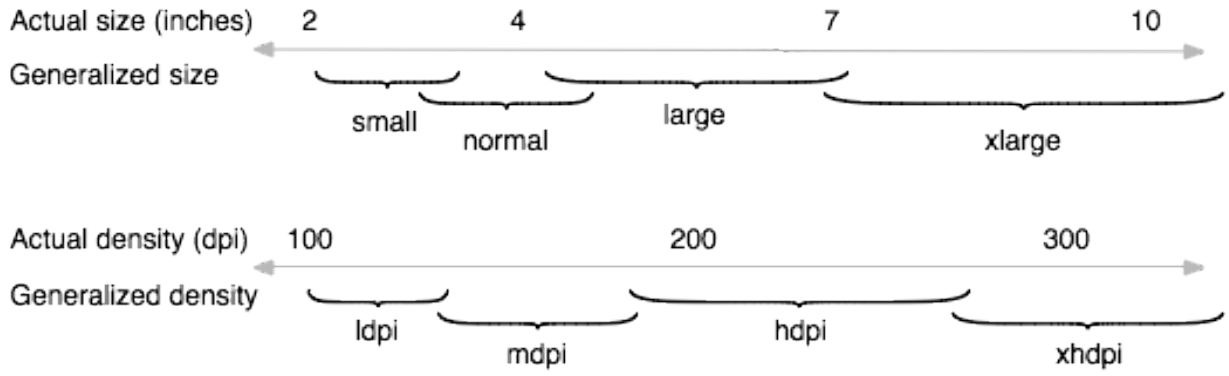


*Ilustración 18: Rangos de Pantallas*

## *4.3.5.- Procesos en Android*

En la mayoría de los casos, una aplicación Android se ejecuta en su propio proceso de Linux. Este proceso es creado para la aplicación cuando la arrancamos y seguirá corriendo hasta que no sea necesario y el sistema reclame recursos para otras aplicaciones, entonces le quitará los recursos a la aplicación antigua para dárselos a la nueva.

De esta forma, el tiempo de vida de un proceso en Android es controlado por el sistema operativo, basándose en las necesidades del usuario, los recursos disponibles, etc. Si tenemos una aplicación que está consumiendo muchos recursos y arrancamos otra nueva aplicación, el sistema operativo probablemente le diga a la aplicación que se queda en segundo plano que libere todo lo que pueda, y si es necesario la cerrará.

En Android los recursos son normalmente muy limitados y por eso el sistema operativo tiene más control sobre las aplicaciones que en programas de escritorio.

Para determinar qué procesos eliminar ante un escenario dónde el dispositivo tenga poca batería u otros escenarios en los que sea importante administrar los recursos, Android les asigna una prioridad a cada uno de ellos basándose en la siguiente jerarquía:

1. **Foreground Proces**s: Es la aplicación que contiene la actividad ejecutada en primer plano en la pantalla del usuario y con la cual está interactuando el usuario en ese instante (se ha llamado a su método onResume()). Por norma general, habrá muy pocos procesos de este tipo corriendo a la vez en el sistema y son aquellos que se eliminarán como última opción si la memoria es tan baja que ni matando al resto de procesos tenemos los recursos necesarios.
2. **Visible Process**: Es un proceso que aloja una Activity que no se está ejecutando en primer plano (es decir, su método onPause() ha sido llamado). Un ejemplo puede ser la aplicación de correo en la cual demos click en algún enlace de interés que nos lance el navegador, este pasaría a ser el *Foreground Process* dejando a la aplicación de correo en el concepto de *Visible Process*. Este tipo de procesos se cerrarán únicamente cuando el sistema no tenga los recursos necesarios para mantener corriendo todos los procesos que estén en primer plano.
3. **Service Process**: Son aquellos que corren cuando un *Service* ha sido invocado. Estos procesos hacen cosas en segundo plano que normalmente son importantes para el usuario (conexión con servidores, actualización del GPS, reproductor de música, etc.), el sistema nunca va a liquidar un servicio a menos que sea necesario para mantener vivos todos los *Visible* y *Foreground Process*.
4. **Background Process**: Es un proceso que contiene una Activity que actualmente no es visible por el usuario y que ya no tienen demasiada importancia. Por ejemplo, los programas que arrancó el usuario hace tiempo y no los ha vuelto a usar, pasan a estar en *Background*. Por eso es importante que cuando nuestra aplicación pase a *Background*, el sistema libere, en la medida de lo posible, todos los recursos que pueda para que su rendimiento sea óptimo.
5. **Empty Process**: Es un proceso que no aloja ningún tipo de componente. Su razón de ser es el de tener una caché disponible para la próxima aplicación que lance el usuario. Es común que el sistema elimine este tipo de procesos con frecuencia para así poder obtener memoria disponible.

Con todo este proceso vemos que realmente el sistema operativo controla el ciclo de vida de los procesos y lo que lo diferencia fundamentalmente con respecto a otros sistemas operativos donde el ciclo de vida lo controla mayoritariamente el usuario.

Una vez explicados y entendidos los procesos en Android vamos a ver algo que está relacionado con ellos, el ciclo de vida de las actividades.

### *4.3.6.- Ciclo de vida de la actividad*

Como hemos visto en el apartado anterior, el sistema operativo es el encargado de pausar, parar o destruir nuestra aplicación según las necesidades de recursos del dispositivo, aun así, nosotros como desarrolladores debemos aprender a controlar todos estos eventos para hacer nuestras aplicaciones robustas y mejorar el rendimiento de los teléfonos. Este ciclo de vida de Android es una gran diferencia respecto a otros sistemas operativos donde es el usuario o la aplicación la que controla el ciclo de vida, no el sistema operativo.

Una aplicación en Android está formada por un conjunto de elementos básicos de interacción con el usuario, conocidos como actividades. Además de varias actividades, una aplicación también puede contener servicios. El ciclo de vida de los servicios se estudiará en el próximo apartado. Son las actividades las que realmente controlan el ciclo de vida de las aplicaciones, dado que el usuario no cambia de aplicación, sino de actividad. El sistema mantiene una pila con las actividades previamente visualizadas, de forma que el usuario puede regresar a la actividad anterior pulsando la tecla “retorno”.

Una aplicación Android corre dentro de su propio proceso Linux. Este proceso se crea con la aplicación y continuará vivo hasta que ya no sea requerido y el sistema reclame su memoria para asignársela a otra aplicación.

Una característica importante, y poco usual, de Android es que la destrucción de un proceso no es controlada directamente por la aplicación, sino que es el sistema el que determina cuándo destruir el proceso. Lo hace basándose en el conocimiento que tiene de las partes de la aplicación que están corriendo (actividades y servicios), en la importancia de dichas partes para el usuario y en cuánta memoria disponible hay en un determinado momento.

Si tras eliminar el proceso de una aplicación, el usuario vuelve a ella, se crea de nuevo el proceso, pero se habrá perdido el estado que tenía esa aplicación. En estos casos, será responsabilidad nuestra como programadores almacenar el estado de las actividades, si queremos que cuando sean reiniciadas conserven su estado.

Como vemos, Android es sensible al ciclo de vida de una actividad; por lo tanto, necesitas comprender y manejar los eventos relacionados con el ciclo de vida si quieres crear aplicaciones estables.

Una actividad en Android puede estar en uno de estos cuatro estados:

- **Activa** (*Running*): La actividad está encima de la pila, en la parte superior, por lo que es visible y tiene el foco.
- **Visible** (*Paused*): La actividad es visible pero no tiene el foco. Se alcanza este estado cuando otra actividad pasa a activa con alguna parte transparente o que no ocupa toda la pantalla. Cuando una actividad está tapada por completo, pasa a estar parada.

- **Parada** (*Stopped*): Cuando la actividad no es visible. El programador debe guardar el estado de la interfaz de usuario, preferencias, etc.
- **Destruida** (*Destroyed*): Cuando la actividad termina al invocarse el método *finish()*, o es matada por el sistema.

Cada vez que una actividad cambia de estado se van a generar eventos que podrán ser capturados por ciertos métodos de la actividad. A continuación se muestra un esquema que ilustra los métodos que capturan estos eventos, procedente de la documentación oficial de Android Developers.

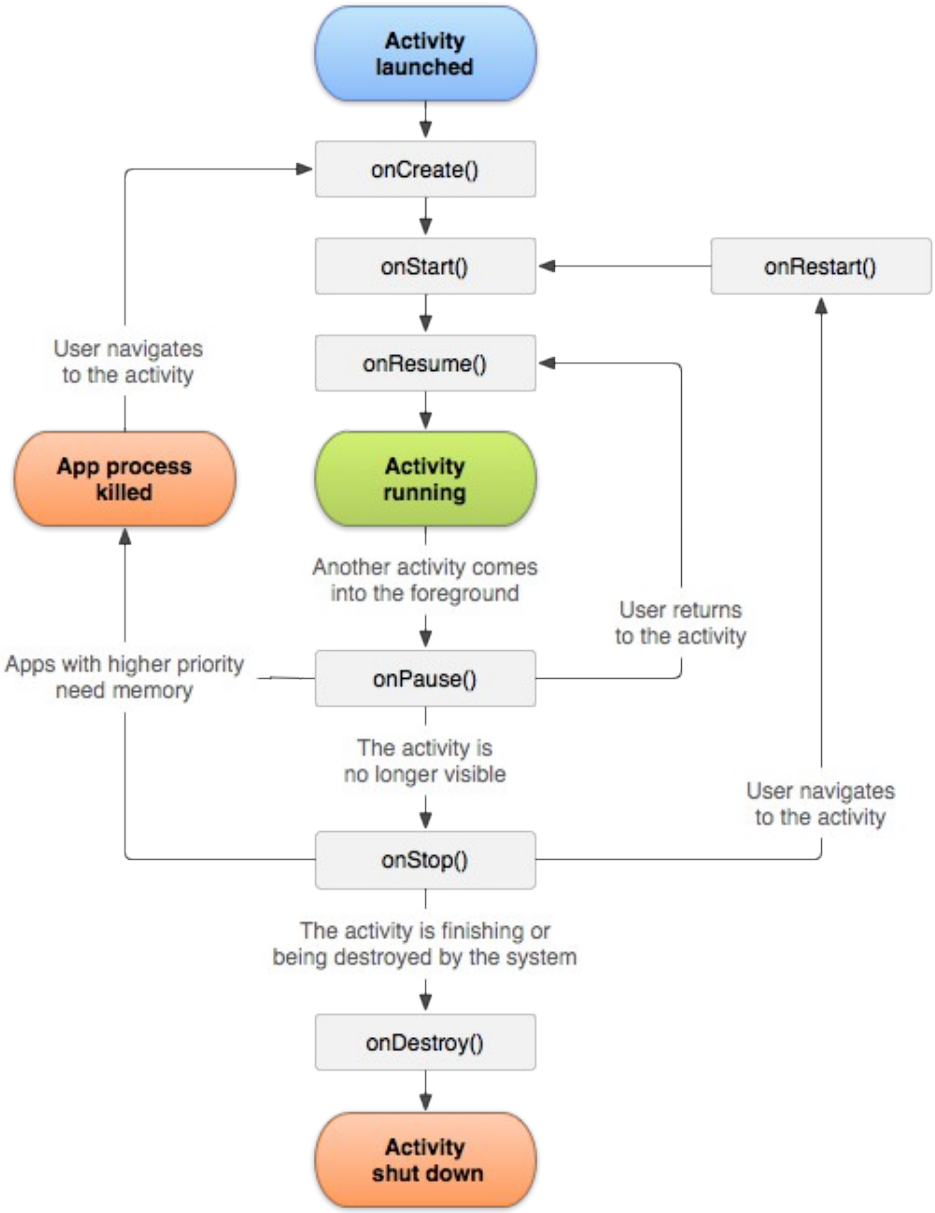


*Ilustración 19: Ciclo de vida de la actividad*

Paso a explicar cada uno de los métodos:

- **onCreate()**: Se dispara cuando la *Activity* es llamada por primera vez, en su creación. Aquí es donde debemos crear la inicialización normal de la aplicación, crear vistas (interfaz de usuario), hacer los *bind* de los datos, etc. Este método te da acceso al estado de la aplicación cuando se cerró, esto es porque puede recibir información del estado de la actividad (en una instancia de la clase *Bundle*), por si se reanuda desde una actividad que ha sido destruida y vuelta a crear. Después de esta llamada siempre se llama al onStart().
- **onStart()**: Se ejecuta cuando la *Activity* está a punto de ser mostrada en la pantalla del dispositivo del usuario.

- **onResume()**: Se ejecuta una vez que la Activity ha terminado de cargarse en el dispositivo y va a empezar a interactuar con el usuario. Es un buen lugar para lanzar las animaciones y la música. Cuando el usuario ha terminado de utilizarla es cuando se llama al método onPause().
- **onPause()**: Se ejecuta cuando el sistema arranca una nueva *Activity* que necesitará los recursos del sistema centrados en ella, es decir, está actividad va a ser lanzada a segundo plano en favor de otra que se pondrá en primer plano. Hay que procurar que la llamada a este método sea rápida ya que hasta que no se termine su ejecución no se podrá arrancar la nueva *Activity*. Después de esta llamada puede venir un onResume() si la *Activity* que haya ejecutado el onPause() vuelve a aparecer en primer plano o un onStop() si se hace invisible para el usuario. Es el lugar adecuado para detener animaciones, música o almacenar los datos que estaban en edición.
- **onStop()**: Se ejecuta cuando la *Activity* ya no es visible para el usuario porque otra *Activity* ha pasado a primer plano. Si hay muy poca memoria, es posible que la actividad se destruya sin llamar a este método. Si vemos el diagrama, después de que se ha ejecutado este método nos quedan tres opciones: ejecutar el onRestart() para que la *Activity* vuelva a aparecer en primer plano, que el sistema elimine este proceso porque otros procesos requieran memoria o ejecutar el onDestroy() para apagar la aplicación.
- **onRestart()**: Se ejecuta cuando la *Activity* ha sido parada (con onStop()), y se quiere volver a utilizar y visualizar. Si vemos el diagrama, se puede ver que después de un onStop() se puede ejecutar el método onRestart() e inmediatamente llama a un onStart().
- **onDestroy()**: Se llama antes de que la *Activity* sea completamente destruida, después de la llamada finaliza la *Activity*. Esto pasa por los requerimientos de memoria que tenga el sistema o porque de manera explícita el usuario manda a llamar este método. Si quisiéramos volver a ejecutar la *Activity* se arrancaría un nuevo ciclo de vida. Si hay muy poca memoria disponible, es posible que la *Activity* se destruya sin llamar a este método.

Espero que con esta explicación haya quedado claro el ciclo de vida las *Activities* de Android.

## *4.3.7.- Ciclo de vida del servicio*

Las aplicaciones además de estar formadas por una serie de A*ctivities* que permiten interacciones con el usuario por medio de una serie de elementos, disponen de otros tipos de componentes, como los *Services* que vamos a estudiar en este apartado.

Los servicios son componentes de las aplicaciones que son la mejor opción a utilizar cuando necesitamos que ciertas tareas se ejecuten en segundo plano, por debajo de las actividades y que continúen ejecutándose aunque se cambie de actividad y que además no requieran interacción con el usuario. Un servicio puede estar controlado desde una actividad o estar en ejecución indefinidamente.

La plataforma Android ofrece una gran cantidad de servicios predefinidos, disponibles típicamente a través de la clase *Manager*. De esta manera, en nuestras actividades podremos acceder a estos servicios a través del método *getSystemService()*.

Por otro lado, si necesitamos utilizar servicios propios, estos deben ir declarados en el archivo *AndroidManifest.xml*.

Un caso común es que el servicio y la actividad que la llama estén estrechamente relacionados. Bajo este escenario, el objeto *IBinder* nos servirá para trabajar sobre la clase que ofrece el servicio que estemos utilizando.

Los servicios en Android cumplen una doble función:

- La primera de las funciones es que indica al sistema que el elemento que estamos creando se va a ejecutar en segundo plano y normalmente por un periodo de tiempo largo. Este tipo de servicios se inicializan mediante el método *startService()*, cuya característica es que le dice al sistema que lo ejecute de forma indefinida hasta que alguien indique lo contrario, deteniéndolo con *stopService()*. Dentro de esta función, se engloban los servicios propios que desarrollemos para nuestras aplicaciones.
- La segunda función permite que nuestra aplicación se comunique con otras aplicaciones, para conseguir esto, debemos ofrecer ciertas funciones para que puedan ser llamadas desde otras aplicaciones. Estos servicios se inician mediante el método *bindService()*, que permite establecer una conexión con el servicio e invocar algunos de los métodos que ofrecemos.

Todos los servicios, independiente de la función que cumplan de las vistas hace un momento, son creados por el sistema mediante el método *onCreate()* del servicio en cuestión.

Los servicios pueden implementar desde trabajos y comportamientos muy simples, escritos en muy pocas líneas de código, hasta ejecuciones más complicadas como invocar a servicios remotos por medio de *APIs* o interfaces.

Los servicios, al igual que el resto de componentes de una aplicación, se ejecuta en el hilo principal del proceso dela aplicación. Debido a esto, si el servicio en su ejecución necesita un uso intensivo dela CPU o tarda en realizar ciertas operaciones como uso de redes o carga de datos, animaciones, etc. puede quedar bloqueado o ralentizar la actividad. Debido a esto y para no entorpecer a la actividad, lo mejor es crear un hilo diferente para ejecutar el servicio y que realice sus acciones correspondientes. Por ejemplo, podemos utilizar la clase *IntentService* para lanzar el servicio en su propio hilo.

En lo referente al ciclo de vida de los servicios, hay dos ciclos de vida dependiendo de cómo han sido creados, si con *startService()* o con *bindService()* y se representan con los siguientes esquemas:

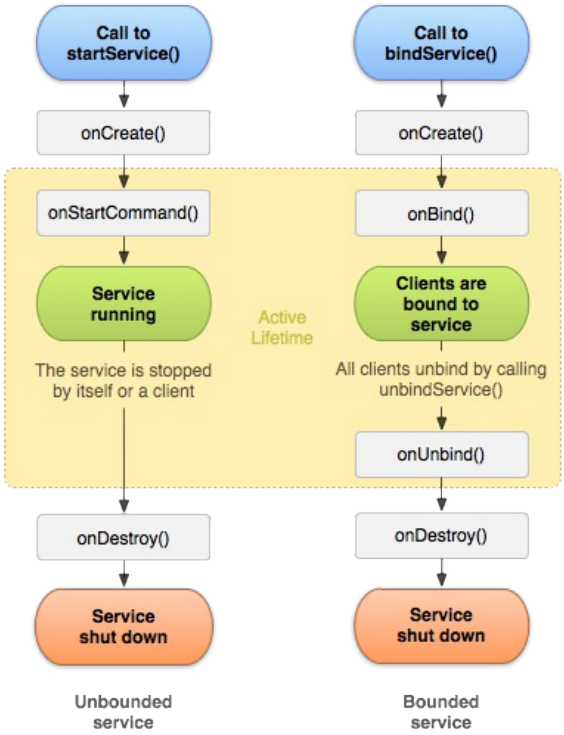


*Ilustración 20: Ciclo de vida de los servicios*

Si el servicio es iniciado mediante *starService()*, el sistema comenzará creándolo y llamando a su método *onCreate()*. A continuación llamará a su método *onStartCommand()* con los argumentos necesario. El servicio continuará en ejecución hasta que sea invocado el método *stopService()* o *stopSelf()*.

Si se producen varias llamadas a *startService()* no supondrá la creación de varios servicios, aunque sí que se realizarán múltiples llamadas a *onStartCommand()*. No importa cuántas veces el servicio haya sido creado, parará con la primera invocación de *stopService()* o *stopSelf()* que se realice. Sin embargo, podemos utilizar el método *stopSelf(int startID)* para asegurarnos que el servicio no parará hasta que todas las llamadas hayan sido procesadas.

Cuando se inicia un servicio para realizar alguna tarea en segundo plano, el proceso donde se ejecuta podría ser eliminado ante una situación de baja memoria. Podemos configurar la forma en que el sistema reaccionará ante esta circunstancia según el valor que devolvamos en *onStartCommand()*. Existen dos modos principales: devolveremos *START_STICKY* si queremos que el sistema trate de crear de nuevo el servicio cuando disponga de memoria suficiente y START_NOT_STICKY si queremos que el servicio sea creado de nuevo solo cuando llegue una nueva solicitud de creación.

Teniendo en cuenta que los servicios pueden estar largo tiempo en ejecución, el ciclo de vida del proceso que contiene nuestro servicio es un asunto de gran importancia. Conviene aclarar que en situaciones donde el sistema necesite memoria, conservar un servicio siempre será menos prioritario que la actividad visible en pantalla, aunque más prioritario que otras actividades en segundo plano. Dado que el número de actividades visibles es siempre reducido, un servicio solo será eliminado en situaciones de extrema necesidad de memoria. Por otra parte, si una actividad visible está conectada a un servicio, el servicio también será considerado como

visible, siendo tan prioritario como la actividad. En el caso de un proceso que contenga varios componentes, por ejemplo una actividad y un servicio, su prioridad se obtiene como el máximo de sus componentes.

Si el servicio es iniciado mediante *bindService()* para obtener una conexión persistente, si dicho servicio no está en ejecución, será creado (siempre que el flag BIND_AUTO_CREATE esté activo), llamándose al método *onCreate()* y a continuación se llamará al método *onBind()* que devolverá un objeto *IBinder* a través del cual se podrá establecer una comunicación entre cliente y servicio. Esta comunicación se establece por medio de una interfaz escrita en AIDL, que permite el intercambio de objetos entre aplicaciones que corren en procesos separados. El servicio permanecerá en ejecución tanto tiempo como la conexión esté establecida, independientemente de que se mantenga o no la referencia al objeto *IBinder.*

También es posible diseñar un servicio que pueda ser arrancado de ambas formas (*startService()* y *bindService()*). Este servicio permanecerá activo si ha sido creado desde la aplicación que lo contiene o si recibe conexiones desde otras aplicaciones.

### *4.3.8.- Fragments*

Un *Fragment* es un trozo o fragmento de una actividad, además un *fragment* tiene propio ciclo de vida y su representación gráfica o *layout*.

Gracias a los *fragments* podemos lograr las máximas de los buenos programadores de modularidad y reusabilidad, ya que los *fragments* se pueden programar una vez y después rehusar el código para que se adapte a distintos tipos de dispositivos.

Un *fragment* siempre estará dentro de una actividad, por lo que el ciclo de vida del *fragment* dependerá de la actividad. El ciclo de vida de un fragment es el siguiente:

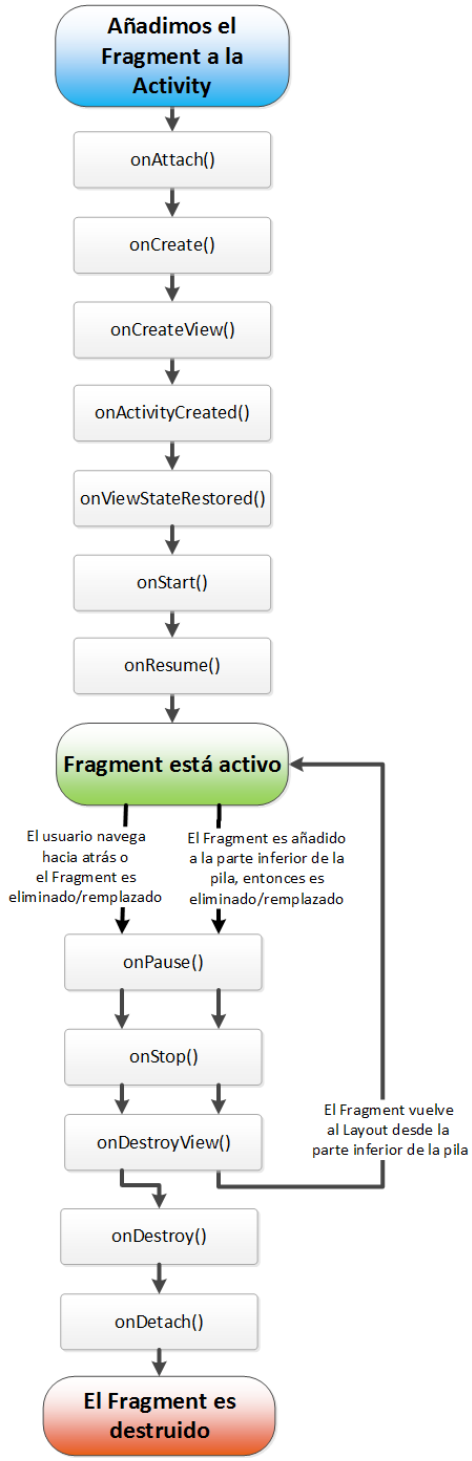


*Ilustración 21: Ciclo de vida de un Fragment*

Los *fragments* tienen la peculiaridad de que pueden ser estáticos, se define su contenido una vez y no varía, o dinámicos donde la estructura permanecerá constante pero el contenido se podrá actualizar.

Vista un poco la teoría sobre los *fragments*, ahora toca explicar su uso. Debido a sus características de que pueden ser dinámicos y se pueden reutilizar, vienen muy bien a la hora de diseñar las pantallas, ya que nos permiten por ejemplo cambiar el contenido de la pantalla de forma dinámica sin cambiar de actividad ni de *layout* o dividir la pantalla en varios *layouts* para mostrar información diferente en cada uno de ellos.

En las siguientes imágenes, muestro un ejemplo de cómo se podrían usar los *fragments* en una aplicación que tiene una lista y un contenido para elemento de la lista. Una vez programados los *fragments* y haciendo uso de las opciones de Android para comprobar el tamaño de las pantallas, se obtendrían diferentes formas de representación en función del tamaño de la pantalla, por ejemplo entre un teléfono móvil y una tableta.

- Teléfono móvil: Tendría dos pantallas una para la lista y otra para el contenido adicional.


*Ilustración 23: Ejemplo Fragment en teléfono móvil 1*


*Ilustración 22: Ejemplo Fragment en teléfono móvil 2*

- Tableta: La pantalla estaría divida en dos, una parte para la lista, que sería estática y otra parte para el contenido adicional, que sería dinámico.

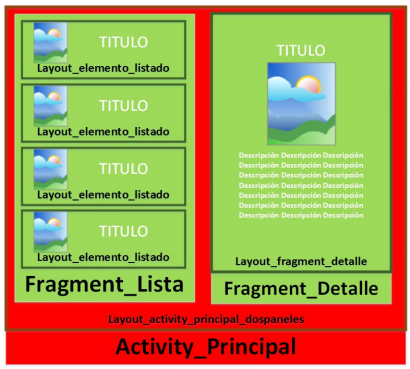


*Ilustración 24: Ejemplo Fragment en tableta*

Podría haber hecho uso de una configuración como la del ejemplo para la aplicación pero no lo consideré oportuno al estar destinada a un público infantil no quería que hubiera muchos elementos en pantalla y porque cuando se muestran los bits, tiene que estar solo el bit en pantalla para que se centre la atención sobre el bit y no en otros elementos.

En el caso concreto de nuestra aplicación he usado la ventaja de los fragments dinámicos al mostrar la galería de bits, permitiéndome cambiar el texto y la imagen del bit sin necesidad de cambiar de *layout* o de actividad.

### *4.3.9.- AsyncTask*

*AsyncTask* es una clase de Android que nos permite el trabajo multihilo y por lo tanto multitarea en Android y además con la ventaja de es asíncrono, ya que nos permite crear un hilo de ejecución secundaria para realizar determinadas tareas, sin influir en el hilo principal, y enviar una notificación al hilo principal cuando ha realizado las tareas, para que el hilo principal utilice los resultados.

Esto es bueno porque permite descargar al hilo principal de tareas pesadas, como descargas desde Internet, tratamiento de imágenes, etc. que si las hiciera el hilo principal, parecería que se hubiera bloqueado ya que no permitiría hacer otras cosas mientras está la tarea pesada en ejecución. Esto provocaría que el usuario se molestara y seguramente desinstalará la aplicación y diera una valoración negativa.

Con el uso de *AsyncTask*, podemos dejar al hilo principal encargarse de solo de controlar la parte de la interfaz con el usuario y las interacciones de este y realizar el resto de tareas de forma paralela.

Con las siguientes imágenes podemos la diferencia entre el procesamiento en un solo hilo y en dos hilos, donde en el primer caso la aplicación se queda "bloqueada" mientras se ejecuta la tarea pesada y en el segundo caso no se aprecia tal "bloqueo".

- Un único hilo:


*Ilustración 25: Procesamiento en un hilo*

- Dos hilos:

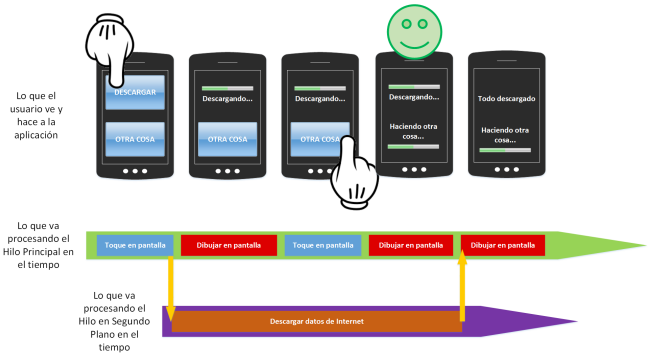


*Ilustración 26: Procesamiento en dos hilos*

Los principales métodos de AsyncTask para su uso son los siguientes:

- **onPreExecute()**: Se ejecuta antes del código principal de la tarea que vamos a ejecutar en segundo plano. Se suele utilizar para preparar la ejecución de la tarea y asignar valores iniciales a variables. Se ejecuta en hilo principal.
- **doInBackgrond()**: Contiene el código para ejecutar la tarea pesada. Se ejecuta después de *onPreExecute()*. Se ejecuta en el hilo en segundo plano y puede comunicarse con el hilo principal mediante *publishProgress()*.
    - **publishProgress()**: Método por el que enviamos información al hilo principal para notificar los avances realizados y notificar al usuario, por ejemplo. Lo recibe el método *onProgressUpdate()*.
- **onProgressUpdate()**: Se ejecuta cuando el método *doInBackgorund()* utiliza el método *publishProgress()* para enviar información. Se ejecuta en el hilo principal con la información que recibe.
- **onPostExecute()**: Se ejecuta cuando termina de forma satisfactoria la ejecución de *doInBackground()*. Se ejecuta en el hilo principal.
- **onCancelled()**: Se ejecuta cuando *doInBackground()* no ha terminado de forma correcta por que se llamó al método *cancel()*. Se ejecuta en lugar de *onPostExecute()*, si se ejecuta uno no se ejecuta el otro. Se ejecuta en el hilo principal.

Un ejemplo gráfico del ciclo de vida de *AsyncTask* y la utilización de estas funciones es el siguiente:

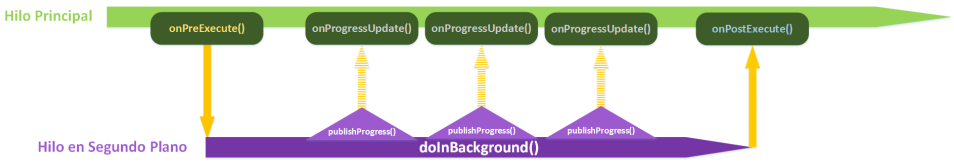


*Ilustración 27: Ciclo de vida de AsyncTask*

En la aplicación he usado *AsyncTask* para la ejecución multihilo en la galería de bits, para controlar el cambio de información en el *fragment* dinámico y controlar la reproducción del sonido de cada bit.

### *4.3.10.- La historia de los dulces nombres de las versiones*

Desde la salida de Android 1.5 *Cupcake*, Google ha seguido la tradición de ponerle un nombre de postre a cada una de sus nuevas versiones de Android siguiendo un orden alfabético: después de ***C**upcake* llegarían ***D**onut*, ***E**clair*, ***F**royo*, ***G**ingerbread*, ***H**oneycomb*, ***I**ce Cream Sandwich*, ***J**elly Beam*, ***K**itKat*, ***L**ollipop*, ***M**arshmallow* y la más reciente ***N**ougat*.

Las primeras dos versiones de Android (1.1 y 1.2) vinieron sin nombre y se les llama simplemente *Alpha* y *Beta*, aunque al representar las dos primeras letras del alfabeto, y para seguir con la tradición, también se conocen comúnmente como ***A**pple Pie* y ***B**anana Bread*.

¿Qué dice Google al respecto?

Google se ha mostrado siempre muy reservado cada vez que se le ha preguntado por el tema. En una ocasión, el portavoz de Google, Randall Sarafa, respondió a esta cuestión explicando que se trataba de una broma interna que comenzó con Android 1.5 *Cupcake*, y que ni siquiera él estaba muy seguro de dónde comenzó todo.

En la presentación de Android 4.4 *KitKat*, Google también explicó que la razón por la que todas las versiones de Android traían nombres de dulces era porque los *smartphones* y *tablets* endulzan nuestra vida y, entre estos, el chocolate es particularmente adictivo, siendo *KitKat* el favorito de la empresa. Una respuesta poco convincente, pero en fin...

¿Cómo conclusión?

Como con la mayoría de casos similares, seguramente la razón es mucho más sencilla y menos misteriosa de lo que pueda parecer y, tal y como explica Google, lo que comenzó como una broma entre trabajadores, se convirtió en el sello de identidad de Android. Una representación que es llamativa y siempre genera expectación entre los seguidores de este sistema operativo. Tanto es así que cada versión de Android cuenta con su propia estatua en Mountain View, California, donde se encuentran las oficinas centrales de la empresa.

Google, además, no es la única compañía que utiliza un sistema de nombres de este tipo. Apple pasó años poniéndole a su software OS X nombres de felinos (*Puma, Leopard, Lion*...) y nunca reveló la razón por la que lo hacía. Sin embargo, Apple dejó de poner estos nombres a su sistema operativo, en parte, suponemos, porque se quedaron sin nombres de felinos que utilizar.

## *4.4.- Aplicaciones Similares*

Para ver si hay alguna aplicación similar a la que quiero desarrollar, accedo a la tienda *Play Store* de Google y realizo una búsqueda con el término “bits de inteligencia”, obteniendo entre otros los siguientes resultados:

- **Bits de Inteligencia – FULL**: Aplicación de *Koy Group* con un coste de 1,20 €. No instalo ni pruebo esta aplicación.
- **Bits de Inteligencia – LITE**: Aplicación de *Koy Group* gratuita. Instalo dicha aplicación para ver su contenido y funcionamiento. Es una versión de prueba de la versión FULL en la que hay accesibles dos galerías de imágenes pero se muestran los títulos delas otras galerías disponibles. La aplicación está en español, las imágenes de los bits son dibujos y dentro de las categorías de bits, las pantallas con los bits pasan de forma automática. Permite la opción de mostrar el texto del bit en mayúsculas, minúsculas o no mostrarlo y dispone de un contador para cada categoría.
- **Bits de inteligencia**: Aplicación de *Educa Kids*. Instalo dicha aplicación para ver su contenido y funcionamiento. La aplicación es gratuita pero contiene publicidad, en forma de una barra inferior y mostrando pantallas de publicidad cada pocas pantallas. Permite la selección de varios idiomas para los bits (alemán, inglés, francés, portugués, español e italiano), dispone de varias categorías de bits y dentro de cada categoría hay que pasar los bits de forma manual.
- **Bits de Inteligencia n Inglés**: Aplicación de *Editorial Septiembr*e. No instalo ni pruebo esta aplicación.
- **BITS de mates, CANTIDADES**: Aplicación de *Educamigos & Smartbrain* con un coste de 9,99 €. No instalo ni pruebo esta aplicación.

Como resultado de esta búsqueda y la instalación y prueba de alguna de las aplicaciones encontradas, obtengo los resultados de que hay varias aplicaciones similares, unas con coste y otras gratuitas con publicidad, siendo las gratuitas las que más descargas tienen (entre 500 y 1000 descargas), ya que en las de pago no indica el número de descargas.

Debido a este análisis puedo destacar que parece que hay cierta competencia pero lo que más me llama la atención es el bajo número de descargas que tienen las aplicaciones, lo que podría indicar el motivo por el que no hay mayor número de aplicaciones con esta temática.

## *4.5.- Buenas prácticas en Android*

Inicialmente no había muchas normas y códigos de buenas prácticas para Android más allá de gustos y opciones personales de los desarrolladores y las buenas prácticas de cualquier otro lenguaje o plataforma de desarrollo como por ejemplo pensar antes de programar (qué vas a necesitar, cómo lo vas a hacer, qué problemas te puedes encontrar, qué código puedes reutilizar...), tener el código bien organizado y agrupado para ello hay que encapsularlo, modularizarlo, crear librerías, etc. lo que te permite un gran control y reutilización, es decir, la pauta de “divide y vencerás”, utilizar una nomenclatura estandarizada para nombrar las clases, archivos, funciones, etc. hacer un uso correcto de las característica *public*, *protected* y *private*. Todas estas recomendaciones siempre hay que tenerlas en consideración

Posteriormente y según se ha hecho más popular y han ido avanzando las versiones, la situación a cambiado y ahora Google, siempre teniendo en cuenta las limitaciones de recursos de los dispositivos, propone ciertas pautas y proporciona más recursos para que la calidad de los

desarrollos sean mejores, además recomienda utilizar ciertas herramientas que aunque no es obligatorio, es altamente recomendable para mejorar la aplicación y la experiencia del usuario como por ejemplo:

- **Utilizar Fragments**: De esta forma podremos disponer de aplicaciones universales no solo para teléfonos inteligentes sino para todo tipo de dispositivos con diferentes tipos de pantallas.
- **Utilizar *<include>* y *<merge>***: Además de reutilizar código, permite reutilizar partes de la interfaz que se repitan y evitar vistas enlazadas:
- **Carpetas adecuadas**: Android proporciona una estructura de carpetas y archivos que nos permite tener cada elemento en su lugar en función de su utilidad, desde las carpetas para guardar las imágenes en función de su tamaño y resolución, archivos de *STRING* para diferentes idiomas...
- ***Layouts* y *dimens***: A la hora de trabajar con distintas resoluciones y tamaños de pantallas, hacer un uso correcto de los *layouts* si los diseños de las pantallas son diferentes o de *dimens* si los diseños de las pantallas son iguales.
- ***Static final***: Cuando declaremos constantes, estas deben ser *static final*. Hay que tener en cuenta que los nombres de extras en un *intent* y los nombres de *ContentValues* son constantes.
- **Conocer el ciclo de vida**: Si conocemos bien el funcionamiento de los ciclos de vida de cada elemento podemos realizar mejores desarrollados, por ejemplo, usar *onPause()* para acciones que consuman GPU o batería y *onStop()* para guardar datos que no queremos que se pierdan.
- ***Context***: Hay que tener cuidado al hacer referencias al *Context* y saber que hay varios *Context* diferentes.
- **Referencias no necesarias**: Hay que tener cuidado de no cargar librerías no necesarias o tener variables que no se usan.
- **Variables estáticas o de clase**: Hay que evitar su uso siempre que no sea imprescindible, ya que este tipo de variables no serán eliminadas por el recolector de basura y estarán más tiempo en memoria que las variables instanciadas.
- **Concatenación de cadenas**: Evitar las concatenaciones de *Strings* ya que cada vez que se concatena se produce un nuevo objeto que consume memoria y recursos, en vez de ello, debemos utilizar *StringBuilder.*
- **Métodos Estáticos**: Siempre que sea posible, debemos utilizar los métodos estáticos ya que son más rápidos y evitan tener que instanciar objetos para ser invocados.
- ***Getters***: Hay que evitar usar *getters* innecesariamente, por ello, dentro de una clase hay que acceder directamente a la variable en vez de usar su *getter.*
- **Enlazar Vistas**: Hay que evitar enlazar muchas vistas, ya que estás se van guardando en la pila y cuantas más vistas haya más recursos utilizamos y más tardarán en cargarse las nuevas vistas.
- ***CompoundDrawable***: Se recomienda usar un *TextView* con un *CompoundDrawable* en vez de usar un *ImageView* junto con un *TextView.*

- ***HTTPURLConnection***: Se recomienda usar *HTTPURLConnection* en lugar de *HTTPClient* ya que la API es más sencilla y tiene mejoras en cuanto a compresión y cacheo por lo que aumenta la velocidad y el ahorro de energía.
- **Almacenamiento**: A la hora de almacenar los datos de las aplicaciones hay que tener en cuenta el tipo de dato que es y donde será más oportuno guardarlo y usar siempre los métodos proporcionados por Android para su uso, hay tres posibilidades:
  - **SharedPreferences**: El resultado es un archivo XML con las preferencias de la aplicación.
  - **Internal Storage**: Se almacena en el propio dispositivo y es solo accesible por la aplicación. Cuando se borra la aplicación se borran estos datos.
  - **External Storage**: Se almacena en la memoria externa del dispositivo, generalmente una tarjeta de memoria. Tiene una visibilidad pública para todos. No se borra cuando se desinstala la aplicación. Hay que comprobar el estado antes de usarlo.
- **Tareas Asíncronas**: El uso de *AsyncTask* para la ejecución de tareas de forma asíncrona y en segundo plano es altamente recomendado para evitar demoras en la ejecución de la aplicación y posibles molestos mensajes de *loading* o de que la aplicación no responde. Esto es debido a que como todos los componentes de una aplicación en Android se ejecutan en el mismo hilo (incluida la generación de la interfaz gráfica), cualquier operación de gran duración o tamaño puede bloquear la ejecución del resto de componentes, pudiendo provocar los mensajes de *loading*, error de la aplicación o una pantalla negra al demorarse la creación dela interfaz gráfica. En caso de llegar a ese caso, el usuario percibe que la aplicación no es fluida, es lenta, está bloqueada, funciona mal o cualquier otro pensamiento que se le pase por la cabeza, que no será bueno para la aplicación y puede provocar que la desinstale. Para evitar esto, se usa el procesamiento asíncrono, que al separar los procesos, proporciona mayor fluidez al realizar las tareas más costosas en segundo plano, sin influir en el proceso principal. Como vemos, esta es una característica muy importante ya que influye mucho en el rendimiento y la sensación que va a percibir el usuario, por lo que la vamos a estudiar más detalladamente.

La clase *AsyncTask* está compuesta por varios métodos, de los cuales los más importantes son:

- ***onPreExecute()***: Se ejecutará antes del código principal de nuestra tarea. Se suele utilizar para preparar la ejecución de la tarea de gran tamaño o que puede causar problemas.
- ***doInBackground()***: Contendrá el código o la llamada al método que contiene el código que puede causar problemas.
- ***onProgressUpdate()***: Se ejecutará cada vez que llamemos al método.
- ***publishProgress()***: se ejecutará al ser llamado desde el método *doInBackground()*.
- ***onPostExecute()***: Se ejecutará cuando finalice nuestra tarea conflictiva, o bien, tras la finalización del método *doInBackground()*.
- ***onCancelled()***: Se ejecutará cuando se cancele la ejecución de las tareas antes de su finalización normal por determinadas causas.

Al usar está clase para las tareas asíncronas, podemos evitar los problemas anteriormente mencionados y usarlo para tareas como cargar datos de la base de datos, recorrer bucles, etc. El

uso de esta clase facilita la tarea de trabajar con varios hilos y en segundo plano, que es prácticamente imprescindible si no quieres que Android te muestre el famoso mensaje de "*Application Not Responding*" (ANR) en el que se pide al usuario que decida entre forzar el cierre de la aplicación o esperar a que termine. Esto sucede ya que Android monitoriza las operaciones realizadas en el hilo principal y detecta aquellas que superen los 5 segundos, barrera que en determinadas ocasiones es fácil superar.

Además de todo lo anterior, para hacer nuestras aplicaciones más exitosas hay que hacerlas visualmente asombrosas, que simplifiquen la vida al usuario y que sean fáciles de usar y muy intuitivas. Para conseguir estos objetivos, es conveniente tener en cuenta los siguientes consejos:

- Usar patrones de diseño para que las pantallas de visualización sean lo más similares posibles.
- Buen control de los gestos a los que el usuario está acostumbrado (pulsar, mantener pulsado, arrastrar, deslizar, doble pulsación, pellizcar)
- Limitar el uso de los cuadros de dialogo, fundamentalmente los que son evitables como las pantallas de carga, avisos de error, promociones y publicidad y limitarse casi exclusivamente las notificaciones con consecuencias importantes, por ejemplo cuando se va a borrar un elemento y no hay vuelta atrás.
- Fluidez y estabilidad en el uso de la aplicación.
- Estar al día y actualizada.
- Uso de librerías: Además delas librerías de Android, hay más librerías disponibles como *androidviews.net* o *theultimateandroidlibrary.com*.

Juntando todo esto veremos cómo nuestras aplicaciones son interesantes para el usuario ya que serán fluidas, llamativas estéticamente e intuitivas y para nosotros como desarrolladores serán fáciles de controlar, mantener y actualizar.

## *4.6.- Google Play Services*

*Google Play Services* es una aplicación de la plataforma Android de uso interno, que nos permitirá tener el resto de aplicaciones de nuestro terminal siempre actualizadas, ya que se encargará de comprobar que todas las aplicaciones instaladas están en la última versión disponible. También se encarga de conectar las aplicaciones de terceros con las APIs de Google.

Las funciones principales de *Google Play Services* son la autentificación de servicios de Google, la sincronización de contactos, el acceso a la última configuración de privacidad del usuario, y los servicios basados en la ubicación de mayor calidad y menor potencia.

Además de todo esto, *Google Play Services* mejora la experiencia general de uso del terminal. Permite agilizar las búsquedas sin conexión, proporciona mapas más envolventes y mejora la experiencia de juego mediante la optimización de memoria RAM.

Otras opciones dentro del menú de *Google Play Services* nos permitirán gestionar las aplicaciones de nuestra cuenta (no necesariamente las que tengamos instaladas), o establecer diferentes ajustes de localización.

*Google Play Services* es una aplicación realmente imprescindible mientras estemos utilizando un terminal con sistema operativo Android, ya que sin ella instalada es posible que muchas otras aplicaciones comiencen a sufrir problemas.

Aunque no todo es bonito, *Google Play Services* a la vez que se encarga de los servicios de Google y mantenerlo todo al día, actualizado y optimizado, hace un uso de los recursos del teléfono y la batería y es una manera que tiene Google de controlar los dispositivos, ya que sin esta aplicación no funcionarían sus APIs que muchas otras aplicaciones de terceros utilizan para acceder a sus servicios como mapas, calendario, correo, etc.

# 5.- Modelo de negocio

## *5.1.- Definición*

Un **modelo de negocio**, es la planificación que realiza una empresa o emprendedor respecto a los ingresos y beneficios que intenta obtener. En un modelo de negocio, se establecen las pautas a seguir para atraer clientes, definir ofertas de producto e implementar estrategias publicitarias, entre muchas otras cuestiones vinculadas a la configuración de los recursos de la compañía.

A la hora de establecer el modelo de negocio es importante que la persona en cuestión analice en profundidad la empresa y dé respuesta a una serie de preguntas pues en base a las respuestas podrá poner en marcha uno u otro tipo de modelo de negocio. En este caso, es importante que establezca si tiene competencia o no en ese servicio o producto que posee, qué es lo que le hace diferente del resto de rivales empresariales, cómo va conseguir clientes, cómo se producirá el crecimiento y cómo se va a ganar el dinero.

## *5.2.- Modelo de negocio para una aplicación móvil*

Desarrollar un modelo de negocio de una aplicación móvil que tenga éxito requiere mucho trabajo, esfuerzo, y un delicado proceso de ensayo y error para entender cuáles son las necesidades del mercado.

Independientemente de la aplicación que se vaya a desarrollar y lanzar o el mercado de destino, hay cuatro funciones esenciales que se deben tener en cuenta dentro del modelo de negocio, todas ellas indispensables para que la aplicación sea un éxito:

1. **Usabilidad**:

Es un elemento clave del modelo de negocio de una aplicación móvil. Cuando un usuario abre una aplicación por primera vez le dedica unos segundos de atención para determinar qué es, cómo funciona o cómo se juega. En esos primeros instantes te estás jugando todo el esfuerzo de su desarrollo. Los usuarios no suelen tener mucha paciencia, por lo que si la aplicación no responde de manera intuitiva, habrás echado a perder todo el trabajo.

En los negocios, como en casi todos los aspectos de la vida, sólo tienes una oportunidad para crear una buena primera impresión y hay que aprovecharla.

2. **Distribución**:

La distribución es el trabajo necesario que tienes que llevar a cabo para llegar a que los usuarios descubran e instalen tu aplicación. Es el momento de promocionar tu aplicación y de crear un plan de marketing. Hay que tener en cuenta los siguientes factores:

- Viralidad:

Es la capacidad de convencer a los usuarios para que recomienden tu aplicación a sus amigos y conocidos. Se trata de la publicidad más eficaz, pero la más difícil de lograr.

El primer paso es eliminar todos los obstáculos y frenos que el usuario pueda tener para recomendar la aplicación. Busca la manera más fácil y que más sentido tenga para tu tipo de aplicación. Facilita al usuario la obtención de sus listas de amigos, conecta la aplicación con redes sociales como *Facebook, Twitter* o *Instagram*, permite el envío de correos electrónicos, sms…etc. Con esto consigues educir la fricción, que es necesario pero no suficiente. Lo más difícil es generar un estímulo e incentivar al usuario para que recomiende la aplicación. Un buen truco es apelar al ego o al sentido de la competencia, por ejemplo mediante puntuaciones que se puedan ver y compartir, premios y logros, etc.

Aunque logres una alta tasa de virilidad, en la que cada nuevo usuario recomienda a otro que decide descargar la aplicación, no tendrás todo el problema resuelto. Ya que si tienes pocos usuarios, la tasa de crecimiento será baja, por lo que inicialmente necesitas dar a conocer la aplicación de alguna manera para lograr un primer número de usuarios suficientes para que la viralización tenga éxito.

- Publicidad:

La publicidad depende principalmente de la inversión que decidas hacer. A través de la compra de espacios publicitarios en medios digitales puedes asegurarte la primera generación de usuarios.

La eficiencia de la publicidad se mide por el coste de adquisición de un cliente (CPA). Esto se calcula dividiendo el precio de cada campaña de publicidad por el número de usuarios que ha conseguido esa acción. Cuanto menor sea la CPA, más eficiente habrá sido la campaña, es decir mayor número de usuario habrás conseguido con la campaña.

- Medios de comunicación - Prensa:

Es el tercer método de distribución de una aplicación. Los medios de comunicación son los encargados en este caso de determinar el atractivo que puede tener la aplicación para el público. Lo único que puedes hacer es facilitarle información interesante y llamativa para que sea "*noticiable*" a los periodistas para que hablen de ti.

Este método es de bajo coste, pero no supone un control total sobre las acciones que se van a llevar a cabo y tampoco hay garantía de éxito.

3. **Monetización**:

El éxito económico de una aplicación y su capacidad para reinvertirlo en acciones de campañas publicitarias se mide por su grado de monetización y por el ARPU (ingreso medio por usuario) que genera.

Estas son las formas más comunes de monetización:

- *Ingresos por publicidad*:

Es el método más común de monetización, en la que el usuario está expuesto a ver publicidad dentro de la aplicación. Cuando el usuario está interesado en un producto se va de la aplicación hacia el otro servicio. Si no está interesado, puedes llegar a molestarlo. Por lo general, la publicidad en las aplicaciones es molesta y atenta contra la paciencia del usuario, de manera que es una buena práctica ofrecer una versión *Premium* que esté libre de publicidad.

Normalmente, del dinero que paga el anunciante, el 60% va para la aplicación, pero si hay muchos intermediarios este valor podría reducirse.

- Ingresos por descargas o versiones *Premium*:

Los ingresos por descarga se generan cuando el usuario tiene que pagar una sola vez para descargarse la aplicación tantas veces como desee. Este método puede ser una estrategia adicional a los ingresos de publicidad o podría ser tu única estrategia. De esos ingresos, hay un porcentaje que se lo quedan los distribuidores, tiendas o *Stores*.

- Compras dentro de la aplicación:

El método de compra dentro de la aplicación se genera cuando el usuario, después de haber probado la aplicación, se da cuenta de que podría mejorar significativamente su experiencia con una inversión mínima de dinero, una cifra que por lo general es casi insignificante, estas mejoras pueden ser por ejemplo para obtener más recursos en los juegos, más vidas, que se reduzcan tiempos de espera, etc.

Estas versiones se han popularizado mucho ya que suponen un gran negocio y fuente de ingresos al tener que hacer una inversión mínima. Se han denominado *Freemium*, ya que los servicios básicos o la descarga de la aplicación es gratis (*Free*) pero después hay servicios avanzados o mejoras por las que hay que pagar (*Premium*). Al igual que con las descargas, hay un porcentaje que se lo quedan las distribuidores, tiendas o *Stores*.

- Suscripción:

Es el método que a menudo utilizan las aplicaciones de generación de contenidos en las que el usuario paga una cuota mensual con la que tiene acceso a todos los contenidos nuevos que se vayan generando e introduciendo en la aplicación. Por lo general se cobra fuera de las distribuidores, tiendas o *Stores* y no sufre sus retenciones ni porcentajes de los pagos.

4. **Compromiso del usuario**:

El último factor que debes tener en cuenta es el *engagement* o el compromiso de los usuarios con la aplicación. Debes medir el número de sesiones de los usuarios y la duración de las mismas.

Todas las acciones y mejoras que realizan los desarrolladores tienen el fin de estimular al usuario para que pase más tiempo utilizando la aplicación, ya que esto genera mayor exposición a la publicidad y aumenta el ARPU y la viralidad. Un buen método para aumentar la frecuencia de uso entre una sesión y la siguiente, es enviar mensajes de alerta con promociones o novedades, que recuerden al usuario que puede volver a usar la aplicación.

Lograr una comunidad de usuarios fieles es la clave para que tu aplicación tenga éxito. Pero recuerda que para ello tendrás que trabajar bien todos los puntos anteriores.

## *5.3.- Modelo de negocio para la aplicación Bits de Inteligencia*

De acuerdo a lo expuesto anteriormente y su estructura, voy a explicar cómo sería el modelo de negocio elegido para la aplicación objeto de este Trabajo de Fin de Grado:

1. **Usabilidad**:

Este tema se ha tenido muy en cuenta debido al público al que va dirigido, que al ser un público mayoritariamente infantil, la aplicación debe de ser muy llamativa, extremadamente fácil de usar e intuitiva para que cualquiera pueda usarla sin tener ningún tipo de conocimiento previo, simplemente usando estímulos visuales.

2. **Distribución**:

Se ha pensado que la forma en la que se distribuirá la aplicación y se dará a conocer es mediante el boca a boca y la viralización. Para ello, se comentara la existencia de la aplicación a familiares, amigos y conocidos, facilitando el enlace de descarga de la aplicación cuando esté publicada en *Google Play Store*, como se tiene previsto, haciendo especial énfasis en las personas que tengan relación directa con niños, como familiares de niños (madres, padres, abuelos, etc.) o profesores y esperando que cuando descubran la aplicación y vean su gran utilidad, les cree el estímulo necesario para que la recomienden.

3. **Monetización**:

Inicialmente no se tiene prevista una monetización de la aplicación al tratarse del Trabajo de Fin de Grado y tener como objetivos el aprendizaje de la plataforma Android y el cómo hay que desarrollar una aplicación.

Para completar el análisis, voy a tener en cuenta este apartado, aunque la aplicación sería gratuita. Debido a la naturaleza de la aplicación se descarta la opción de la suscripción ya que no va a haber una generación de contenidos continua que avale esta opción. Debido al público al que va dirigido, no considero apropiado que la aplicación tenga publicidad, por lo que se descarta esa opción también. Respecto a las dos que quedan de que la aplicación se *Premium* o *Freemium*, ambas las veo posibles, explicare cada caso:

- *Premium*: Esta sería la opción que considero más propicia, ya que con un pago inicial para descargar la aplicación, estarían todas las galerías de Bits disponibles y la aplicación estaría plenamente operativa y funcional, permitiendo a los usuarios finales disfrutarla y sin que los usuario adultos tengan que preocuparse de que haya partes accesibles o no o que los usuarios infantiles puedan tener problemas con la aplicación por la publicidad o por que quieran acceder a alguna parte que requiera un pago previo. Se pondría un precio simbólico de 0,99 € para que el precio no alejase a potenciales clientes, pero al tener un precio monetario, los usuario lo valorarían más y como desarrolladores obtendríamos un *feedback* monetario para amortizar parte del tiempo y esfuerzo invertido.
- *Freemium*: Esta opción la considero porque Android introdujo el control parental para la opción de compras dentro de las aplicaciones, por lo que no sería posible que el usuario infantil realizara compras por error y sería un usuario adulto el que las realizase. Esta opción sería viable si se ofrece la aplicación de forma gratuita para la descarga con alguna galería de Bits accesible y que el resto de galerías de Bits estuvieran visibles para que pudieran ver de qué se trata pero cuando se va a acceder a ella solicite un pago. Es decir habría que pagar por cada galería de Bits adicional que se quisiera ver. No sería la opción principal de monetización que escogería por el público al que va destinada, ya que al ser niños no entenderían lo que pasa y por qué no pueden acceder a todo desde el principio.

4. **Compromiso del usuario**:

Considero que el compromiso del usuario sería alto ya que para los usuarios adultos (padres, profesores, etc.) la aplicación resultaría de gran interés al facilitar el aprendizaje de los usuario infantiles, y para los usuarios infantiles resultaría entretenida y divertida ya que podrían imitar a los adultos usando un dispositivo electrónico y además la aplicación será llamativa y se pretende lograr captar su atención.

# 6.- Plan de Negocio

## *6.1.- Definición*

El **plan de negocios** es un documento que describe, de manera general, un negocio y el conjunto de estrategias que se implementarán para su éxito. En este sentido, el plan de negocios presenta un análisis del mercado y establece el plan de acción que seguirá para alcanzar el conjunto de objetivos que se ha propuesto.

Como tal, el plan de negocios tiene un uso interno, desde el punto de vista de gestión y planificación, y otro externo, como herramienta de promoción y comunicación de la idea del negocio, bien sea para venderla, bien para obtener financiamiento.

El plan de negocios, en este sentido, sirve de brújula para el emprendedor, pues le permite tener un mejor entendimiento del negocio, al mismo tiempo que lo obliga a investigar, reflexionar y visualizar todos los factores, tanto internos como externos, que incidirán en la marcha de su negocio. Del mismo modo, los planes de negocio son documentos que se encuentran sujetos a constantes actualizaciones y replanteamientos, conforme con las dinámicas propias de la gestión empresarial.

En todos los planes de negocio, es importante que el emprendedor incluya información veraz. Las predicciones que realice sobre ingresos del negocio deben ser conservadoras, de modo que la sostenibilidad del negocio esté prevista en el plan de negocios sin grandes números. Siempre es preferible que las ventas superen las previsiones y no viceversa.

Estructura de un plan de negocio

Un plan de negocio debe constar de los siguientes puntos:

1. Resumen Ejecutivo:

El resumen ejecutivo es un resumen de las otras partes que conforman el plan de negocios, por lo que a pesar de ir al inicio del plan, debe ser desarrollado después de haber culminado las demás partes. Con una sola lectura debe enseñarnos en que consiste el plan de negocio. Debe incluir los siguientes elementos:

- Los datos básicos del negocio como nombre, ubicación, tipo de empresa, etc.
- Una descripción del negocio: una breve descripción que explique el negocio y los productos/servicios que se van a ofrecer.

- Características diferenciadoras: Las características innovadoras con las que va a contar el negocio, producto y servicio que van a servir para distinguirse de la competencia.
- La idea de la necesidad u oportunidad de negocio: las razones que justifican el negocio.
- Los objetivos del negocio: los objetivos que se quieren alcanzar un vez que el negocio esté en marcha.
- Las estrategias del negocio: Las estrategias que se llevaran a cabo para conseguir los objetivos.
- El equipo: las personas que se encargarán de poner en marcha y gestionar el negocio.
- Inversión requerida: Es la inversión necesaria para poner en marcha el negocio y hacerlo funcionar durante el primer ciclo productivo.
- Rentabilidad esperada: son los resultados de los indicadores de rentabilidad utilizados.
- Impacto ambiental: es un resumen del impacto ambiental que tendrá el negocio.
- Conclusiones del proyecto: son las conclusiones que se obtienen una vez realizado el plan de negocio.

2. Definición del negocio, descripción del producto y valor distintivo:

En esta sección se define el negocio que se va a realizar y los aspectos relacionados con dicho negocio, tales como las razones que justifican el negocio, los objetivos, las estrategias, etc.

3. Estudio de mercado:

En este apartado se desarrolla el análisis del mercado objetivo, pronósticos de demanda y otros elementos relacionados.

3.1. Competencia:

Análisis y descripción de los competidores de nuestro negocio.

4. Estudio Técnico:

En este apartado se describen los requerimientos físicos necesarios para el funcionamiento del negocio, su proceso productivo y la infraestructura, tamaño y demás características del local en donde funcionará.

5. Organización del negocio:

En la organización del negocio se describe la estructura jurídica y orgánica del negocio, las áreas o departamentos que lo conformarán, y otros elementos relacionados con estas. Aquí se incluye la estructura jurídica, cargos y funciones, personal, etc.

6. Estudio de la financiero:

Es un análisis financiero del negocio, tanto para ponerlo en marcha como una vez que está funcionando para recuperar la inversión y tener beneficios viendo la rentabilidad del negocio.

6.1. Estudio de la inversión:

En este estudio se señala la inversión que se va a requerir para poner en marcha el negocio y hacerlo funcionar durante el primer ciclo productivo, y la financiación que se va a utilizar o necesitar.

6.2. Estudio delos ingresos y gastos:

En este apartado se estudian los ingresos y gastos que va a tener el negocio para el periodo de tiempo en que está proyectado el plan de negocios.

7. Estrategia de Marketing y ventas:

En este apartado se debe especificar cuál va a ser la estrategia a seguir para captar el volumen de usuarios deseados y cuál va a ser su coste de adquisición.

8. Principales Riesgos y Estrategias de Salida:

Aquí se realiza una evaluación de los riesgos que pueden afectar al negocio, es necesario incluir medidas concretas para hacer frente a dichos riesgos y una valoración alternativa de la compañía si se variasen algunos de los parámetros clave del modelo; como por ejemplo, tasa de crecimiento de usuarios, etc. Se pueden plantear planes de contingencia con posibles estrategias en caso de que no se logren los objetivos deseados.

## *6.2.- Plan de Negocio aplicado al Trabajo de Fin de Grado:*

En este apartado voy a aplicar los puntos anteriormente descritos a este Trabajo de Fin de Grado para obtener un Plan de Negocio personalizado:

1 . Resumen Ejecutivo:

El negocio se basa en la distribución de una aplicación móvil de carácter educativo para un público infantil que estará desarrollada bajo la plataforma Android.

La idea del negocio surge porque cada vez a más temprana edad los niños empiezan a usar los dispositivos móviles y se requiere que haya aplicaciones destinadas a ellos, que puedan utilizar de forma segura y que sirvan para su desarrollo. Hay varias aplicaciones de este tipo, por lo que hay una cierta competencia, nuestra idea para diferenciarnos es utilizar un método

didáctico muy intuitivo y útil, además de no incluir dentro de la propia aplicación compras o publicidad que puedan distraer al niño de uso.

El equipo inicialmente está compuesto por una única persona que realizara todas las labores y en función del crecimiento de la aplicación y las necesidades, la plantilla podría incrementarse.

La inversión necesaria es inicialmente la del desarrollo de la aplicación para obtener la aplicación deseada y una vez que se vaya a comercializar, es necesaria una nueva inversión para hacer frente a los gastos de marketing, impuesto de sociedades o autónomos (impuestos recurrentes en el tiempo) y registro de marca.

El objetivo principal de cubrir la inversión y obtener beneficios es difícil de cumplir, porque aunque el mercado es maduro con un altísimo porcentaje de utilización de dispositivos móviles inteligentes por parte de la sociedad, el mercado es muy dinámico y cambiante por lo que resulta complicado saber que necesidades van a tener los usuarios, a estos datos del mercado hay que añadir que hacen falta un gran número de descargas para cubrir la inversión inicial y que es difícil que una aplicación nueva y poco conocida alcanza tal número de descargas. En este caso los estudios y estadísticas están en nuestra contra.

2 . Definición del negocio, descripción del producto y valor distintivo:

Bits de Inteligencia para Android es una aplicación destinada a dispositivos Android, cuyo objetivo es colocarse como una de las aplicaciones más valoradas y descargadas de su sector, juegos educativos para niños, haciendo que los usuarios aprendan y se diviertan.

La aplicación consiste en elegir un idioma y posteriormente navegando por un menú visual de categorías se elige una, si la categoría consta de subcategorías, se muestra un nuevo menú donde se selecciona la subcategoría deseada, finalmente, se muestra una galería de imágenes y se reproduce un sonido por cada imagen.

**Misión**: La aplicación tiene como misión proporcionar un rato agradable y entretenido al usuario a la vez que se facilita su aprendizaje.

**Visión**: La visión que se pretende lograr es ser una aplicación reconocida y de prestigio, consiguiendo gran número de descargas y valoraciones positivas en la *Google Play Store*.

**Valor distintivo**: Ofrecer una aplicación útil, amigable, sencilla, de calidad, intuitiva y de alto valor didáctico, en la que el usuario se divierta al usarla y quiera volver a usarla, sin molestias de publicidad.

Las principales claves en las que vamos a basar el existo de la aplicación son:

- Desarrollo de calidad, sin errores y muy optimizado para que el tamaño de la aplicación sea lo menor posible, cosa que está estrechamente relacionada con la cantidad y calidad de los materiales didácticos que son las imágenes y los sonidos. Con la optimización se persigue que el uso de recursos y de batería sea lo menor posible.
- Sencillez y usabilidad: Queremos que la aplicación sea simple y fácil de usar de forma intuitiva, sin necesidad de leer un manual o unas instrucciones.
- Descripción clara de la aplicación en *Google Play Store* para que el potencial cliente sepa perfectamente que es la aplicación antes de descargarla.

- Tener en cuenta y valorar las opiniones de los usuarios, tanto positivas como negativas, para saber qué es lo que necesita el usuario y adaptarnos a sus gustos y necesidades.

## 3 . Estudio de mercado:

Esta sección está dedicada al estudio de mercado en el que va a competir la aplicación. Este mercado está centrado en las aplicaciones educativas para los dispositivos móviles inteligentes, fundamentalmente teléfonos inteligentes, pero también pueden incluirse tabletas u otros elementos debido a las grandes y constantes evoluciones que se producen. Toda la información que obtengamos y analicemos en esta sección, nos facilitará la comprensión del mercado en el que nos vamos a adentrar, la competencia, el público objetivo y potenciales clientes, etc. Esto aporta un valor adicional a nuestro análisis. Para realizar un correcto estudio de mercado hay que hacer referencia a diversos puntos relacionados con el sector en el que competimos, los tipos de usuarios, los potenciales clientes, etc. Ahora paso a desarrollar los puntos correspondientes:

### 3.1 . Análisis del sector de los dispositivos móviles inteligentes:

El sector de la telefonía móvil ha cambiado radicalmente debido a la gran revolución que viene dándose desde haces unos años con los dispositivos inteligentes, tanto móviles como tabletas inicialmente hasta los últimos dispositivos *weareables* o de Realidad Virtual. Esto ha propiciado la forma en la que vemos y utilizamos los dispositivos móviles que han pasado de solo poder realizar y recibir llamadas a realizar infinidad de funciones desde las más típicas hasta las más variopintas que se nos puedan ocurrir, por ello se los ha denominado dispositivos inteligentes.

El crecimiento en la venta de teléfonos inteligentes ha experimentado un crecimiento exponencial. A principios de 2013 superaron en ventas a los teléfonos tradicionales, cuya tendencia está claramente a la baja. Actualmente los teléfonos inteligentes son el dispositivo electrónico más vendido.

Otro dato que nos muestra la importancia del mercado de dispositivos móviles es la comparación entre este mercado y el mercado tradicional de los ordenadores personales. Según el informe "*La Sociedad de la Información en España*" en su 16ª edición, nos indica que el teléfono móvil inteligente es el dispositivo principal con el que los españoles se conectan a Internet con un 88.3% de los usuarios. Y en lo referente a ventas, mientras que los ordenadores personales, tanto de sobremesa como portátiles tienen progresión negativa, vendiéndose cada vez menos, los teléfonos móviles inteligentes y las tabletas están en auge y cada se venden más. Se estima que en el 2017 la proporción de ventas ordenadores-tabletas sea de 1 a 6, es decir por cada ordenador que se venderá, se venderán 6 tabletas, relación que a la que ya se llegó y superó con los teléfonos móviles inteligentes.

El crecimiento en la venta de teléfonos móviles inteligentes y tabletas, ha venido unido a una progresión en las características, donde cada vez los dispositivos tienen mejores y más rápidos procesadores, memoria RAM y fundamentalmente las pantallas, que cada vez tienen mayor tamaño y resolución. Toda esta evolución sirve para que con uno de estos dispositivos podamos realizar multitud de tareas.

Para mostrar la evolución de ventas, podemos ver la siguiente tabla elaborada por IDC sobre la venta de dispositivos:

| Categoría de producto | Ventas 2013 | Cuota de mercado 2013 | Ventas 2017 | Cuota de mercado 2017 | Crecimiento 2013-2017 |
|---|---|---|---|---|---|
| **PC sobremesa** | 134,4 | 8,6% | 123,1 | 5,0% | -8,4% |
| **PC portátil** | 180,9 | 11,6% | 196,6 | 8,0% | 8,7% |
| **Tableta** | 227,3 | 14,6% | 406,8 | 16,5% | 78,9% |
| **Teléfono Inteligente** | 1013,2 | 65,1% | 1733,9 | 70,5% | 71,1% |
| **Total** | 1556,8 | 100% | 2460,4 | 100% | 58,1% |

*Tabla 11: Dispositivos inteligentes por categoría, ventas y cuota de mercado, entre 2013 y 2017 (ventas en millones de unidades)*

También podemos ver la siguiente gráfica:

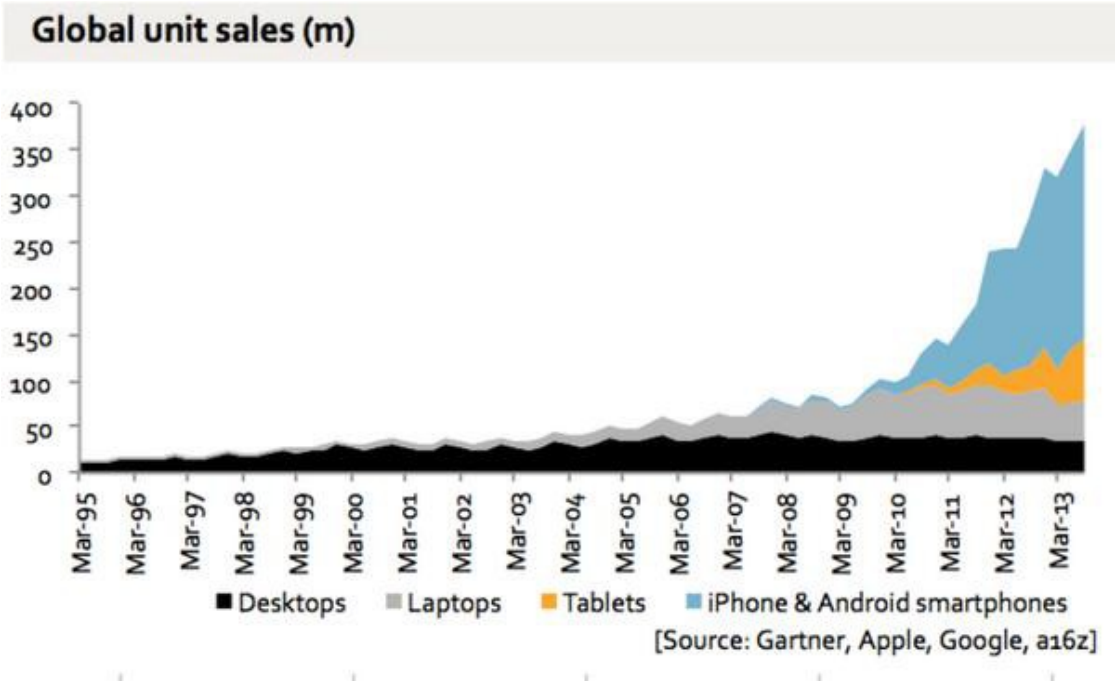


*Ilustración 28: Evolución de ventas de dispositivos*

3.1.1 . Análisis a nivel mundial:

Como hemos visto en el anterior punto, estamos en un continuo crecimiento de los dispositivos móviles, lo que marca un cambio en la mentalidad de la sociedad, donde ahora queremos poder comunicarnos con cualquier persona, cercana o lejana, estemos donde estemos. A esto se unen otras necesidades como buscar información, escuchar música, hacer fotografías, etc. estas nuevas necesidades marcan tendencias y es lo que ha organizado este gran cambio.

Para adaptarse a esta nueva tendencia y mentalidad, las compañías han tenido que adaptarse. Todo tipo de compañías, las fabricantes de dispositivos, las empresas de desarrollo, las

compañías de telecomunicaciones para ofrecer cada vez mejores servicios (velocidad de datos, tráfico mensual de datos, coberturas, tecnologías...), por nombrar algún tipo de compañías. La mayoría de compañías han sabido adaptarse bien a este nuevo mercado a cambio de hacer un gran esfuerzo en mejorar los dispositivos e innovarlos para saber entender y satisfacer las nuevas necesidades. Aunque no todas las compañías han sabido adaptarse, las que no lo han hecho han caído en picado hasta prácticamente desaparecer como *Nokia* o *RIM* con *Balckberry*, otras como *Microsoft* siguen luchando por adaptarse a este mercado y subir su cuota de mercado.

Cuando alguien quiere un nuevo dispositivo, quiere un aparato robusto, con gran conectividad, grandes capacidades multitudinaria, gran potencia, etc. para conseguir todo eso, además de la parte hardware de la que se encargan los fabricantes, se necesita un software apropiado y para esa tarea, la mayoría de fabricantes optan por utilizar Android frente a otros sistemas operativos, como ya vimos en la parte "**Estado del Arte**".

### 3.1.2 . Análisis a nivel nacional:

Ahora nos vamos a centrar en el mercado a nivel nacional. Según dice Google en su estudio "*Our Mobile Planet: Global Smartphone User*", España es el segundo país a nivel mundial en cuanto a penetración de teléfonos inteligentes en el mercado, solo superado por Reino Unido. En esta versión del estudio, España tiene un 44% de penetración de mercado, que supone un aumento de 11 puntos respecto al anterior estudio donde la penetración era del 33%. Este aumento es increíblemente grande ya que comparándolo con el aumento de Estados Unidos, una de las más grandes potencias mundiales, España lo supera en 3 puntos. Con esta progresión, España podría ponerse primera en el ranking mundial, ya que Reino Unido solo la supera por un punto, al tener una penetración de mercado del 45%.

Dentro de nuestro mercado nacional, si distinguimos entre los diferentes sistemas operativos móviles, atendiendo a las fuentes de *Kantar World Panel*, para la adquisición de nuevos dispositivos móviles, la mayoría de usuarios se decanta por Android con un 93.9% de los usuarios. Por lo que vemos que es una opción de futuro clara para nuestro desarrollo profesional.

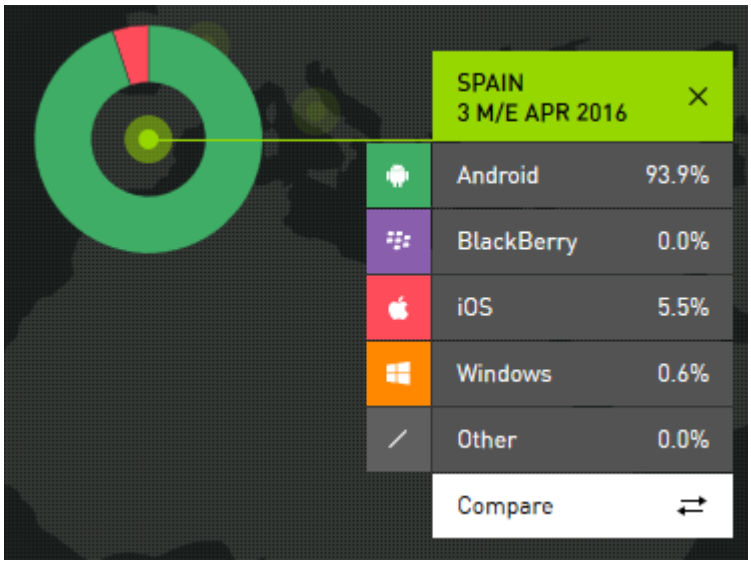


*Ilustración 29: Venta de dispositivos móviles en España en Abril de 2016. Fuente Kantar World Panel*

Haciendo referencia de nuevo al informe "*La Sociedad de la Información en España*" en su 16ª edición, en España ha crecido el uso de Internet en la franja de edad comprendida entre los 55 y 64 años, reduciéndose la brecha digital y sabemos que el 78.7% de la población entre 16 y 74 años se conecta regularmente a Internet para acceder a información, noticias, correo electrónico, servicios multimedia, etc. Aunque ese rango de edades y usos no es nuestro público objetivo, los datos del estudio nos sirven para afianzar nuestra decisión de haber optado por las plataformas móviles para este proyecto, ya que están en pleno auge y tienen gran futuro.

### 3.2 . Análisis del mundo de las aplicaciones móviles:

Con lo que hemos visto hasta ahora, sabemos que los dispositivos móviles han tenido un gran crecimiento y predeciblemente lo seguirán teniendo y probablemente cada vez tengan más funcionalidades. Para todas estas funcionalidades, además del sistema operativo, serán necesarias toda una serie de aplicaciones diseñadas y desarrolladas para cubrir esas necesidades y funciones. Ya que para realizar cualquier tarea, ya sea jugar, ver un archivo de texto, escuchar una canción, tomar una foto, etc. es necesaria una aplicación que lleve a cabo el trabajo.

Según diferentes estudios y fuentes como *GIGAOM*, *Vision Mobile*, *Ditrendia*, *IDC*, etc. se estima que las descargas de aplicaciones desde dispositivos móviles superan los 70 mil millones, donde el mayor mercado es Android con un 59%, seguido de Android con un 34% y el resto es para otras plataformas. Además se estima que el número de usuarios que usan aplicaciones móviles crecerá a una tasa aproximada del 30% anual. Actualmente el número de usuarios que utilizan aplicaciones móviles se estima que esta sobre los 3400 millones y que a finales de 2017 será de 4400 millones.

El nivel anual de descargas de aplicaciones oscila entre los 67.000 y los 92.000 millones de aplicaciones. De todas esas aplicaciones descargadas, los usuarios suelen centrar su uso en ocho o diez aplicaciones a las que dedican una media de dos horas y media diarias.

Dentro de las aplicaciones más usadas, destacan las aplicaciones de redes sociales y mensajería, siendo *Whatsapp* la reina de la lista, seguida por *Facebook* o *Instagram*, tanto en número de usuarios como de descargas.

Según los estudios, el uso de los dispositivos móviles y aplicaciones cada vez se usan más para realizar compras y transacciones, con incrementos de hasta el 35% en tareas tales como reserva de vuelos, hoteles, economías colaborativas como *Blablacar*, etc.

Gracias a todo este crecimiento de la industria móvil en los últimos tiempos y que seguirá teniendo, este mercado se presenta como uno de los más rentables y que tiene grandes oportunidades de negocio para los próximos años. También se muestra como un mercado muy atrayente para campañas de marketing, ya que es un mercado muy directo y con gran interacción con el consumidor final.

Ahora que vemos que es un gran mercado con mucho futuro y muchas oportunidades de negocio, hay que ver cómo poner las aplicaciones a disposición del usuario final. Para ello existen los "*markets*" o tiendas de aplicaciones. Estas tiendas están dominadas fundamentalmente por:

- Google Play Store: Servicio de descarga de aplicaciones de Google para la plataforma Android. Actualmente tiene más de un millón de aplicaciones.
- Apple Store: Servicio de descarga de aplicaciones de Apple para su plataforma IOS. Actualmente tiene más de un millón de aplicaciones.
- Windows Phone Market: Servicio de descarga de aplicaciones de Microsoft para su plataforma Windows Phone. Ha sido la última en llegar y la que menos aplicaciones tiene, alrededor de 300.000.

Hay que destacar que aunque estos *markets* son los principales, no son los únicos, ya que al darse cuenta del gran negocio que suponen, otras empresas han creado sus propios *markets*, como por ejemplo *Amazon* o *Samsung*.

Me gustaría hacer referencia a que los datos de descargas y de aplicaciones instaladas pueden dar lugar a engaños, ya que un usuario puede descargase una aplicación y tras el primer uso borrarla porque no es lo que esperaba o simplemente no volver a abrirla, como refleja el dato de que alrededor del 25% de aplicaciones que tienen los usuarios solo se han utilizado una vez.

Otro dato que es necesario conocer sobre este mercado, es que a pesar de que mueve gran cantidad de dinero y volumen de descargar, más del 90% de las aplicaciones son gratuitas a la hora de descargarlas. Aunque luego puedan generar reportes económicos mediante publicidad o descargas bajo pago, como se vio en el apartado "**Estado del Arte**".

### 3.3 . Análisis del entorno de desarrollo:

Este apartado está dedicado a explicar la plataforma de desarrollo Android, pero como ya se explicado en la sección de "**Estado del Arte**" y en los **anexos** dedicados a las versiones de Android y a como instalar *Android Studio* y crear la primera aplicación, remito al lector a dichos apartados y no me extiendo más en este apartado.

### 3.4 . Análisis de los usuarios de aplicaciones móviles:

Para determinar el mercado al que queremos dirigirnos, hay que estudiar el tipo de usuarios que lo componen, ya que no todos los usuarios utilizan los dispositivos móviles para los mismos fines. Para detallar ese estudio, hay que conocer a los usuarios, que dispositivos usan, para que finalidad los usan, cuanto tiempo dedican a su uso, etc. Podemos servirnos del estudio que realizó *G+J EMS Mobile*, mediante el cual se divide a los usuarios en cuatro tipos:

1. **Entusiastas**: Representan en torno al 18% de los usuarios de dispositivos móviles, oscilando sus edades entre los 14 y 39 años. Realizan un uso intensivo del teléfono inteligente orientado a redes sociales y las compras de aplicaciones. Han adoptado el teléfono inteligente como parte fundamental de su vida.
2. **Experimentados**: Suelen ser hombres entre 30 y 50 años que representan el 24% de los usuarios de teléfonos inteligentes. Centran la utilización de estos dispositivos en facilitar su vida, tanto personal como laboral con aplicaciones orientadas a ese fin.
3. **Jóvenes de rutina**: Es la parte de los usuarios comprendida entre los 20 y los 29 años que realizan un uso intensivo del terminal. Sus principales usos son los videojuegos, escuchar música y comunicarse por redes sociales y mensajería instantánea. Representan el 28% de los usuarios.
4. **Selectivos**: Representa el 30% de usuarios restante. Son personas de más de 40 años con altos ingresos que están descubriendo las posibilidades de estos dispositivos. Buscar simplificar su vida y poder comunicarse con el entorno que los rodea.

### 3.5 . Análisis de los clientes:

Ahora que ya conocemos como son los usuarios de las aplicaciones móviles, ahora tenemos que definir cuáles de esos usuarios serían nuestros potenciales clientes.

Teniendo en cuenta las característica de nuestra aplicación y que va dirigida a un público infantil que no tiene un dispositivo en propiedad, nuestro público objetivo se centra en el entorno de los niños, fundamentalmente sus familia, profesores y cuidadores. Debido a esto, cualquiera de los cuatro grupos de usuarios nos podría servir para hacer llegar la aplicación a los niños. Los jóvenes de rutina y los entusiastas, por los rangos de edad podrían ser los padres de los niños o familiares jóvenes como primos por ejemplo, aunque los padres podrían formar parte de los entusiastas. Los padres también podrían formar parte del grupo de experimentados, tanto por el rango de edad como por el uso del dispositivo. Finalmente en el rango de selectivos, podrían entrar los abuelos de los niños, ya que normalmente se hacen cargo de los niños cuando los padres trabajan, que por la edad y uso de los dispositivos estarían en este grupo.

### 3.6 . Competencia:

Sacar una nueva aplicación al mercado no es una tarea banal, hay un largo camino que recorrer con muchos obstáculos y baches que superar para llegar a la meta y lograr los objetivos. Uno de estos obstáculos para tener éxito es la competencia, ya que al haber un gran mercado de aplicaciones, prácticamente siempre habrá alguna aplicación con la que tendremos que competir de cara al público objetivo y finalidad que nos hemos marcado. Para nuestra aplicación de **Bits de Inteligencia**, hay varias aplicaciones muy similares, que usan el mismo método didáctico, por lo que tienen el mismo fin y público objetivo, por lo que son estrecha y firme competencia. Cabe destacar las siguientes aplicaciones, que se han encontrado al buscar Bits de Inteligencia en *Google Play Store*:

- **Bits de Inteligencia – FULL**: Aplicación de *Koy Group* con un coste de 1,20 €. No instalo ni pruebo esta aplicación.
- **Bits de Inteligencia – LITE**: Aplicación de *Koy Group* gratuita. Instalo dicha aplicación para ver su contenido y funcionamiento. Es una versión de prueba de la versión FULL en la que hay accesibles dos galerías de imágenes pero se muestran los títulos delas otras galerías disponibles. La aplicación está en español, las imágenes de los bits son dibujos y dentro de las categorías de bits, las pantallas con los bits pasan de forma automática. Permite la opción de mostrar el texto del bit en mayúsculas, minúsculas o no mostrarlo y dispone de un contador para cada categoría.
- **Bits de inteligencia**: Aplicación de *Educa Kids*. Instalo dicha aplicación para ver su contenido y funcionamiento. La aplicación es gratuita pero contiene publicidad, en forma de una barra inferior y mostrando pantallas de publicidad cada pocas pantallas. Permite la selección de varios idiomas para los bits (alemán, inglés, francés, portugués, español e italiano), dispone de varias categorías de bits y dentro de cada categoría hay que pasar los bits de forma manual.
- **Bits de Inteligencia n Inglés**: Aplicación de *Editorial Septiembr*e. No instalo ni pruebo esta aplicación.
- **BITS de mates, CANTIDADES**: Aplicación de *Educamigos & Smartbrain* con un coste de 9,99 €. No instalo ni pruebo esta aplicación.

Todas estas aplicaciones tienen en común el uso del método didáctico de los Bits de Inteligencia y que no son gratuitas al 100% ya que conllevan algún tipo de coste, bien pagando por la descarga de la aplicación o compras posteriores o bien porque tienen publicidad integrada, que no es un coste monetario, pero es un coste temporal y de atención.

4 . Estudio Técnico:

El equipo técnico en el que me centraré en este apartado consta de los elementos físicos requeridos para el proyecto. En este caso al ser un proyecto en el que soy solo trabajo yo y hago todas las tareas, no se necesita un local dedicado para el proyecto por ejemplo. Los elementos indispensables son:

- Ordenador: En este caso he usado un ordenador portátil, con que el realizar las tareas de análisis, diseño, búsqueda de información, desarrollo y documentación.
- Teléfono inteligente: para realizar pruebas en un dispositivo físico y no solo virtual.
- Conexión a Internet: Aunque no es un elemento físico como tal, lo incluyo en este apartado ya que es necesario para la realización del proyecto.
- Una habitación donde realizar el trabajo, que al realizar el trabajo en casa, vale una habitación que se use a modo de despacho o que tenga un escritorio para poder trabajar.
- Materiales de papelería como papel en blanco, bolígrafos de colores, una pizarra con rotuladores, etc. para poder realizar los análisis, diseños, esquemas, etc. antes de comenzar a desarrollar.

Como se puede ver no son necesario muchos elementos físicos para la realización del proyecto, sobre todo teniendo en cuenta que se solo hay una persona trabajando y además trabaja desde casa.

5 . Organización del negocio:

En esta sección primero hay que distinguir entre si uno de los objetivos de la aplicación es el ánimo de lucro con una monetización de la aplicación o no. Ya que si pretendemos ganar dinero con la aplicación, hay que hacerlo de forma legal y tener en cuenta los aspectos fiscales correspondientes.

Aunque inicialmente la aplicación será gratuita, no se descarta que se cambien a un modelo en el que se tenga un *feedback* económico, poniendo la aplicación a un precio simbólico de 0'99 €.

Teniendo en cuenta esa posibilidad de cobrar por la aplicación, estaríamos ejerciendo una actividad empresarial con ánimo de lucro, por lo que hay que estudiar qué tipo de empresa o sociedad nos conviene, pudiendo ser autónomo, sociedad limitada, sociedad anónima u otros tipos.

Inicialmente lo más conveniente sería establecerse como autónomo, ya que solo hay un miembro en todo el proceso que realiza todas las tareas y toma las decisiones. Esto favorece la rápida respuesta ante cambios que probablemente se producirán, ya que estamos en un mercado muy activo en constante evolución en función de las necesidades de los clientes. Las ventajas de establecerse como autónomo respecto a establecerse como una sociedad son:

- Proceso de formalización rápido, sencillo y sin involucrar mucho coste. Se pueden hacer uso de las ventanillas únicas que facilitan el proceso.
- No es necesario un capital inicial para establecerse como autónomo.
- El autónomo tiene responsabilidad directa sobre sus proveedores. Esto significa que en caso de impagos o quiebra, la persona física responderá con sus propios bienes.

Ese es el paso inicial, si el negocio va bien y avanza y se hace necesario mantenerla a un nivel alto o la creación de nuevas aplicaciones, puede surgir el caso de la necesidad de contar con más personal especializado. Para ello lo mejor sería constituir una sociedad, donde lo idóneo sería una Sociedad Limitada, ya que el paso de autónomo a sociedad limitada tiene las siguientes ventajas:

- La responsabilidad recae sobre la sociedad en vez de sobre la persona física, por lo que mantendría a salvo su patrimonio personal.
- Menor carga fiscal, el impuesto de sociedades es del 30% mientras que el tipo que pueden llegar a pagar los autónomos es del 48%.
- Se facilita la contratación de empleados y existen ventajas fiscales y deducciones en el impuesto de sociedades si se realizan contrataciones.

Ahora que ya sabemos las opciones que hay para establecernos como empresa para poder tener un beneficio con la aplicación, entendemos que la aplicación y por lo tanto la empresa ha

tenido éxito y que una única persona no se podrá hacer cargo de todo por lo que hay que tener un plan de recursos humanos (RRHH).

### 5.1 . Plan de organización y recursos Humanos:

El objetivo fundamental de un plan de organización y de recursos humanos es el de establecer el organigrama empresarial que defina la jerarquía de mando, establecer las funciones a realizar y determinar quién tiene que realizar cada función, aunque también incluye otros objetivos relacionados con el personal como características necesarias, formación, etc.

Llegados a este apartado supongo, que la empresa ha tenido éxito y está creciendo por lo que habrá que formar un grupo personal para trabajar con los siguientes cargos:

- **Gerente**: Será la cabeza y el responsable de la empresa. Deberá tomar las decisiones más importantes de cara a los planes y objetivos de la empresa. Sus labores fundamentales serían:
  - Definir los objetivos y planes de operaciones.
  - Estudio de la evolución del mercado y la competencia.
  - Control y gestión de cuentas y balances junto con el administrador.
  - Evaluación de la empresa y del personal.
- **Administrador**: Sus labores estarán enfocadas a temas de administración y contables como por ejemplo la gestión de las nóminas, control de balances y cuentas, control de cobros y pagos a proveedores, etc.
- **Analista**: Será el encargado de plasmar a objetivos y tareas las ideas que le proponga el gerente y ver si son factibles o no, realizar el control y seguimiento de las tareas propuestas, encargado de I+D+I, deberá estar en contacto tanto con el gerente para recibir las ideas y actualizaciones del mercado y funcionalidades, como el diseñador y desarrollador para llevar a cabo las tareas y aplicaciones.
- **Diseñador**: Será el encargado de realizar los diseños necesarios, tanto para el responsable de marketing a la hora de realizar campañas publicitarias y de marketing, para el *community manager* para los recursos necesarios para la página web y las redes sociales, como para realizar los diseños requeridos por la aplicación para lo que tendrá que estar en contacto con el analista y el desarrollador.
- **Responsable de Marketing**: Será el responsable de la creación y gestión del marketing de la empresa. En función de la carga de trabajo que tenga, podrá realizar también las tareas de *Community Manager.* Si lo considera oportuno podrá realizar campañas de marketing y *merchandising*, realizándolas él mismo o subcontratando a empresas externas especializadas.
- **Community Manager**: Será en el encargado de la vida de la empresa y las aplicaciones en el ámbito de las redes sociales, ya que son una fuente de publicidad y marketing. En caso de no ser la misma persona, estará en estrecha relación con el responsable de Marketing. Se encargara también de la página web (creación, actualización y posicionamiento). Deberá realizar informes periódicos del estado de la web y la vida en las redes sociales.

- **Desarrollador**: Será el encargado de desarrollar la aplicación, mantenerla y ejecutar las mejoras o las ampliaciones. Tendrá que estar en continuo contacto con el analista y el diseñador gráfico para entre todos estudiar la viabilidad de las nuevas ideas y propuestas y llevarlas a cabo.

Además de estos puestos descritos, se podrá hacer uso de ayudas externas a la empresa para temas concretos, contratando a empresas especialistas en Marketing o en Asesoría Legal por ejemplo.

6 . Estudio de la financiero:

Para realizar un estudio financiero es necesario conocer al máximo el mercado en el que se va a mover la empresa y los potenciales usuarios que hay en el mercado. Como ya he explicado, el mercado de las aplicaciones móviles está en constante cambio y evolución, por lo que conocer el bien el mercado es muy complicado ya que esté en constante cambio y es complicado obtener un escenario realista, debido a ello tenemos una incertidumbre máxima en el mercado. Debido a estas características, voy a emplear el **método del punto de equilibro**. Este método busca el punto de equilibrio donde los ingresos generados por la aplicación son iguales a los costes. Una vez que se obtiene el punto de equilibrio, sabemos el número de descargas que necesitamos para cubrir los costes y obtener beneficios.

Para obtener cual es el punto de equilibrio, primero necesitamos saber cuáles son los gastos para obtener la aplicación. Si volvemos al apartado "**3.8.5 Coste total de la aplicación**" vemos que el coste de la aplicación es de **9.154,91 €**. A esta cantidad habría que sumar la cuota de autónomos de 2016 que es de 267,04 €, una campaña inicial de marketing y *merchandising* por valor de 300 € y el precio de las tasas para registrar la marca a nivel mundial cuyo coste es de 185 €. Todo ello sumando nos da un coste inicial de 9.906,95 €.

Para encontrar el punto de equilibrio, necesitamos encontrar un número fijo de descarga/venta de aplicaciones, ya que el método definido para obtener ingresos es la venta de la aplicación a un precio de 0,99 € sin tener publicidad la aplicación por lo que no habrá ingresos por publicidad. Estos ingresos son brutos, habría que descontar las retenciones, pero voy a calcular el punto de equilibrio con ese ingreso por aplicación.

Teniendo en cuenta el coste inicial de la aplicación y los ingresos por aplicación, obtenemos que para que la aplicación sea rentable y el negocio sea viable hay que vender un mínimo de 10.008 aplicaciones. Este número es muy elevado teniendo en cuenta que es una aplicación nueva y desconocida. Sobre todo teniendo en cuenta que las aplicaciones similares de la competencia tienen un rango de descargas entre 500 y 1000.

Teniendo en cuenta estos datos y el estudio realizado por *App Promo* que llega a la conclusión de que el 59% de las aplicaciones no supera la inversión y que el 80% de los desarrolladores no puede sostener su negocio solo en este mercado, nos inclina a pensar que nuestra aplicación no va a ser rentable salvo que se produzca prácticamente un milagro.

6.1 . Estudio de la inversión:

La inversión inicial para el desarrollo del proyecto está definida en el punto "**3.8.5 Coste total de la aplicación**" donde vemos que el coste de desarrollo de la aplicación es de **9.154,91 €**. A esta cantidad habría que sumar los gastos de autónomos, la campaña inicial de marketing y la tasa de registro de la marca, como hemos visto en el punto anterior. Eso en el caso de que se quiera poner un precio a la aplicación y haya que darse de alta como empresa o autónomo.

6.2 . Estudio de los ingresos y gastos:

En el estudio de los ingresos, hay que destacar que solo tenemos una fuente de ingresos, que es la venta de aplicaciones, ya que la aplicación no contiene publicidad que pueda generar algún tipo de ingresos después de la venta de la aplicación. Por el lado contrario, los gastos, si quitamos el coste inicial de desarrollo de la aplicación, nos quedan los gastos que originen la empresa o autónomo como la cuota de autónomo o el impuesto de sociedad y las cantidades de IVA que habría que pagar, que van en función de la facturación y por lo tanto de la venta de aplicaciones.

7 . Estrategia de Marketing y ventas:

La estrategia de marketing y su posterior plan de marketing, lo que buscan es tanto la captación de nuevos clientes como la fidelización y satisfacción de los clientes existente.

Una buena estrategia de Marketing se basa en:

- **Posicionamiento**: Con el posicionamiento lo que se quiere conseguir es que el público objetivo conozca nuestra marca y la vea como algo positivo, de calidad y de valor, para conseguir una ventaja competitiva respecto a la competencia. Hay que intentar adelantarse a lo que quiere el cliente para crearle una necesidad y ofrecerle nuestra aplicación para satisfacer esa necesidad.
- **Comunicación**: Para conseguir el objetivo de crear una necesidad al cliente y tener un buen posicionamiento, la comunicación es fundamental, hay que dar a conocer el producto y su calidad para que el cliente lo desee. Podemos definir varias formas de comunicación con el objetivo de conseguir el mayor impacto y difusión posible con el mínimo coste. Para ello se pueden optar por estrategias en Internet, fundamentalmente en las redes sociales, creando perfiles corporativos para la aplicación donde realizar campañas publicitarias y permitir al usuario expresar su opinión, estrategias de publicidad, por ejemplo repartir panfletos publicitarios en las guarderías y colegios cuando los padres van a recoger a sus hijos o poner publicidad en catálogos de juguetes que vean los niños o estrategias de promoción como poner un dispositivo con nuestra aplicación para que se pueda utilizar como ejemplo en jugueterías o comercios donde haya gran afluencia de niños.
- **Producto**: Hay que tener un producto, en este caso una aplicación que sea útil, atrayente, funcional y de calidad. Además tiene que tener un nombre llamativo y distintivo que permita crear una imagen de marca y atraer a los clientes. Se ha elegido el nombre de "Bits de Inteligencia", que aunque no es muy llamativo, define perfectamente la aplicación y es fácil de encontrar en búsquedas ya que es el nombre del método didáctico. Hay que tener especial cuidado en el icono de la aplicación, ya que es lo primero que se ve una vez instalada y la descripción y capturas de pantalla que se ponen en la página de

*Google Play Store* dedicada a nuestra aplicación para que sea atrayente y el potencial cliente instale la aplicación.

- **Distribución y ventas**: Para la distribución y venta de la aplicación, se va a utilizar la plataforma que Google tiene para tal fin, que es el *market Google Play Store*. Es un servicio ampliamente utilizado y de referencia en el sector, ya que todos los dispositivos Android lo tienen instalado de serie y los usuarios recurren a este servicio para buscar cualquier tipo de aplicación. Se ofrecerá la versión completa de la aplicación a un precio de 0,99 €, ya que es un precio que está como barrera mental en los usuario y son reticentes a superar esa barrera y a comprar aplicaciones que superen ese precio. En función de la repercusión de la aplicación, si no tiene muchas descargas, podría plantearse la idea de ofrecer una versión gratuita de prueba, sin publicidad pero solo con alguna de las galerías de Bits disponibles para que el usuario la pruebe y si le gusta que descargue la aplicación completa.

8 . Principales Riesgos y Estrategias de Salida:

Esta sección está dedicada a explicar los posibles riesgos derivados de la actividad, las posibles contingencias o medidas correctoras que se puedan aplicar a los riesgos que aparezcan y en caso necesario, alguna estrategia de salida en caso de que los riesgos se vuelvan insalvables y haya que desistir del plan de la actividad.

Los posibles riesgos que nos podemos encontrar son los siguientes:

- **Descargas superiores a lo previsto**: El número de descargas de la aplicación es superior al previsto. Habría que realizar un análisis del crecimiento de la aplicación y valorar si es necesario contratar personal de forma ordenada para ofrecer un correcto servicio a los usuarios y planear mejoras y actualizaciones para que el ritmo de descargas no disminuya.
- **Descargas inferiores a lo previsto**: El número de descargas de la aplicación es inferior al previsto. Habría que realizar un análisis de la aplicación y valorar por que no se están consiguiendo las descargadas deseadas. Habría que valorar realizar más campañas de marketing o hacerlas de otro forma y si fuera necesario, llegar a hacer un recorte de la plantilla.
- **Valoraciones de la aplicación**: Nos encontramos con que las valoraciones que hacen los usuarios de la aplicación son negativas en el *market Google Play Store*. Habría que analizar detalladamente cada uno de los comentarios negativos para ver que en está fallando la aplicación y por qué no gusta a la gente. Una vez estudiados los fallos por parte de toda la empresa (gerente, analista, diseñador, desarrollador), habría que proponer posibles soluciones mediante modificaciones y adaptaciones para ajustarnos a los gustos de los usuarios.
- **Cambio de tendencia de los sistemas operativos**: Este riesgo surgiría cuando la tendencia actual cambiara y Android no fuera la plataforma dominante, en favor de otras plataformas en alza. Habría que realizar un análisis de tendencias para ver cuál sería la nueva plataforma dominante y con mayor proyección, para entonces formar a la plantilla en esa nueva plataforma y adaptar la aplicación a dicha nueva plataforma.
- **Competencia**: Riesgo que aparecería cuando surgieran nuevas aplicaciones en competencia directa con nuestra aplicación, bien usando el mismo método didáctico o

que tengan un objetivo y características similares. Habría que revisar las estrategias y plan de marketing, para seguir estando bien posicionados en el mercado y atraer clientes hacia nuestra aplicación en vez de a las de la competencia, con nuevas campañas, por ejemplo.

- **Legalidad**: Este riesgo surgiría si hubiera problemas con temas legales. Siempre hay que estar de acuerdo con lo que dicta la ley, por lo tanto habría que estar pendientes de cualquier cambio o modificación que se produjera en el marco legal que nos afecta, para realizar los cambios que se requirieran para no cometer ninguna infracción. También hay que tener en cuenta el marco legal a la hora de hacer modificaciones en la aplicación ya que alguna modificación podría requerir de nuevos ámbitos legales, como por ejemplo cumplir con la ley de protección de datos si en alguna actualización se requiere algún dato personal del usuario.
- **Fiscalidad**: Riesgo que surgiría si se produjeran cambios en las políticas fiscales que afectan a las aplicaciones móviles, a *Google Play Store* o cambios en las normativas de sociedades y autónomos. Habría que analizar los cambios que se han producido, seguramente con ayuda de una Asesoría legal externa y adaptarse a dichos cambios.
- **Personal**: En cuanto al personal, el riesgo sería que hubiera que contratar a alguien de forma urgente, para ello se realizaría un proceso de selección, con diversas entrevistas y valoraciones en las que intervendrían los miembros importantes del equipo para escoger al candidato idóneo. Otro riesgo sería el abandono de alguien del personal, para prevenirlo habría que incluir cláusulas en los contratos de que si van a abandonar la empresa tienen que avisar con un tiempo de antelación prudente para encontrar un sustituto y si se produjera de forma inmediata e inesperada, el miembro inmediatamente inferior o superior en el escalafón lo sustituiría pero consultaría parte de las decisiones a tomar con los otros miembros del equipo que se vean afectados por las decisiones a tomar.

## *6.3.- Facturación de Google Play Store*

En este apartado vamos a ver como gestiona Google el IVA de las ventas realizadas:

1. **Dentro de la Unión Europea**: Hay que pagar el 21% de IVA. Se puede configurar *Google Checkout* para que añada automáticamente el 21% a las ventas dentro de la Unión Europea, o descontarlo nosotros posteriormente.
2. **Fuera de la Unión Europea**: Si el comprador está fuera de la Unión europea, no se le puede cobrar el IVA y además tenemos que estar inscritos en el registro de operadores intracomunitarios.

En *Google Play Store*, los pagos se hacen a través de la plataforma *Google Checkout*, que mensualmente te remite un listado con las descargas/ventas de tu aplicación, haciendo más fácil su contabilización.

En cuanto a las comisiones de Google, ya que Google no te emite ninguna factura, es aconsejable introducir una factura a nombre de Google pagando las comisiones que descuenta de las ventas, correspondientes al 30%.

# 7.- Diferencias entre Modelo de Negocio y Plan de Negocio

Ahora que ya hemos visto cada concepto por separado, está bien aclarar las diferencias entre ellos.

El **modelo de negocio** se utiliza desde la etapa de "concepción de la idea" hasta que logras generar y validar tu proyecto**.** Esta herramienta te ayudara a analizar el problema, el mercado de tu proyecto, la solución propuesta, los canales de distribución, costos e ingresos.

Todas las ideas y negocios son diferentes. No existe un tiempo específico para dejar de usar esta herramienta. Normalmente, tu modelo de negocio debería acompañarte en tu camino de emprendedor hasta que llegue a encontrar el producto o servicio final.

Un **plan de negocio** es necesario cuando ya hayas encontrado las bases que te ayudaran a escalar el negocio. Es decir, se basa en el modelo de negocio y su enfoque y construye encima.

El plan de negocio es algo estratégico y táctico que marca la dirección que quieres que tome tu proyecto (empresa, producto o servicio). Especifica las metas y los objetivos que tienes y como cumplirlas. Tendrás que el tiempo necesario para realizarlo y pensar en el futuro de tu proyecto. Si estas empezando y no tienes las bases de tu modelo de negocio, ni pienses en escribir un plan de negocios. Solo estarás perdiendo tu tiempo.

## *7.1.- Interdependencia*

Por lo que hemos visto hasta ahora, el plan de negocios es completamente dependiente del modelo de negocios. Este modelo explica el flujo de dinero dentro de la compañía y el plan de negocios explica a la estructura necesaria para obtener ese flujo. Si cambias de modelo, también necesitarás cambiar de plan. Aunque puede que no tengas que cambiar todo el plan de negocios, puede que tengas que hacer cambios en el personal, equipo, ubicación y marketing. Estos cambios también pueden requerir más cambios en la parte financiera si se requieren compras adicionales u otros cambios drásticos.

## *7.2.- Cambios*

El cambio es una fuerza común en el mundo de los negocios. Los negocios deben poder adaptarse a los cambios de la industria y a la demanda de sus clientes. No adaptarse a este cambio puede resultar en pérdidas de clientes y en una reducción de los márgenes de ganancias. Es importante que tus decisiones sobre el cambio se alineen con tu modelo de negocios para evitar desperdiciar tiempo y recursos. Además, es importante que actualices tu plan de negocios para reflejar los cambios en tu negocio. Actualizar a tu plan de negocios no sólo te ayudará a mantenerte en buen camino hacia tus metas y misiones, también te ayudará a observar el progreso de tu negocio y a desarrollar nuevas estrategias hacia el éxito.

## *7.3.- Conclusiones*

Para concluir con este apartado, hay que destacar que tanto el modelo de negocio como el plan de negocio son importantísimos para cualquier proyecto y dependientes uno del otro ya que el modelo de negocio fija lo que quieres hacer y en función de ese modelo se construye el plan de negocio que dicto como llevarlo a cabo.

Además de esta importancia y dependencia entre ellos, hay que recalcar que ambos tienen que estar abiertos a posibles cambios y modificaciones para adaptarse a los cambios de los consumidores y del mercado.

# SECCIÓN II: DOCUMENTACIÓN TÉCNICA

En esta sección nos metemos de lleno en el apartado análisis y diseño de nuestra aplicación. Esta labor es fundamental y esencial ya que es donde se va a apoyar el resto del proyecto. Un buen estudio y análisis del problema y de los requisitos y el posterior diseño de los procedimientos a seguir, pueden marcar la diferencia entre el éxito y el fracaso, ya que pueden marcar que el producto o servicio que realicemos, en este caso una aplicación para Android sea bueno, mediocre o malo y eso influye en la aceptación en el mercado y en el éxito.

A lo largo de esta parte de la documentación, estudiaremos todos los pasos necesarios para analizar y describir la aplicación y saber a qué atenernos y como organizarnos a la hora del desarrollo. Sin más preámbulos, voy a empezar.

# 1.- Especificación de Requisitos

La aplicación se diseña para que sea usada fundamentalmente por un público infantil, aunque también puede ser usada por público adulto acompañando al infantil o por pura diversión del mencionado público adulto. En lo que se refiere al funcionamiento de la aplicación, debe permitir seleccionar un idioma, seleccionar una categoría de bits y en caso necesario una subcategoría, mostrar la lista de bits y volver al menú. Por tanto, los requisitos son los siguientes:

1. Atraer visualmente.
2. Uso fácil e intuitivo.
3. Ser amigable.
4. Seleccionar idioma.
5. Seleccionar Categoría de Bits.
6. Seleccionar Subcategoría de Bits, si la Categoría seleccionada está compuesta por varias subcategorías.
7. Mostrar la galería de Bits selecciona dentro de la Categoría o Subcategoría.
8. Volver al menú cuando se termina la galería de Bits.

# 2.- Requisitos Funcionales

Los requisitos funcionales describen la labor que tiene que realizar la aplicación, es decir, son los requisitos que marcan la función que ha de llevar a cabo la aplicación. Van estrechamente ligados a los casos de uso ya que describen la labor que realiza cada caso de uso.

Partiendo de que el usuario ha iniciado la aplicación y que automáticamente se muestra la pantalla de inicio, los requisitos funcionales son:

- **Comenzar**: La aplicación dejará la pantalla de inicio para pasar a la siguiente pantalla que nos permite seleccionar el idioma.
- **Seleccionar Idioma**: Se mostrará una pantalla con un botón para idioma disponible (Español e Inglés) y por medio de la pulsación en el botón correspondiente se elegirá por un idioma para la aplicación.
- **Seleccionar Categoría**: Se mostrará en pantalla una lista de las categorías disponibles. Cada categoría estará representada por medio de un icono visual, una numeración y un nombre. Si la categoría no tiene ninguna subcategoría, aparecerá además un contador que indique el número de veces que se ha visualizado la categoría. Se podrá seleccionar cualquier categoría de la lista y entonces nos mostrará la lista de Bits correspondientes a esa categoría o la lista de subcategorías, en función de si la categoría seleccionada tiene subcategorías o no.
- **Seleccionar Subcategoría**: Una vez seleccionada una categoría que esté compuesta por varias subcategorías, se mostrará en pantalla una lista con las subcategorías disponibles. Cada subcategoría estará representada por medio de un icono visual, una numeración, un nombre y un contador que indique el número de veces que se ha visualizado la subcategoría. Se podrá seleccionar cualquier subcategoría de la lista y entonces nos mostrará la lista de Bits correspondientes a esa subcategoría.
- **Mostrar Galería de Bits**: Una vez que se ha seleccionado la categoría deseada, o categoría y subcategoría, se mostrará en pantalla la lista de Bits correspondientes. Cada Bit estará representado en una pantalla diferente. La transición de un Bit a otro se realizará de forma automática. Cada pantalla de representación de un Bit estará compuesta por la imagen del Bit, por el texto que indique el nombre del Bit y por un sonido que nos indicara el nombre del Bit. El texto se representará en función de la configuración que esta seleccionada, pudiendo estar el texto en mayúsculas, minúsculas o no aparecer. Al finalizar la secuencia de Bits, se incrementará el contador que nos indica el número de veces que hemos visto la galería y se volverá a la pantalla anterior, de selección de categorías o subcategorías, la que corresponda.
- **Reinicializar contadores**: Mediante una opción en el menú que aparece al pulsar el botón de opciones del dispositivo, se nos permitirá poner todos los contadores de las categorías y subcategorías a cero, todos a la vez.

- **Configuración del texto**: Mediante una opción en el menú que aparece al pulsar el botón de opciones del dispositivo, se nos permitirá configurar como deseamos que aparezca el texto en la pantalla de representación. Al seleccionar esta opción, deberá aparecer otra pantalla con tres opciones, que nos permitan seleccionar entre texto en mayúsculas, texto en minúsculas o sin texto para que no aparezca el texto. Se deberá guardar esta opción y mantenerse hasta que se vuelva a cambiar la configuración del texto mediante el mismo procedimiento. Por defecto la aplicación tendrá configurado que el texto aparezca en minúsculas.
- **Sonidos**: Los sonidos de cada Bit deben ser perfectamente audibles y claros.
- **Imágenes**: Las imágenes de cada Bit deben ser perfectamente visible, sobre un fondo blanco y mostrar solo el elemento del Bit en cuestión (concreta y concisa).

# 3.- Requisitos No Funcionales

Los requisitos no funcionales son un tipo de requisitos que aunque no marcan y definen funciones y tareas concretas a realizar por la aplicación, nos marcan otro tipo de requisitos que se deben cumplir para que la aplicación pueda funcionar. Estos requisitos para nuestra aplicación son:

- **Tamaño de la aplicación**: De acuerdo con las directiva marcadas por *Google* para poder distribuir aplicaciones desde su *Play Store*, el archivo ejecutable *APK* de la aplicación no puede superar los 100 MB (antes de Noviembre de 2015 este límite era de 50 MB), aunque el tamaño total de la aplicación, con los archivos de expansión puede llegar al límite de 4 GB.
- **Tamaño de RAM**: Se busca una aplicación optimizada que puedan ejecutar todos los terminales, incluidos los de gama baja, por lo que el tamaño de RAM necesario será de 512 MB que todos los dispositivos actuales cumplen.
- **Adaptabilidad**: La aplicación se tiene que poder ejecutar en dispositivos con cualquier tamaño de pantalla, ajustándose a dicho tamaño.
- **Usabilidad**: La aplicación tiene que ser simple e intuitiva para que se pueda usar sin necesidad de aprendizaje previo.
- **Fiabilidad**: La aplicación no tiene que tener errores que interrumpan o impidan su funcionamiento.
- **Rapidez**: La transición entre pantallas tiene que ser rápida y sin tiempos muertos ni tiempos de carga.
- **Versión de Android**: La aplicación se podrá ejecutar en una versión de Android 4.0 o superior.

# 4.- Diagramas de Casos de Uso

Entendemos por diagrama de caso de uso una representación gráfica que muestra las posibles interacciones (usos) que pueden hacer los usuarios o actores con el sistema (nuestra aplicación), en respuesta a una interacción que el usuario hace son el sistema. Es decir, después de una interacción se produce otro caso de uso que producirá otra interacción donde se podrá elegir una acción u otra. Refiriéndonos a nuestra aplicación en concreto, una explicación más fácil sería que cada vez elegimos una opción en una pantalla, llegamos a otra pantalla donde podemos realizar otra acción.

Como la aplicación tiene que ser lo más fácil de usar posible, en cada caso de uso (pantalla) solo se podrá realizar una acción, salvo en el menú principal de categorías que mediante el botón de opciones del teléfono se podrán elegir más acciones. Además siempre está la opción del botón volver atrás del dispositivo.

Para caso de uso vamos a detallar cuando se produce, su diagrama, nombre, objetivo, actor, flujo de datos principal, flujo de datos excepcional, precondición, postcondición y escenario.

Ahora empezamos con la descripción de los casos de uso:

1. **Caso de uso cuando el usuario está en la pantalla de bienvenida**:

Se produce cuando el usuario ejecuta la aplicación y llega a la pantalla de inicio del programa.

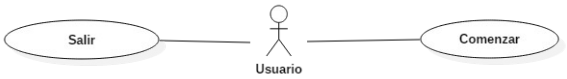


*Ilustración 30: Casos de uso en la pantalla de inicio*

Los detalles de estos casos de uso son:

| **Nombre** | Comenzar |
|---|---|
| **Objetivo** | Pasar a la siguiente pantalla de la aplicación |
| **Actor** | Usuario |
| **Flujo de datos principal** | El usuario pasa a la siguiente pantalla |
| **Flujo de datos excepcional** | La aplicación se cierra o pasa a segundo plano en favor de otra aplicación |
| **Precondición** | Ejecutar la aplicación y estar en la pantalla inicial |
| **Postcondición** | ninguna |
| **Escenario** | El usuario ejecuta la aplicación y llega a la pantalla de presentación |

*Tabla 12: Caso de Uso Continuar en la pantalla inicial*

| **Nombre** | Salir |
|---|---|
| **Objetivo** | Salir de la aplicación |
| **Actor** | Usuario |
| **Flujo de datos principal** | El usuario sale de la aplicación |
| **Flujo de datos excepcional** | La aplicación se cierra o pasa a segundo plano en favor de otra aplicación |
| **Precondición** | Presionar el botón atrás del dispositivo estando en la pantalla inicial |
| **Postcondición** | ninguna |
| **Escenario** | El usuario está en la pantalla de presentación |

*Tabla 13: Caso de Uso Salir en la pantalla inicial*

2. **Caso de uso cuando el usuario está en la pantalla de selección de idioma**:

Se produce cuando el usuario presiona el botón continuar de la pantalla de inicio del programa para pasar a la siguiente pantalla.

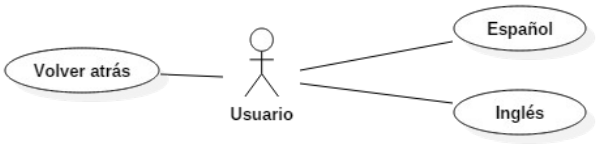


*Ilustración 31: Casos de uso en la pantalla de selección de idioma*

Los detalles de estos casos de uso son:

| **Nombre** | Español |
|---|---|
| **Objetivo** | Seleccionar el idioma Español para los Bits |
| **Actor** | Usuario |
| **Flujo de datos principal** | El usuario selecciona el idioma Español y pasa a la siguiente pantalla con la lista de Categorías |
| **Flujo de datos excepcional** | La aplicación se cierra o pasa a segundo plano en favor de otra aplicación |
| **Precondición** | Haber pasado de la pantalla inicial |
| **Postcondición** | Se guarda el idioma Español como seleccionado |
| **Escenario** | El usuario está en la pantalla de selección de idioma |

*Tabla 14: Caso de Uso Español en la pantalla de Selección de Idioma*

| **Nombre** | Inglés |
|---|---|
| **Objetivo** | Seleccionar el idioma Inglés para los Bits |
| **Actor** | Usuario |
| **Flujo de datos principal** | El usuario selecciona el idioma Inglés y pasa a la siguiente pantalla con la lista de Categorías |
| **Flujo de datos excepcional** | La aplicación se cierra o pasa a segundo plano en favor de otra aplicación |
| **Precondición** | Haber pasado de la pantalla inicial |
| **Postcondición** | Se guarda el idioma Inglés como seleccionado |
| **Escenario** | El usuario está en la pantalla de selección de idioma |

*Tabla 15: Caso de Uso Inglés en la pantalla de Selección de Idioma*

| **Nombre** | Volver atrás |
|---|---|
| **Objetivo** | Volver |
| **Actor** | Usuario |
| **Flujo de datos principal** | El usuario sale de la aplicación |
| **Flujo de datos excepcional** | La aplicación se cierra o pasa a segundo plano en favor de otra aplicación |
| **Precondición** | Presionar el botón atrás del dispositivo estando en la pantalla inicial |
| **Postcondición** | ninguna |
| **Escenario** | El usuario está en la pantalla de presentación |

*Tabla 16: Caso de Uso Volver atrás en la pantalla de Selección de Idioma*

3. **Caso de uso  cuando el usuario está en la pantalla de Categorías Principal**:

Se produce cuando el usuario ha seleccionado un idioma en la pantalla de selección de idioma y continúa la ejecución de la aplicación.

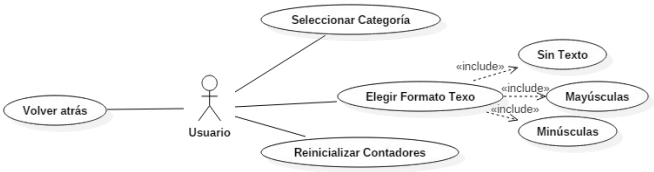


*Ilustración 32: Casos de uso en la pantalla de Categorías*

Los detalles de estos casos de uso son:

| **Nombre** | Seleccionar Categoría |
|---|---|
| **Objetivo** | Seleccionar una categoría concreta de Bits |
| **Actor** | Usuario |
| **Flujo de datos principal** | El usuario selecciona una categoría y pasa a la siguiente pantalla (Galería de Bits o Seleccionar Subcategoría) |
| **Flujo de datos excepcional** | La aplicación se cierra o pasa a segundo plano en favor de otra aplicación |
| **Precondición** | Estar en la pantalla de selección de categorías |
| **Postcondición** | Se almacena la categoría seleccionada |
| **Escenario** | El usuario está en la pantalla de Categorías |

*Tabla 17: Caso de Uso Seleccionar Categoría en la pantalla de Selección de Categoría*

| **Nombre** | Reinicializar Contadores |
|---|---|
| **Objetivo** | Poner a cero los contadores de las Categorías y Subcategorías |
| **Actor** | Usuario |
| **Flujo de datos principal** | El usuario selecciona la opción Reinicializar Contadores disponible al presionar el botón de opciones del dispositivo |
| **Flujo de datos excepcional** | La aplicación se cierra o pasa a segundo plano en favor de otra aplicación |
| **Precondición** | Presionar el botón de opciones del dispositivo |
| **Postcondición** | Los contadores tienen cero como valor |
| **Escenario** | El usuario está en la pantalla de Categorías |

*Tabla 18: Caso de Uso Reinicializar Contadores en la pantalla de Selección de Categoría*

| **Nombre** | Elegir Formato Texto |
|---|---|
| **Objetivo** | Seleccionar el formato del texto que aparecerá en cada uno de los Bits de la Galería de Bits |
| **Actor** | Usuario |
| **Flujo de datos principal** | El usuario selecciona la opción Configuración Texto disponible al presionar el botón de opciones del dispositivo |
| **Flujo de datos excepcional** | La aplicación se cierra o pasa a segundo plano en favor de otra aplicación |
| **Precondición** | Presionar el botón de opciones del dispositivo |
| **Postcondición** | Seleccionar un formato de configuración de las que aparecerán en pantalla |
| **Escenario** | El usuario está en la pantalla de Categorías |

*Tabla 19: Caso de Uso Elegir Formato Texto en la pantalla de Selección de Categoría*

| **Nombre** | Sin Texto |
|---|---|
| **Objetivo** | Configurar que no aparezca texto en los Bits de la Galería de Bits |
| **Actor** | Usuario |
| **Flujo de datos principal** | El usuario selecciona la opción Sin Texto de las opciones que aparecen en pantalla y se almacena esta configuración |
| **Flujo de datos excepcional** | La aplicación se cierra o pasa a segundo plano en favor de otra aplicación |
| **Precondición** | Haber seleccionado la opción  Configuración Texto disponible al presionar el botón de opciones del dispositivo |
| **Postcondición** | Se guarda la configuración seleccionada |
| **Escenario** | El usuario está en la pantalla de Categorías |

*Tabla 20: Caso de Uso Sin Texto en la pantalla de Selección de Categoría*

| **Nombre** | Mayúsculas |
|---|---|
| **Objetivo** | Configurar que el texto que aparece en los Bits de la Galería de Bits aparezca en mayúsculas |
| **Actor** | Usuario |
| **Flujo de datos principal** | El usuario selecciona la opción Mayúsculas de las opciones que aparecen en pantalla y se almacena esta configuración |
| **Flujo de datos excepcional** | La aplicación se cierra o pasa a segundo plano en favor de otra aplicación |
| **Precondición** | Haber seleccionado la opción Configuración Texto disponible al presionar el botón de opciones del dispositivo |
| **Postcondición** | Se guarda la configuración seleccionada |
| **Escenario** | El usuario está en la pantalla de Categorías |

*Tabla 21: Caso de Uso Mayúsculas en la pantalla de Selección de Categoría*

| **Nombre** | Minúsculas |
|---|---|
| **Objetivo** | Configurar que el texto que aparece en los Bits de la Galería de Bits aparezca en minúsculas |
| **Actor** | Usuario |
| **Flujo de datos principal** | El usuario selecciona la opción Minúsculas de las opciones que aparecen en pantalla y se almacena esta configuración |
| **Flujo de datos excepcional** | La aplicación se cierra o pasa a segundo plano en favor de otra aplicación |
| **Precondición** | Haber seleccionado la opción Configuración Texto disponible al presionar el botón de opciones del dispositivo |
| **Postcondición** | Se guarda la configuración seleccionada |
| **Escenario** | El usuario está en la pantalla de Categorías |

*Tabla 22: Caso de Uso Minúsculas en la pantalla de Selección de Categoría*

| **Nombre** | Volver atrás |
|---|---|
| **Objetivo** | Volver a la pantalla anterior |
| **Actor** | Usuario |
| **Flujo de datos principal** | El usuario vuelve a la pantalla anterior de la aplicación |
| **Flujo de datos excepcional** | La aplicación se cierra o pasa a segundo plano en favor de otra aplicación |
| **Precondición** | Presionar el botón atrás del dispositivo estando en la pantalla inicial |
| **Postcondición** | ninguna |
| **Escenario** | El usuario está en la pantalla de Categorías |

*Tabla 23: Caso de Uso Volver Atrás en la pantalla de Selección de Categoría*

4. **Caso de uso cuando el usuario está en la pantalla de Subcategorías**:

Se produce cuando el usuario selecciona una Categoría en la pantalla de Categorías y dicha categoría está compuesta por varias subcategorías.


*Ilustración 33: Casos de uso en la pantalla de Subcategorías*

Los detalles de estos casos de uso son:

| **Nombre** | Seleccionar Subcategoría |
|---|---|
| **Objetivo** | Seleccionar una subcategoría concreta de Bits |
| **Actor** | Usuario |
| **Flujo de datos principal** | El usuario selecciona una subcategoría y pasa a la siguiente pantalla a ver la Galería de Bits |
| **Flujo de datos excepcional** | La aplicación se cierra o pasa a segundo plano en favor de otra aplicación |
| **Precondición** | Estar en la pantalla de selección de subcategorías |
| **Postcondición** | Se almacena la categoría seleccionada y se lanza la Galería de Bits |
| **Escenario** | El usuario está en la pantalla de Subcategorías |

*Tabla 24: Caso de Uso Seleccionar Subcategoría en la pantalla de Selección de Subcategoría*

| **Nombre** | Volver atrás |
|---|---|
| **Objetivo** | Volver a la pantalla anterior |
| **Actor** | Usuario |
| **Flujo de datos principal** | El usuario vuelve a la pantalla anterior de la aplicación |
| **Flujo de datos excepcional** | La aplicación se cierra o pasa a segundo plano en favor de otra aplicación |
| **Precondición** | Presionar el botón atrás del dispositivo estando en la pantalla inicial |
| **Postcondición** | ninguna |
| **Escenario** | El usuario está en la pantalla de Subcategorías |

*Tabla 25: Caso de Uso Volver Atrás en la pantalla de Selección de Subcategoría*

5. **Caso de uso cuando el usuario está en la pantalla de Galería de Bits**:

Se produce cuando el usuario selecciona una Categoría en la pantalla de Categorías y dicha categoría no tiene subcategorías, o cuando selecciona una subcategoría en la pantalla de Subcategorías. Los Bits van pasando de forma automática, solo existe la opción de volver atrás.


*Ilustración 34: Casos de uso en la pantalla de Galería de Bits*

Los detalles del caso de uso son:

| **Nombre** | Volver atrás |
|---|---|
| **Objetivo** | Volver a la pantalla anterior |
| **Actor** | Usuario |
| **Flujo de datos principal** | El usuario vuelve a la pantalla anterior de la aplicación |
| **Flujo de datos excepcional** | La aplicación se cierra o pasa a segundo plano en favor de otra aplicación |
| **Precondición** | Presionar el botón atrás del dispositivo estando en la pantalla inicial |
| **Postcondición** | ninguna |
| **Escenario** | El usuario está en la pantalla de Galería de Bits |

*Tabla 26: Caso de Uso Volver Atrás en la pantalla de Galería de Bits*

# 5.- Diagrama de Clases

El diagrama de Clases es un diagrama UML donde se muestran las Clases que usa la aplicación (con sus atributos y funciones) y las relaciones existentes entre dichas clases.

El diagrama de clases de la aplicación es el siguiente:

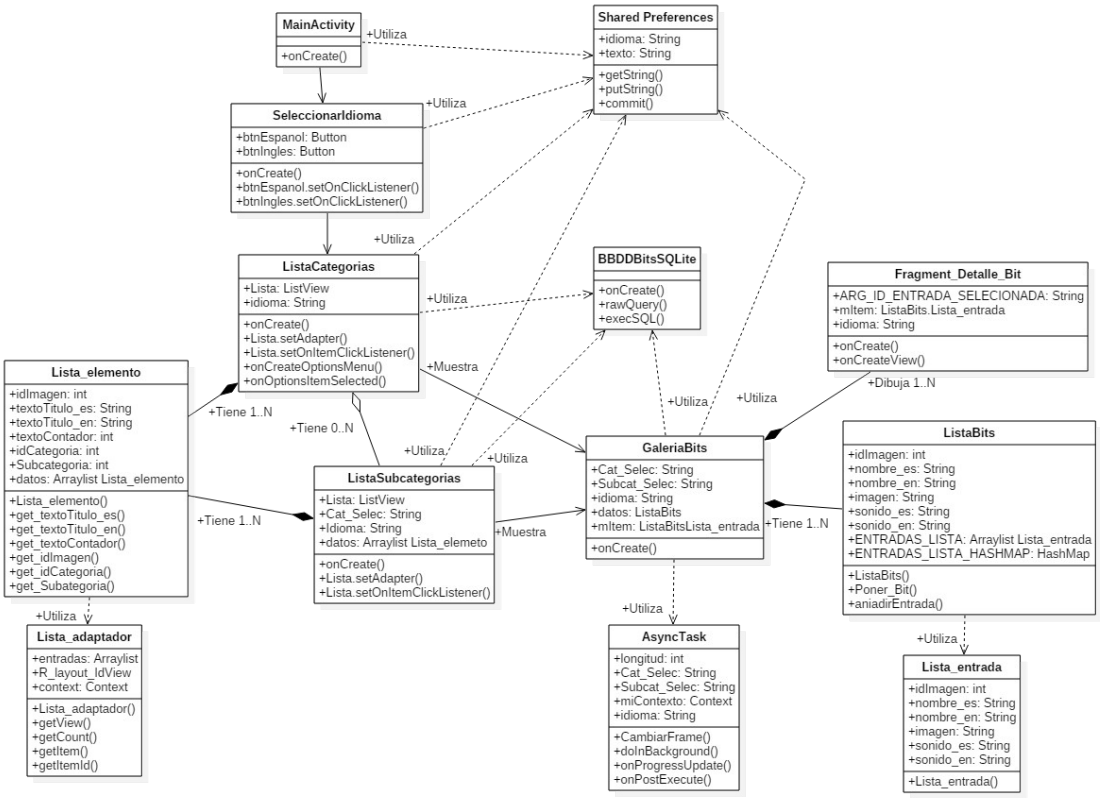


*Ilustración 35: Diagrama de Clases*

El programa se inicia en la clase *MainActivity*, que es representada por una pantalla de bienvenida, al avanzar llegamos a la clase *SeleccionarIdima* donde podemos elegir el idioma de la aplicación y lo guardamos en *SharedPreferences*.

Después de seleccionar el idioma, llegamos a la clase *ListaCategorias*, que es la pantalla de la lista principal de Categorías que se cargan desde la base de datos. Desde aquí mediante el menú de opciones podemos reinicializar los contadores, seleccionar el formato del texto que se representará al mostrar los Bits (guardándose en *SharedPreferences*) o podemos elegir una opción de la lista, que nos mandará a la clase *GaleriaBits* o a la clase *ListaSubcategorias*, si existen subcategorías.

Si llegamos a la clase *ListaSubcategorias*, se nos mostrará una lista con las subcategorías correspondientes, obtenidas de la base de datos, clase *BBDDBitsSQLite*, y al seleccionar alguna llegaremos a la clase *GaleriaBits*.

Una vez que estemos en la clase GaleriaBits, leeremos de la base de datos, clase *BBDDBitsSQLite*, y las preferencias desde *SharedPreferences* para ver en que formato hay que presentar el texto y en qué idioma. Cuando tengamos las preferencias, creamos una lista de bits, clase *Listabits*, y mostramos cada bit en pantalla actualizando el *fragment* de la pantalla, clase *Fragment_Detalle_Bit*. Esta actualización se hace de forma asíncrona con la clase *AsyncTask*.

# 6.- Diagramas de Secuencia

Los diagramas de secuencia son diagramas UML que muestran cómo se relacionan en interactúan las clases y elementos de una aplicación a lo largo del tiempo.

Para mostrar los diagramas de secuencia y facilitar la legibilidad, voy a hacer varios diagramas en función de la acción que se vaya a llevar a cabo. Habrá un primer diagrama que representará el inicio de la aplicación hasta que se llega a la pantalla del menú con la lista de categorías y los siguientes partirán de este punto para mostrar el resto de acciones que serán representar una galería de bits desde la lista de categorías, representar una lista de bits pasando por la lista de subcategorías, reinicializar los contadores y seleccionar una opción de configuración del texto que se muestra en la pantalla de cada Bit.

1. Diagrama desde el inicio de la aplicación hasta la pantalla de la lista de categorías:
2. Diagrama de representar una galería de bits desde la lista de categorías:
3. Diagrama de representar una galería de bits desde la lista de subcategorías:
4. Diagrama de la opción reinicializar contadores:
5. Diagrama de la opción de configuración del formato del texto en la pantalla de los bits:

Ahora paso a mostrar los diagramas:

1. **Diagrama desde el inicio de la aplicación hasta la pantalla de la lista de categorías**:

Como ya comenté, este diagrama muestra la secuencia desde que se inicia la aplicación hasta que el usuario llega a la pantalla con la lista de categorías. Pasa por la pantalla de presentación y por la de selección de idioma.

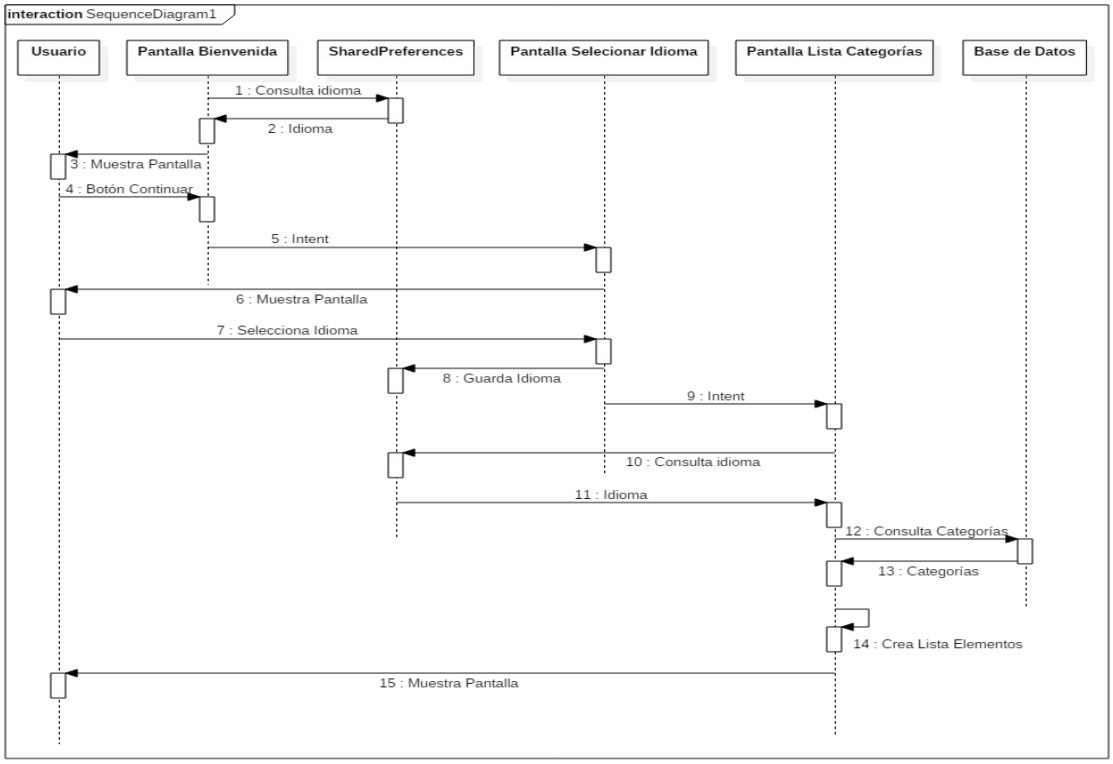


*Ilustración 36: Diagrama de Secuencia Inicial*

Como podemos observar, lo primero que se hace es consultar el idioma que está almacenado en *SharedPreferences* para definirlo como idioma de la aplicación. Se muestra la pantalla de Bienvenida, el usuario presiona el botón continuar y se carga la pantalla de selección de idioma, habrá que escoger uno mediante su botón correspondiente, este idioma se guardará en *SharedPreferences* y se pasará a cargar la lista de categorías, para ello se consulta la base de datos y el idioma guardado, se crea la lista con las categorías en el idioma elegido y se muestra por pantalla al usuario para que escoja una opción.

2. **Diagrama de representar una galería de bits desde la lista de categorías**:

Este diagrama empezará cuando se muestra al usuario la pantalla con la lista de categorías y el usuario selecciona una categoría que no está compuesta por subcategorías, por lo que se representará la galería de bits.

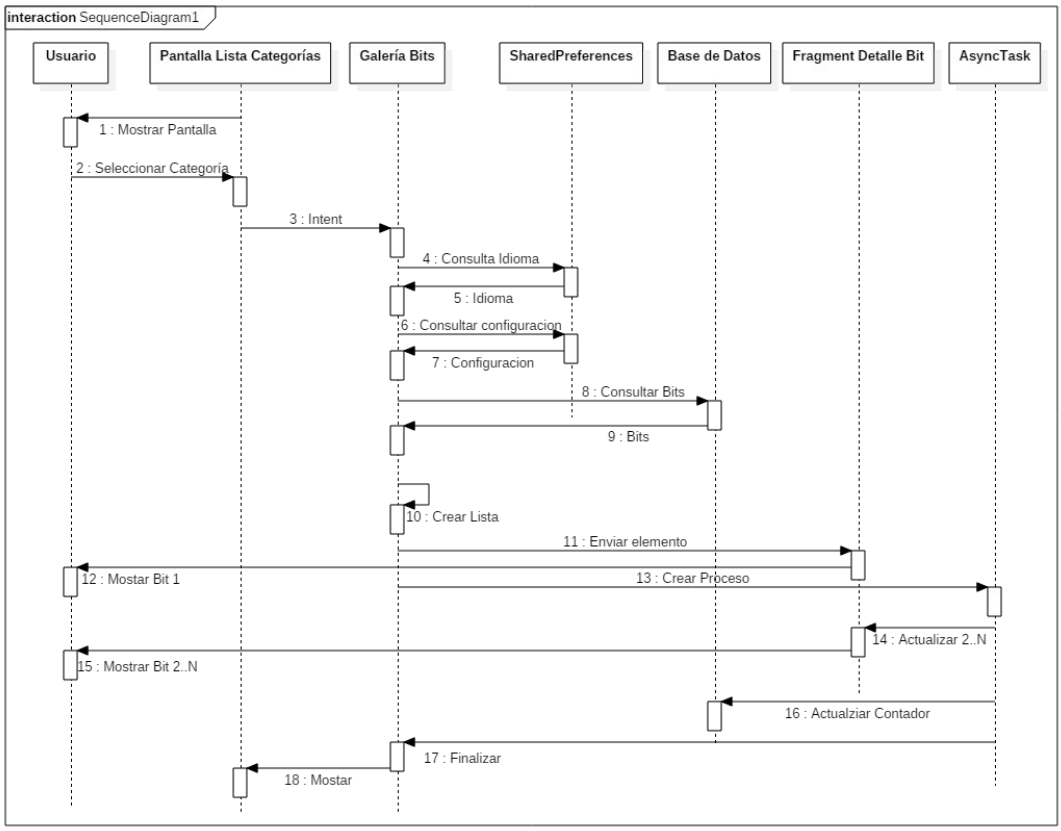


*Ilustración 37: Diagrama de Secuencia Galería de Bits desde Lista de Categorías*

El usuario selecciona la categoría deseada, entonces se carga la galería de bits correspondiente. Se leen las variables idioma y configuración del texto almacenadas en *SharedPreferences* y los bits correspondientes de la base de datos. Se crea la lista de Bits y se muestra el primero por pantalla, conforme a la configuración. Después se crea un hilo asíncrono que controlará la actualización de los *fragments* de los bits y la reproducción del sonido. Cuando finaliza la lista, se actualiza el contador de la base de datos y se termina el proceso de la Galería de bits, volviendo a la pantalla anterior de la lista de categorías.

3. **Diagrama de representar una galería de bits desde la lista de subcategoría**s:

Este diagrama empezará cuando se muestra al usuario la pantalla con la lista de categorías y el usuario selecciona una categoría que está compuesta por subcategorías. Entonces se mostrará la lista de subcategorías, el usuario elegirá una subcategoría y entonces se representará la galería de bits.

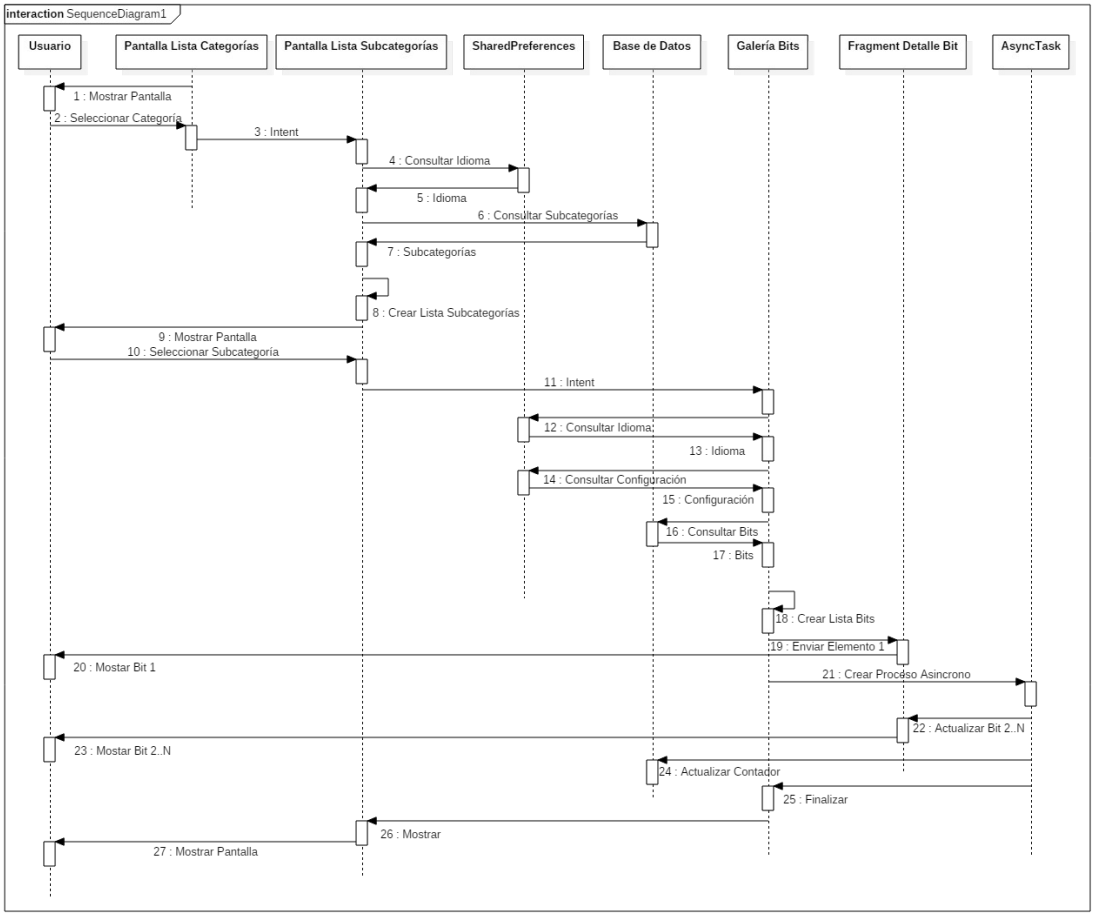


*Ilustración 38: Diagrama de Secuencia mostrar Galería de Bits desde Lista de Categorías con Subcategorías*

El usuario selecciona la categoría que quiere desde la lista. Como tiene varias subcategorías, se solicita la pantalla de lista de subcategorías, para ello se lee la variable idioma desde *SharedPreferences* y la lista de subcategorías desde la base de datos. Entonces se muestra la pantalla al usuario, que escoge una de las subcategorías. Se solicita la galería de bits correspondiente a la elección. Para ello, se leen las variables idioma y configuración del texto desde *SharedPreferences* y la lista de Bits desde la base de datos. Se crea la lista de Bits y se muestra el primer bit de la lista en el *fragment*. Después se crea un hilo secundario de procesamiento asíncrono que va actualizando el *fragment* con los nuevos bits y reproduciendo los sonidos. Al finalizar la lista de bits, se actualiza el contador de la subcategoría en la base de datos y se finaliza el hilo de la galería de bits por lo que se vuelve a mostrar la pantalla anterior de la lista de subcategorías.

4. **Diagrama de la opción reinicializar contadores**:

Este diagrama empezará cuando se muestra al usuario la pantalla con la lista de categorías y el usuario presiona el botón opciones del dispositivo, por lo que se muestra el menú de opciones y el usuario escogerá la opción reinicializar contadores para poner a 0 todos los contadores de las categorías y subcategorías que nos indican las veces que se ha visualizado cada una. Entonces se volverá a la pantalla de la lista de categorías.

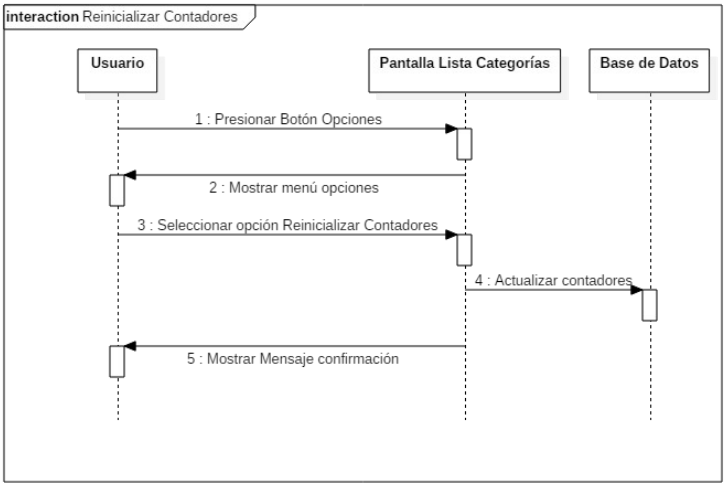


*Ilustración 39: Diagrama de Secuencia de Reinicializar Contadores*

Desde la pantalla de la lista de categorías, el usuario presiona el botón de opciones del dispositivo, entonces se muestra un menú de opciones. El usuario escoge la opción de reinicializar contadores, se actualizan los contadores en la base de datos poniendo su valor a cero y se muestra un mensaje emergente de confirmación. Cuando desaparece el mensaje, el usuario sigue en la pantalla de selección de categorías y puede realizar la acción que desee.

5. **Diagrama de la opción de configuración del formato del texto en la pantalla de los bits**:

Este diagrama empezará cuando se muestra al usuario la pantalla con la lista de categorías y el usuario presiona el botón opciones del dispositivo, por lo que se muestra el menú de opciones y el usuario escogerá la opción de configurar el texto que aparece en la galería de bits con cada bit. Aparecerá un menú donde se podrá escoger entre tres opciones para los textos, "*Sin texto*" para que no aparezca texto, "*Mayúsculas*" para que el texto aparezca en mayúsculas o "*Minúsculas*" para que el texto aparezca en minúsculas. Entonces se volverá a la pantalla de la lista de categorías.

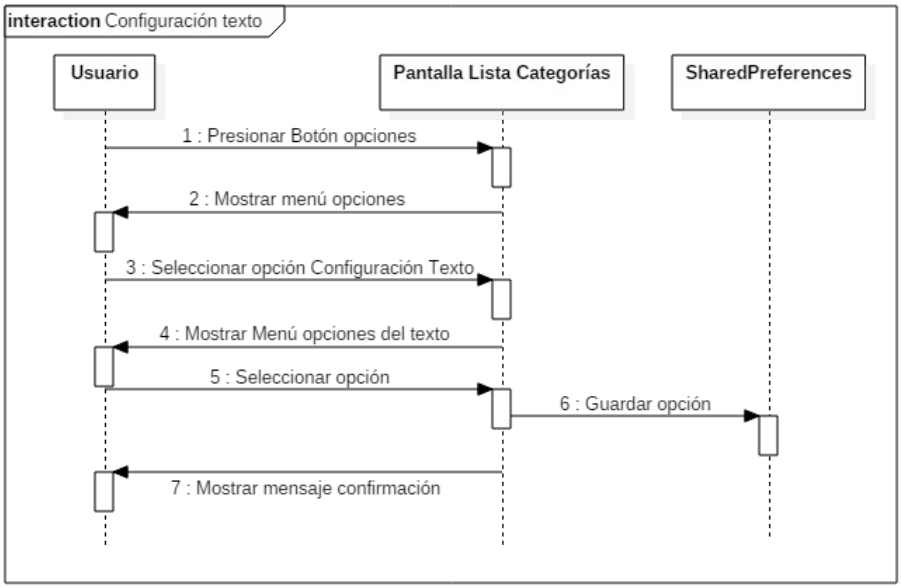


*Ilustración 40: Diagrama de Secuencia de elegir configuración del formato del texto en los Bits*

Desde la pantalla de la lista de categorías, el usuario presiona el botón de opciones del dispositivo, entonces se muestra un menú de opciones. El usuario escoge la opción de configurar el texto de los bits, se le mostrará un nuevo menú con las opciones disponibles, el usuario seleccionará una, se guardará su selección en *SharedPreferences* y se muestra un mensaje emergente de confirmación. Cuando desaparece el mensaje, el usuario sigue en la pantalla de selección de categorías y puede realizar la acción que desee.

# 7.- Diagrama de Actividad

Los diagramas de actividad son diagramas UML que nos muestran cómo se produce el flujo de trabajo a través de acciones.

El diagrama de actividad es:

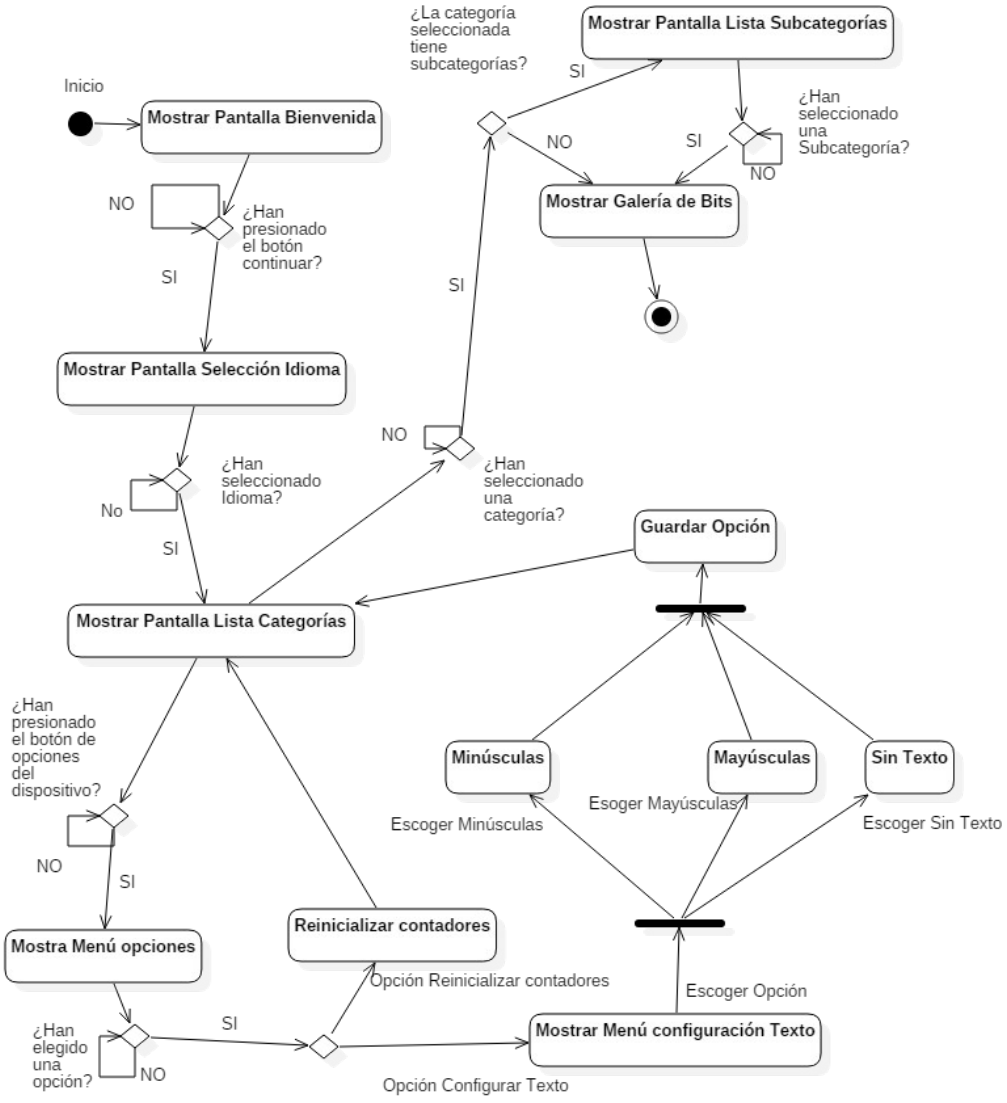


*Ilustración 41: Diagrama de Actividad*

# 8.- Diseño de la Base de Datos

La aplicación para funcionar utiliza una base de datos donde están almacenadas las categorías, las subcategorías y los bits. Cada uno de estos elementos está compuesto por una serie de atributos que se almacenan y que son necesarios para el funcionamiento de la aplicación.

La forma de representar una base de datos es mediante esquemas. Hay dos tipos de esquemas fundamentales, el de Entidad-Relación y el Relacional.

El modelo Entidad-Relación (ER) se basa en representar los objetos como Entidades, con unos atributos y las Relaciones que existen entre dichas Entidades.

El modelo Relacional se basa en que todos los datos son almacenados en relaciones, el esquema que lo representa es un diagrama en forma de una estructura de tablas y la información que contienen dichas tablas en forma de campos o atributos.

Android para las bases de datos usa el motor SQLite que no requiere de servidor y es muy rápido sin hacer un uso excesivo de recursos. Para trabajar con él se utiliza el lenguaje SQL.

## *8.1.- Esquema del modelo Entidad-Relación*

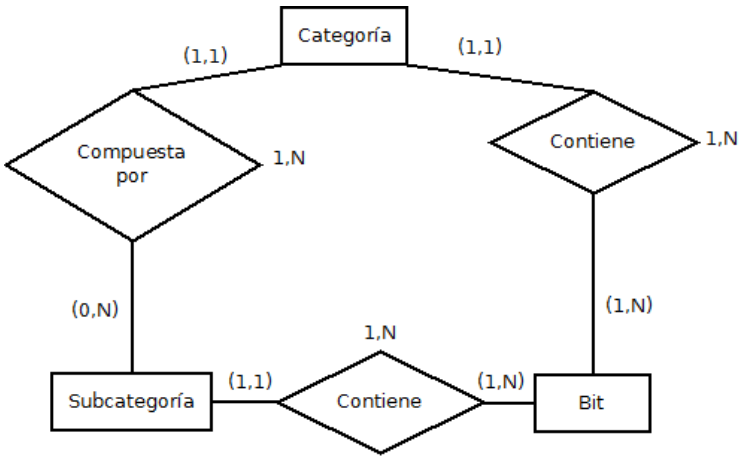


*Ilustración 42: Diagrama Entidad-Relación*

## 8.2.- *Esquema del modelo Relacional*

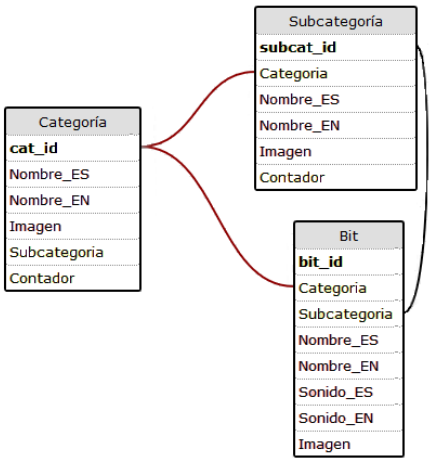


*Ilustración 43: Diagrama Relacional*

## 8.3.- *Diccionario de Datos*

Independientemente del diagrama con el que dibujemos y representemos la base de datos, los atributos de las entidades son los mismos, y a continuación paso a detallarlos:

1. **Entidad Categoría**:
   - cat_id:
     - Tipo: Entero Autoincremental.
     - Descripción: Identificador de categoría. Define de forma única una categoría.
     - Carácter: Obligatorio. Es la clave primaria de la tabla.
   - Nombre_ES:
     - Tipo: Texto.
     - Descripción: Define el nombre de la categoría en español.
     - Carácter: Obligatorio.
   - Nombre_EN:
     - Tipo: Texto.
     - Descripción: Define el nombre de la categoría en inglés.
     - Carácter: Obligatorio.

- Imagen:
  - Tipo: Texto.
  - Descripción: Define el nombre de la imagen de la categoría.
  - Carácter: Obligatorio.
- Subcategoría:
  - Tipo: Entero.
  - Descripción: Es usado como boolean, indica si la categoría tiene subcategorías o no.
  - Carácter: Obligatorio.
- Contador:
  - Tipo: Entero.
  - Descripción: Identificador de categoría. Define de forma única una categoría.
  - Carácter: Obligatorio.

2. **Entidad Subcategoría**:
   - subcat_id:
     - Tipo: Entero Autoincremental.
     - Descripción: Identificador de subcategoría. Define de forma única una subcategoría.
     - Carácter: Obligatorio. Es la clave primaria de la tabla.
   - Categoría:
     - Tipo: Entero.
     - Descripción: Clave foránea que nos permite saber a qué categoría pertenece cada subcategoría. Hace referencia al atributo *cat_id* de la entidad Categoría.
     - Carácter: Obligatorio.
   - Nombre_ES:
     - Tipo: Texto.
     - Descripción: Define el nombre de la subcategoría en español.
     - Carácter: Obligatorio.
   - Nombre_EN:
     - Tipo: Texto.
     - Descripción: Define el nombre de la subcategoría en inglés.
     - Carácter: Obligatorio.

- Imagen:
  - Tipo: Texto.
  - Descripción: Define el nombre de la imagen de la subcategoría.
  - Carácter: Obligatorio.
- Contador:
  - Tipo: Entero.
  - Descripción: Identificador de categoría. Define de forma única una categoría.
  - Carácter: Obligatorio.

3. **Entidad Bit**:
   - bit_id:
     - Tipo: Entero Autoincremental.
     - Descripción: Identificador de bit. Define de forma única un bit.
     - Carácter: Obligatorio. Es la clave primaria de la tabla.
   - Categoría:
     - Tipo: Entero.
     - Descripción: Clave foránea que nos permite saber a qué categoría pertenece cada bit. Hace referencia al atributo *cat_id* de la entidad Categoría.
     - Carácter: Obligatorio.
   - Subcategoría:
     - Tipo: Entero.
     - Descripción: Clave foránea que nos permite saber a qué subcategoría pertenece cada bit. Hace referencia al atributo *subcat_id* de la entidad Subcategoría.
     - Carácter: Obligatorio.
   - Nombre_ES:
     - Tipo: Texto.
     - Descripción: Define el nombre del bit en español.
     - Carácter: Obligatorio.
   - Nombre_EN:
     - Tipo: Texto.
     - Descripción: Define el nombre del bit en inglés.
     - Carácter: Obligatorio.
   - Sonido_ES:

    - Tipo: Texto.
    - Descripción: Define el nombre del sonido perteneciente al bit en español.
    - Carácter: Obligatorio.
- Sonido_EN:
    - Tipo: Texto.
    - Descripción: Define el nombre del sonido perteneciente al bit en inglés.
    - Carácter: Obligatorio.
- Imagen:
    - Tipo: Texto.
    - Descripción: Define el nombre de la imagen del bit.
    - Carácter: Obligatorio.

# 9.- Usabilidad

La usabilidad de una aplicación es un punto clave para que tenga éxito. A la hora de interactuar con una aplicación, el usuario busca una forma de uso sencilla, intuitiva y que no requiera de mucho esfuerzo pensando que hay que hacer o que conlleve un periodo largo de aprendizaje. Todo esto hay que tenerlo muy en cuenta a la hora de diseñar y desarrollar la aplicación y las interfaces y pantallas de comunicación con el usuario, es decir tanto la estructura interna como la parte gráfica y visual de la aplicación. Por eso que realizarse las siguientes preguntas:

- **¿Qué quiero que haga la aplicación?** El objetivo de la aplicación es entretener y facilitar el aprendizaje de los niños de forma amena. El resto de objetivos están marcados por los requisitos generales, funcionales y no funcionales.
- **¿A quién va dirigida la aplicación?** Sirve para determinar a quién va dirigida la aplicación, para ello se realiza un análisis de mercado y de usuarios. En este caso en concreto, la aplicación va destinada a un público infantil.
- **¿Cómo funcionará la aplicación?** En función del uso que se dé a las aplicaciones, se diseñarán de una forma u otra, no es lo mismo que una aplicación la use un usuario en su dispositivo que varios usuarios simultáneos haciendo uso de un servidor, si se usa de forma ocasional o muy seguida y es necesario automatizar ciertos procesos, etc. En este caso nuestra aplicación será usa en un único dispositivo que generará toda la aplicación en dicho dispositivo de forma estática y amigable.

# 10.- Batería de Pruebas

Una vez la parte del desarrollo de la aplicación ha concluido, llega el momento de comprobar el correcto funcionamiento de la aplicación. Para este objetivo, necesitamos diseñar y llevar a cabo una batería de pruebas para analizar y comprobar que la aplicación una vez terminada cumple con todas las funciones y requisitos necesarios de forma satisfactoria. El fin último de la batería de pruebas es comprobar si se produce algún error para corregirlo y que la aplicación llegue al usuario final sin fallos y funcionando perfectamente cumpliendo todos los objetivos y requisitos.

Las pruebas realizadas han sido las siguientes:

- **Iniciar la aplicación**: Comprobamos que una vez instalada en un dispositivo real, al ejecutarla se abre y llegamos a la pantalla da inicio.
- **Comprobar categorías**: En la pantalla de Categorías, comprobamos que se muestren todas las que están en la base de datos y que las que no tienen subcategorías que aparezcan con el contador a la derecha.
- **Comprobar Subcategorías**: Comprobamos una a una todas las categorías que tienen subcategorías para ver que están correctamente listadas, es decir, en cada categoría aparecen todas sus subcategorías que hay en la base de datos y ninguna otra.
- **Comprobar Galerías de Bits**: Acceder a todas las galerías de bits, seleccionando su categoría y subcategoría si es necesario y ver que la representación en pantalla es correcta, que no se carga rápidamente y sin problemas la galería de bits, que se muestran todos los bits de su lista y ninguno otro, que se muestra la imagen correspondiente, el texto correspondiente y se escucha el sonido correspondiente. También se comprueba que se incremente el contador y que la transición de un bit a otro se hace forma automática con el tiempo programado y de manera fluida.
- **Comprobar que se reinicializan bien los contadores**: Desde la pantalla de selección de categoría, apretar el botón de opciones del dispositivo, comprobar que se abre el menú de opciones correspondiente, que esta accesible la opción de “Reinicializar contadores” y presionar sobre dicha opción para que se reinicializan, a continuación, ver la lista de categorías y las subcategorías y comprobar que todos los contadores están a cero.
- **Comprobar la configuración de texto**: Desde la pantalla de selección de categoría, apretar el botón de opciones del dispositivo, comprobar que se abre el menú de opciones correspondiente, que esta accesible la opción de “Configurar texto” y que al presionar en esta opción aparece un submenú para poder elegir entre “Mayúsculas”, “Minúsculas” y “Sin texto”.
- **Comprobar la configuración de texto Mayúsculas**: Desde la pantalla de selección de categoría, apretar el botón de opciones del dispositivo, comprobar que se abre el menú de opciones correspondiente, que esta accesible la opción de “Configurar texto”, que al

presionar en esta opción aparece un submenú para poder elegir entre “Mayúsculas”, “Minúsculas” y “Sin texto”, seleccionamos “Mayúsculas” y seleccionamos una categoría que no tenga subcategorías para acceder a un galería de bits, entonces comprobamos que el texto de cada bit aparece en mayúsculas. Salimos de la aplicación, volvemos a entrar y accedemos de nuevo a una galería de bits para comprobar que el texto sigue apareciendo en mayúsculas y la opción está correctamente guardada.

- **Comprobar la configuración de texto Minúsculas**: Desde la pantalla de selección de categoría, apretar el botón de opciones del dispositivo, comprobar que se abre el menú de opciones correspondiente, que esta accesible la opción de “Configurar texto”, que al presionar en esta opción aparece un submenú para poder elegir entre “Mayúsculas”, “Minúsculas” y “Sin texto”, seleccionamos “Minúsculas” y seleccionamos una categoría que no tenga subcategorías para acceder a un galería de bits, entonces comprobamos que el texto de cada bit aparece en minúsculas. Salimos de la aplicación, volvemos a entrar y accedemos de nuevo a una galería de bits para comprobar que el texto sigue apareciendo en minúsculas y la opción está correctamente guardada.
- **Comprobar la configuración de texto Sin texto**: Desde la pantalla de selección de categoría, apretar el botón de opciones del dispositivo, comprobar que se abre el menú de opciones correspondiente, que esta accesible la opción de “Configurar texto”, que al presionar en esta opción aparece un submenú para poder elegir entre “Mayúsculas”, “Minúsculas” y “Sin texto”, seleccionamos “Sin texto” y seleccionamos una categoría que no tenga subcategorías para acceder a un galería de bits, entonces comprobamos que el texto que normalmente aparece de cada bit en esta ocasión no aparece. Salimos de la aplicación, volvemos a entrar y accedemos de nuevo a una galería de bits para comprobar que sigue sin aparecer el texto y la opción está correctamente guardada.
- **Seleccionar Idioma**: Seleccionamos un Idioma y comprobamos que después de seleccionarlo, los textos de los botones, menús, listas, mensajes, galería de bits y los sonidos de la galería de Bits están en el idioma seleccionado. Cerramos la aplicación, la volvemos a abrir y vemos que sigue seleccionado el mismo idioma y todos los mensajes se muestran en el idioma que está seleccionado.
- **Cambiar idioma seleccionado**: Teniendo seleccionado un idioma, seleccionamos el otro y comprobamos de nuevo que todas las pantallas, menús, listas, mensajes, sonidos, etc. están en el idioma seleccionado. Cerramos la aplicación, la volvemos a abrir y comprobamos que siga seleccionado el mismo idioma y todo siga mostrándose bien.
- **Rotar pantalla**: Desde cada una de las pantallas de la aplicación, incluida la galería de bits rotamos la pantalla a ver si todo sigue su correcto funcionamiento y que las opciones siguen guardadas y configuradas, probamos a desplazarnos por las pantallas para ver que todo está correcto, posteriormente volvemos a poner la pantalla en la posición original y volvemos a comprobar que todo está correcto, que se visualiza bien y que se siguen aplicando las opciones guardadas.
- **Volver Atrás**: Comprobamos la funcionalidad del botón volver atrás del dispositivo para ver que desde cada una de las pantallas volvemos a la anterior al pulsarlo.

# SECCIÓN III: MANUAL DE USUARIO

Este aparatado de la documentación está dedicado a un manual de usuario, para que todo aquel que lo desee, pueda consultar información sobre la aplicación.

# 1.- Descripción del sistema

Bienvenidos a la ayuda de la aplicación Bits de Inteligencia, aquí podrán encontrar la información necesaria para resolver cualquier duda que tengan con la aplicación y su funcionamiento.

La aplicación Bits de Inteligencia es una aplicación desarrollada para Android en su versión 4.0 o posterior, por lo que por favor, compruebe que versión de Android tiene su dispositivo antes de intentar instalar la aplicación.

La aplicación está diseñada para que sea de uso fácil, sencillo, intuitivo y amigable para que cualquier persona pueda utilizarla sin conocimientos previos pueda utilizarla, se ha hecho hincapié en la facilidad de uso, ya que esta aplicación está destinada a un público infantil.

La aplicación desarrolla el método didáctico de los Bits de Inteligencia, que vamos a desarrollar un poco a continuación.

Este método va dirigido a niños/as de entre 0-6 años para que puedan mejoran la atención, facilitar la concentración y que desarrollen y estimulen el cerebro, la memoria y el aprendizaje. Con esto, desarrollan la inteligencia, facultad de actuar eficazmente ante situaciones nuevas. Este comportamiento se da gracias a que el cerebro es capaz de relacionar conocimientos nuevos con datos de experiencias pasadas (sinapsis).

Creado por el médico estadounidense Glenn Doman, con el objetivo de estimular al niño para que aprenda.

Su metodología se basa en mostrar información visual y auditiva de forma escueta y rápida, mediante tarjetas de información. Se ha comprobado que los estímulos cortos son más eficaces que los largos, por lo que los bits se mostrarán rápidos, repetidamente en varias sesiones cortas y con gran alegría para atraer su atención y motivación.

El método se trabaja a través de categorías que agrupan un número de imágenes relacionas por áreas de conocimiento.

Los niños aprende de este modo, sin darse apenas cuenta, nuevos términos y significados, clasificados y estructurados, que servirán para desarrollar y ampliar el lenguaje, el vocabulario y la memoria.

Es un método potencial que les favorecerá de forma significativa en etapas educativas posteriores.

En el ámbito pedagógico, entendemos **BIT** como cualquier dato simple que pueda almacenar el cerebro y que llegue a través de los sentidos. Los bits de inteligencia son unidades de información que son presentadas a los niños de forma breve, con lo que se consigue captar su atención. Los bits son estímulos.

Los bits son unidades de información visuales, por medio de imágenes claras, precisas y bien definidas o una fotografía de buena calidad (estímulo visual) unidas a información auditiva, que consiste en enunciar en voz alta lo que se representa en la información visual (estímulo auditivo). Un bit de inteligencia es un bit de información.

Este método no pretende enseñar directamente, sino estimular las áreas cerebrales, especialmente de la vista y el oído.

Como vemos este método es muy estimulante y útil para ayudar al desarrollo de los niños por lo que esta aplicación sirve como apoyo en la gratificante tarea de estimular el aprendizaje de los niños y que además podrán usar ellos mismos.

# 2.- Instalación de la aplicación

Para instalar la aplicación, necesitamos el archivo "*bitsinteligencia.apk*" en nuestro dispositivo. Una vez que lo tengamos presionamos sobre el para que se instale.


*Ilustración 44: Instalación de la aplicación. Paso1*

Nos aparecerá una pantalla que nos indicará que no necesita ningún permiso especial para su funcionamiento y tendremos que dar a "Instalar".

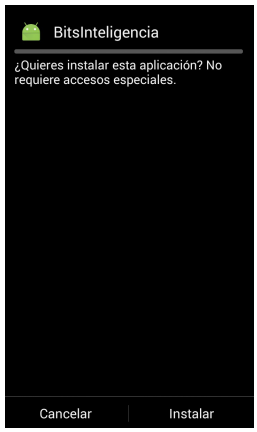


*Ilustración 45: Instalación de la aplicación. Paso2*

Una vez que se instale la aplicación, nos mostrará una pantalla de que se ha instalado correctamente, desde donde podremos abrir la aplicación y nos habrá creado un icono para ejecutarla, buscamos dicho icono, presionamos sobre él y la aplicación empezará a funcionar.

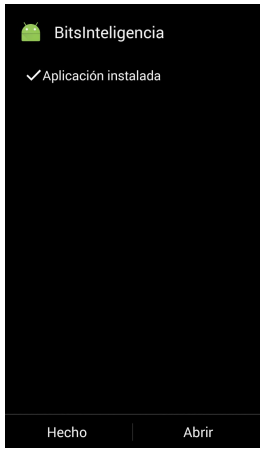


*Ilustración 46: Instalación de la aplicación. Paso3*

# 3.- Uso de la aplicación

Una vez instalada la aplicación, presionamos sobre el icono que nos habrá creado y la aplicación se abrirá.

Tendremos una primera pantalla de bienvenida, donde se nos mostrará un saludo y tendremos que presionar en el botón continuar.

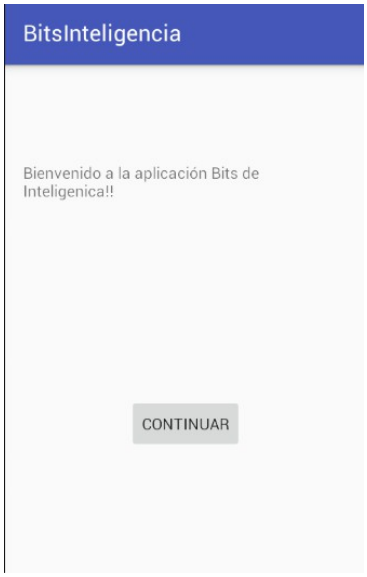


*Ilustración 47: Manual de usuario. Pantalla de Bienvenida*

Llegaremos a otra pantalla, donde se nos pedirá que seleccionemos el idioma en el cual queremos que aparezcan las listas de categorías y los bits. Presionamos sobre el botón del idioma que deseemos para llegar a la siguiente pantalla.


*Ilustración 48: Manual de usuario. Pantalla de selección de idioma*

Llegaremos a una pantalla donde se nos mostrará la lista de categorías que podemos elegir.

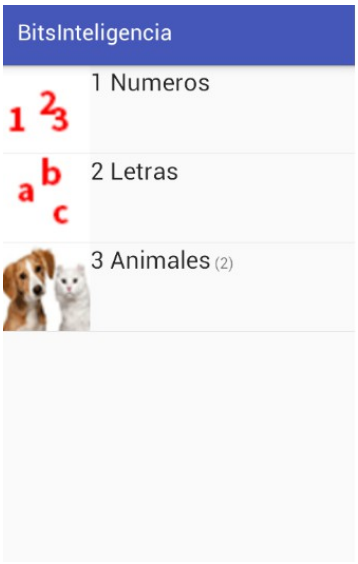


*Ilustración 49: Manual de usuario. Pantalla de selección de categorías*

En esta lista hay dos tipos de elementos diferentes, las categorías que tienen subcategorías y las categorías que no tienen subcategorías. Se diferencian en que las categorías que no tienen subcategorías, a la derecha de su nombre aparece un número entre paréntesis que nos indica el número de veces que se ha visualizado la galería de bits correspondiente a dicha categoría.

- Categorías con subcategorías:

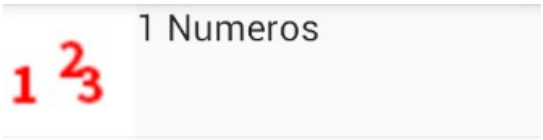


*Ilustración 50: Manual de usuario. Categoría con Subcategorías*

- Categorías sin subcategorías:

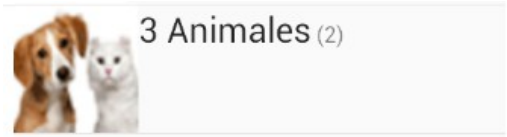


*Ilustración 51: Manual de usuario. Categoría sin Subcategorías*

Si elegimos una categoría que no tiene subcategorías, se nos mostrará la galería de bits correspondiente. Cada pantalla de bit estará compuesta por una imagen, un texto que indica el bit y lo que aparece en la imagen y un sonido que nos dice el nombre del bit. Al finalizar la galería, se incrementará en uno el contador que muestra la cantidad de veces que se ha reproducido la galería y se volverá a la pantalla inmediatamente anterior.


*Ilustración 52: Manual de usuario. Pantalla de representación de un Bit*

Si elegimos una categoría con subcategorías, llegaremos a otra pantalla donde aparece la lista de subcategorías, ahora todos los elementos de la lista tienen a su derecha el número que indica la cantidad de veces que se han reproducido. Al seleccionar una subcategoría, se nos mostrará la galería de bits correspondiente, igual que las galerías que se muestran desde la pantalla anterior con la lista de categorías.


*Ilustración 53: Manual de usuario. Pantalla de selección de subcategorías*

## *3.1.- Configuración*

La aplicación tiene unas pequeñas opciones de configuración. Estas opciones consisten en poner a cero los contadores que nos indican la cantidad de veces que se han visualizado las diferentes galerías de bits y poder elegir la forma del texto que aparece en la pantalla de los bits en la galería.

Estás opciones se encuentran disponibles solo en la pantalla de selección de categorías, y para acceder a ellas hay que presionar el botón de opciones del dispositivo.

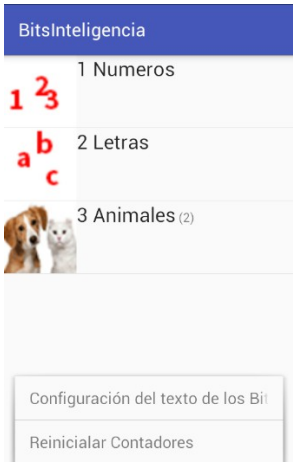


*Ilustración 54: Manual de usuario. Menú de opciones*

Vamos a detallar cada opción.

### *3.1.1.- Reinicializar contadores*

Con el menú representado en pantalla, elegimos la opción de “reinicializar contadores”. Se pondrán todos los contadores a cero y nos aparecerá un mensaje confirmándolo y volveremos a la pantalla de la lista de categorías.

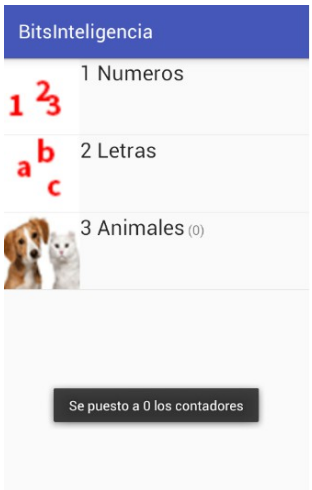


*Ilustración 55: Manual de usuario. Reinicializar contadores*

### *3.1.2.- Configuración del texto de los bits*

Con el menú representado en pantalla, elegimos la opción de “Configuración del texto de los Bits”. Se nos mostrará otro menú con las opciones a elegir.

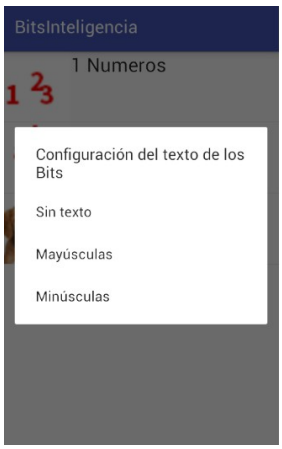


*Ilustración 56: Manual de usuario. Menú opciones del texto de los Bits*

Seleccionamos la opción que deseemos y entonces aparecerá un mensaje confirmándolo y volveremos a la pantalla de la lista de categorías.

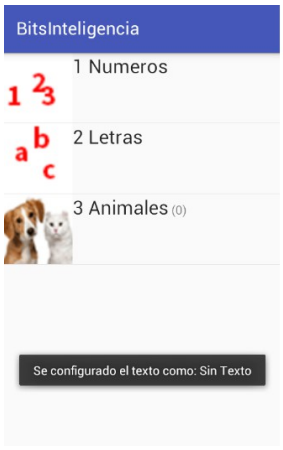


*Ilustración 57: Manual de usuario. Confirmación de la opción del texto de los Bits*

Una vez guardada la opción, al representar una galería de Bits, el texto aparecerá con la opción seleccionada. En este caso del ejemplo, sin texto.


*Ilustración 58: Manual de usuario. Comprobación de la opción del texto de los Bits*

Aquí concluye el manual de usuario.

# CONCLUSIONES

A la hora de entregar este Trabajo de Fin de Grado estoy contento de haber conseguido tener la aplicación operativa después de todo el esfuerzo dedicado y de haber superado los múltiples problemas encontrados por el camino. Aunque no está terminada de la forma en que me hubiera gustado, ya que estéticamente no es llamativa ni muy visual y en lo que respecta a los Bits de Inteligencia incluidos, no están todos los que me hubiera gustado incluir y para los que busqué y preparé imágenes y sonidos al principio. Esto es debido en primer lugar a la falta de tiempo disponible que he tenido para realizar el proyecto, que tenía que compatibilizar con el trabajo y a los problemas mencionados que me llevo bastante tiempo solucionarlos.

A la hora de solucionar los problemas y probar que la aplicación funcionaba

La idea inicial era que para cuando entregara el proyecto y lo defendiera, la aplicación estuviera disponible en *Google Play Store*, pero aunque está operativa no está concluida como tenía pensado que estuviera, de momento no está en *Google Play Stor*e y no lo estará hasta que esté terminada al 100% de acuerdo con mi criterio personal.

Los problemas no ha sido fáciles de resolver ya que tanto Java como Android eran completamente nuevos para mí y no ha resultado sencillo adaptarse a ellos, ya que la forma que hay que desarrollar en Android está basada en sus librerías y funciones y hay que entender muy bien su funcionamiento para elegir las librerías y funciones adecuadas y que estén disponibles en la versión de Android escogida. Otro inconveniente que tiene esto es que cada vez que se produce una actualización de algún tipo en Android estudio, es probable que aparezcan fallos en el desarrollo que haya que buscar, analizar y resolver.

El principal problema que he tenido ha estado relacionado con los *fragments* utilizados para representar la galería de bits, ya que inicialmente aparecieron en la versión 3 de Android y se incorporaron librerías que los soportaran en versiones anteriores. Hice uso de dichas librerías, ya que estaba usando la versión 2.0 de Android, pero dichas librerías tenían fallos y problemas y por lo tanto a la hora de representar los bits y hacer la transición de un bit a otro, la pantalla se quedaba negra hasta llegar al último bit a representar. Después de mucho tiempo tratando de resolver este problema, opté por cambiar la versión de Android a la 4.0 para ya tener soporte nativo y no librerías adaptadas y poder usar los *fragments* de manera correcta. Esto supuso un gran esfuerzo ya que hay determinadas librerías que había que cambiar y tuve que repasar el proyecto completo para que funcionara correctamente. Con este cambio se solucionaron parte de los problemas de los cambios en la representación de las pantallas, pero tenía problemas con el sincronismo entre el cambio de pantallas y la reproducción de los sonidos en cada pantalla. Para solucionar esto, opte por incluir un procesamiento asíncrono creando otro hilo de ejecución mediante *AsyncTask*.

Aunque estoy contento y satisfecho con el trabajo realizado hasta este punto, aún queda trabajo por realizar para que la aplicación sea estéticamente impactante y llamativa para causar sensación y ser atrayente, además de completar la información de Bits a utilizar. Por ejemplo, la categoría Animales que está presente, inicialmente iba a estar divida en subcategorías de Animales Domésticos, Animales de Granja y Animales Salvajes, iba a haber más categorías como Colores, Ropa, Alimentos, Material Escolar, Banderas... que iré incluyendo en la aplicación según tenga tiempo disponible, para al final hacerla pública en la *Google Play Store*.

He aprendido mucho sobre Android y tengo muchas ganas de seguir trabajando con ello y crear nuevas aplicaciones que hagan uso de más funciones y del potencial de esta plataforma, que es muy elevado y con potencial número de clientes impresionante en todo el mundo.

# BIBLIOGRAFÍA

En esta sección voy a enumerar las diferentes fuentes que he consultado para la realización del presente Trabajo de Fin de Grado, tanto para adquirir conocimientos como para resolver dudas o problemas o para buscar los diferentes materiales didácticos necesarios como imágenes y sonidos. Los voy a dividir en libros en papel y direcciones de Internet.

# 1.- Libros


- El gran libro de Android 3ª Edición
  - Autor: Jesús Tomas Gironés
  - Editorial: Marcombo
- Android desarrollo de aplicaciones ganadoras
  - Autor: Wei-Meng Lee
  - Editorial: Anaya Multimedia
- El gran libro de Android Avanzado
  - Autores: Jesús Tomas, Vicente Carbonell, Miguel García, Jordi Bataller, Daniel Ferri y Carsten Vogt
  - Editorial: Marcombo
- El gran libro de programación avanzada con Android
  - Autor: José Enrique Amaro Soriano
  - Editorial: Marcombo


# 2.- Direcciones de Internet


- https://developer.android.com/index.html: Página oficial depara desarrolladores de Android. Contiene extensa documentación sobre desarrollo, funciones, librerías, etc.
- http://www.androidcurso.com: Curso online de programación en Android ofrecido por la Universidad de Valencia.
- http://www.sgoliver.net: Blog sobre Android con información, guías y consejos sobre desarrollo.

- http://jarroba.com: Dirección web sobre diferentes temas de programación, desarrollo y tecnologías. Entre su contenido se encuentran múltiples entradas sobre Android con información, guías y consejos sobre desarrollo. Tiene la posibilidad de descargar un libro en PDF.
- http://stackoverflow.com: Dirección web sobre sobre consultas, dudas y problemas a la hora de desarrollar.
- https://es.wikipedia.org: Enciclopedia Online de consulta.
- http://www.wordreference.com: Sitio web que dispone de un sintetizador de voz, recurso usado para los sonidos de los bits.
- http://dictionary.cambridge.org/es: Sitio web que dispone de un sintetizador de voz, recurso usado para los sonidos de los bits.
- http://esl-kids.com: Sitio web que dispone de gran cantidad de imágenes para los bits.
- http://bitsinteligencia.blogspot.com.es: Blog sobre los Bits de Inteligencia, con información del método e imágenes de Bits.
- http://www.disanedu.com: Portal educativo de donde poder obtener información sobre el método didáctico de los Bits de Inteligencia.
- http://losbitsdeinteligenciaanaegido.blogspot.com.es: Blog sobre los Bits de Inteligencia, con información del método e imágenes de Bits.
- http://www.elmundo.es: Dirección web del periódico El Mundo. Consultado para obtener información sobre algunos artículos publicados.
- http://blogthinkbig.com: Blog con artículos de diversa temática, usado para obtener información de ciertas entradas publicadas.
- https://hipertextual.com: Blog con artículos de diversa temática, usado para obtener información de ciertas entradas publicadas.
- http://es.slideshare.net: Sitio web donde se publican todo tipo de trabajos y documentos, utilizado para obtener determinadas informaciones de alguno de los documentos publicados.
- http://www.significados.com: Sitio web dedicados a definiciones y conceptos, utilizado para obtener información y definiciones.
- http://definicion.de: Sitio web dedicados a definiciones y conceptos, utilizado para obtener información y definiciones.
- http://www.crecenegocios.com: Sitio web dedicado a los negocios, utilizado para obtener información y ayuda para el modelo de negocio y plan de negocio.
- http://emprende.unir.net: Sitio web dedicado a emprender y a los negocios, utilizado para obtener información y ayuda para el modelo de negocio y plan de negocio.
- http://www.mediasplash.co: Sitio web con temática empresarial, utilizado para obtener información y ayuda para el modelo de negocio y plan de negocio.
- https://www.lancetalent.com: Blog de temática empresarial, utilizado para obtener información y ayuda para el modelo de negocio y plan de negocio.

- http://lavoztx.com: Sitio web del periódico La Voz de Houston, utilizado para consultar información en determinados artículos publicados.
- https://github.com/ondras/wwwsqldesigner: Sitio web donde descargar wwwsqldesignar, página web de uso en local para diseñar bases de datos.

# AGRADECIMIENTOS

En primer lugar quiero dar las gracias a Lucía, Nora y Oscar mis queridos sobrinos, que son los principales causantes de que se me ocurriera esta idea y quisiera llevarla a cabo.

Quiero dar las gracias también a Luis Ignacio, tutor del proyecto, por su apoyo, ánimo y ayuda para que siguiera adelante con el proyecto, que no me desanimara y llegara a buen puerto.

También quiero dar las gracias a Carlos y María por sus consejos y opiniones sobre los Bits de Inteligencia, creo que vuestros alumnos son afortunados de teneros como profesores por vuestra pasión y forma de trabajar.

Por último, también quiero dar las gracias a Diego y Rodrigo por su ayuda y apoyo a la hora de realizar el proyecto.

# ANEXOS

# ANEXO I: VERSIONES DE ANDROID


*Ilustración 59: Versiones de Android*

En este Anexo I voy a hablar sobre las diferentes versiones que ha habido de Android desde sus inicios hasta hoy.

Cada versión de Android que ha se ha ido publicando, ha servido para ampliar, mejorar y corregir fallos de la versión anterior, incluyendo nuevas funcionalidades, características, librerías, métodos, etc. Además de su número de versión, cada versión lleva asociado un Nivel de API, que es un control puramente numérico y que lo haca característica e identificable a la versión y a través del cual las aplicaciones saben si pueden instalarse y correr en esa versión de Android o no. Cada versión a partir de Android 1.5 también cuenta con su propio logotipo, nombre dulce y estatua en la sede de Google, aunque para seguir esta tradición también se dio el nombre de un dulce a las versiones existentes antes de Android 1.5.

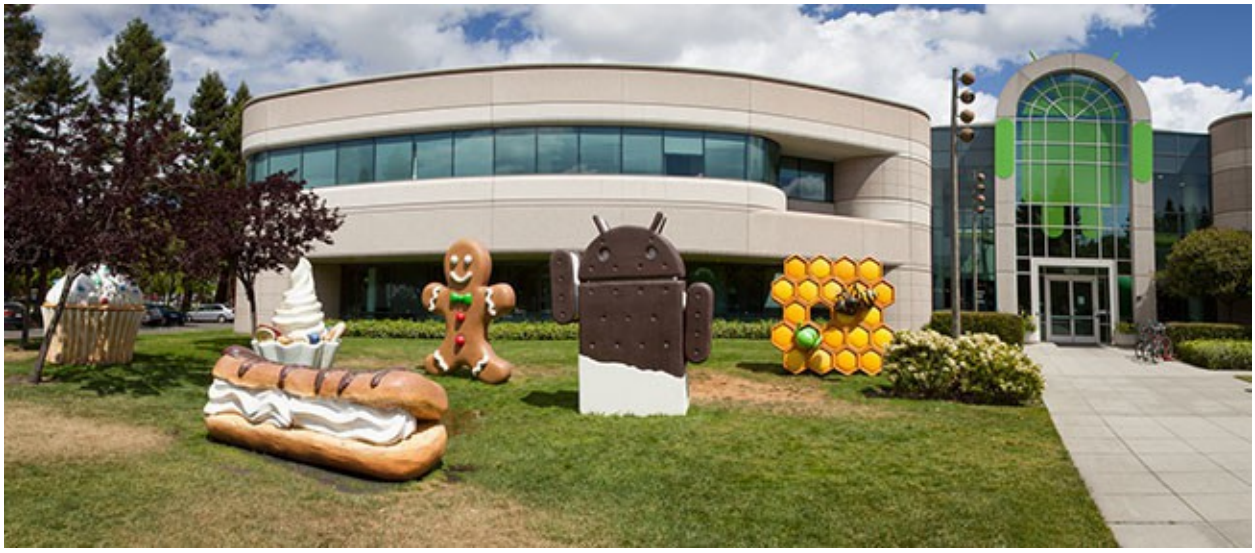

*Ilustración 60: Sede de Google*

Como ya mencione en el apartado **Estado del Arte**, pero considero importante repetir, el desarrollador deberá tener muy en cuenta la versión de Android sobre la que desarrollar la aplicación, ya que dependiendo el nivel de API que se elija, estarán disponibles o no ciertas funcionalidades, se limitará la visibilidad de la aplicación (solo los terminales que tengan instalada la versión elegida o una posterior podrán instalar y ejecutar la aplicación).

Antes de comenzar con las versiones, quiero comentar que el Kit de Desarrollo o *Software Development Kit* (SDK) se lanzó el 12 de noviembre de 2007. Después dela fecha de lanzamiento de las versiones *Alfa* y *Beta*.

Comencemos con el resumen dela versiones de Android y los cambios que se han ido produciendo:

1. Android Alfa

Hubo al menos dos lanzamientos internos del software dentro de Google y la OHA antes del lanzamiento de la versión beta en noviembre del 2007. Para los lanzamientos internos se escogieron nombres de robots ficticios, como “Astroboy”, “Bender” y “R2-D2”.

Dan Morril creó algunos de los primeros logotipos de mascotas, pero el actual logo verde de Android fue diseñado por Irina Blok. El líder del proyecto, Ryan Gibson, concibió la idea de darles nombres de pastelería a la mayoría de versiones públicas, comenzando con Android 1.5 "Cupcake".

2. Android Beta

   La beta de Android fue lanzada el 5 de noviembre de 2007.

3. Android 1.0 – Apple Pie

   - Nivel de API 1

*Ilustración 61: Logo Android 1.0 Apple Pie*

Android 1.0 *Apple Pie* (Tarta de Manzana) es la primera versión comercial del software, fue lanzada el 23 septiembre de 2008. Ofrecía un conjunto de características y funcionalidades que ahora consideramos básicas, pero que para la época eran punteras. Proponía una tienda de aplicaciones llamada *Android Market*, así como compatibilidad y sincronización con los múltiples servicios de aplicaciones de Google (Maps, Contactos, Calendar, mensajería instantánea e incluso Youtube) o notificaciones.

El primer dispositivo Android, el HTC Dream incorporó las siguientes características de Android 1.0:

- Android Market: Programa con un mercado para la descarga y actualización de aplicaciones.
- Navegador Web para visualizar páginas webs en full HTML y XHTMAL. Múltiples páginas pueden ser mostradas como ventanas.
- Soporte para cámara. Aunque soporta la cámara, no da soporte para cambiar la resolución de la cámara, balance de blancos, calidad, etc.
- Carpetas que permiten agrupar iconos de aplicaciones en uno solo dentro de la pantalla principal.
- Acceso a servidores de correo electrónico por web. Soporte POP3, IMAP4 y SMTP.
- Sincronización de *Gmail* con la aplicación *Gmail*.
- Sincronización de *Google Contacts* con la aplicación de contactos/personas.
- Sincronización de *Google Calendar* con la aplicación de calendario
- *Google Maps* con *Latitude* y *Street View* para ver mapas e imágenes por satélite, así como para encontrar negocios locales y obtener direcciones de conducción usando GPS.
- *Google Sync* que permite la administración de la sincronización OTA de *Gmail*, *Personas* y *Calendario*.
- *Google Search* que permite a los usuarios buscar en Internet, en aplicaciones del teléfono móvil, en contactos, en calendario, etc.
- Mensajería instantánea con *Google Talk*.

- Mensajería instantánea, mensajes de texto, MMS.
- Reproductor multimedia que permite la administración, importación y reproducción de archivos multimedia, sin embargo, esta versión carece de soporte de vídeo y estéreo por Bluetooth.
- Las notificaciones aparecen en la barra de estado, con opciones para configurar alertas por timbre, LED o vibración.
- Marcación por voz que permite marcar y llamar sin escribir nombre o número.
- Fondo de escritorio que permite al usuario configurar una imagen de fondo o una foto detrás de los iconos y *widgets* de la pantalla de inicio.
- Reproductor de vídeo Youtube.
- Otras aplicaciones disponibles son Alarma, Calculadora, Marcación (teléfono), Pantalla de inicio (*launcher*), Imágenes (Galería) y ajustes.
- Soporte para Wi-Fi y Bluetooth.

4. Android 1.1 – Banana Bread
   - Nivel de API 2

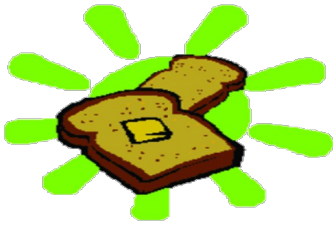

*Ilustración 62: Logo Android 1.1 Banana Bread*

La actualización Android 1.1 *Banana Bread* (Pan de Plátano) fue lanzada, inicialmente sólo para el HTC Dream. Android 1.1 fue conocido como "Petit Four" internamente, aunque este nombre no se utilizó oficialmente. La actualización resolvió fallos y problemas, cambio la API y agregó una serie de características. Mantuvo la interfaz del sistema casi intacta, las bases de Android ya estaban asentadas. No existen apenas usuarios con esta versión actualmente, aunque es la versión que debemos elegir si queremos una compatibilidad total con todos los dispositivos. Las características que se añadieron fueron:

- Detalles y reseñas disponibles cuando los usuarios buscan negocios en *Mapas*.
- Pantalla en llamada más larga por defecto cuando están en uso el manos libres, además la habilidad de mostrar/esconder el marcador.
- Posibilidad de guardar archivos adjuntos en los mensajes.
- Añadido soporte para marquesina en diseños de sistemas.

5. Android 1.5 – Cupcake
   - Nivel de API 3

*Ilustración 63: Logo Android 1.5 Cupcake*

La actualización de Android 1.5 *Cupcake* (Magdalena) se lanzó el 30 de Abril de 2009, está basada en el núcleo de Linux 2.6.27. La actualización incluye varias nuevas características y correcciones de interfaz de usuario. Fue la primera en presentarse con el nombre dulce con la letra C, las versiones 1.0 y 1.1 lo recibirían después. Fue la primera gran versión de Android, que paso del gran desconocimiento a ser una plataforma con cierto reconocimiento. Empezó a recibir

el apoyo de muchos fabricantes que diseñaron y desarrollaron dispositivos con esta versión de Android y por lo tanto tubo ya un número de usuarios considerable, aunque ya casi no queda ninguno. Las principales características y mejoras fueron:

- Soporte para teclados virtuales de terceros con predicción de texto y diccionarios de usuarios para palabras personalizadas, ya no es necesario que los dispositivos tengan teclado físico.
- Soporte para *Widgets*, vistas de miniaturas de las aplicaciones que pueden ser incrustadas en otras aplicaciones (como la pantalla de inicio).
- Grabación y reproducción en formatos MPEG-4 y 3GP. Proporciona grabación avanzada en audio y vídeo.
- Auto-sincronización y soporte para *Bluetooth* estéreo añadido (perfiles A2DP y AVRCP). Permite conectarse automáticamente a auriculares *Bluetooth*.
- Características de Copiar y Pegar añadidas al navegador web.
- Fotos de los usuarios mostradas para favoritos en los contactos.
- Marcas de fecha/hora mostradas para eventos en registro de llamadas y acceso con un toque a la tarjeta de un contacto desde un evento del registro de llamadas.
- Pantallas de transiciones animadas.
- Agregada la opción de auto-rotación.
- Agregada la animación de inicio por defecto actual.
- Habilidad de subir vídeos a *Youtube*.
- Habilidad de subir fotos a *Picasa*.

6. Android 1.6 – Donut
   - Nivel de API 4

*Ilustración 64: Logo Android 1.6 Donut*

Android 1.6 *Donut* (Donut), se lanzó el 15 de Septiembre de 2009. Está basado en el núcleo Linux 2.6.29. En la actualización se incluyen numerosas características nuevas.

Debido al relativo éxito de la versión *Cupcake*, pocos meses más tardes se lanzó está nueva versión con la que Android continuó creciendo y asentándose como plataforma alternativa. Sus principales característica y novedades fueron:

- Mejora en la búsqueda por entrada de texto y voz (**reconocimiento de voz**) para incluir historial de favoritos, contactos y la web.
- Habilidad de los desarrolladores de incluid su contenido en los resultados de las búsquedas.
- Motor multi-lenguaje de *Síntesis de habla* para permitir a cualquier aplicación de Android "hablar" una cadena de texto.
- Búsqueda facilitada y habilidad para ver capturas de las aplicaciones en el *Android Market* (actualmente *Google Play Store*).

- Galería, cámara y videocámara con mejor integración, con rápido acceso a la cámara.
- La galería ahora permite a los usuarios seleccionar varias fotos para eliminarlas.
- Actualización del soporte a tecnología para CDMA/EVDO, 802.1x, VPNs y un motor *text-to-speech*.
- Aplicación para trabajar con diferentes densidades de pantalla y soporte para resoluciones de pantalla WVGA.
- Mejoras de velocidad en búsqueda y aplicaciones de cámara.
- *Framework* de gestos ampliado y una nueva herramienta de desarrollo *GestureBuilder*.
- Aparece un nuevo atributo XML, *onClick*(), que puede especificarse en una vista.

7. Android 2.0 / 2.1 – Eclair
   - Nivel de API 5-7

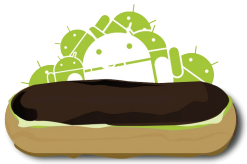

*Ilustración 65: Logo Android 2.0/2.1 Eclair*

Android 2.0 *Eclair* (Pepito) fue lanzada el 26 de Octubre de 2009. Está basado en el núcleo de Linux 2.6.29. Solo un mes más tarde de que se lanzara la versión 1.6, Google lanzó esta versión 2.0 (Nivel de API 5) que ha sido la segunda gran versión de Android, vino acompañada de una amplia lista de mejoras y novedades. Esta versión en concreto apenas tuvo pocos usuarios, ya que la mayoría de fabricantes pasaron de la versión 1.6 a la 2.1. En esta época Android tuvo una evolución realmente rápida liberando varias versiones en pocos meses, como demuestra que la esta versión de Android tuvo varias subversiones que son Android 2.0.1 (Nivel de API 6) y Android 2.1 (Nivel de API 7) de las que haré una reseña también.

Las principales novedades y características de la versión Android 2.0 fueron:

- Sincronización de cuenta expandida, permitiendo a los usuarios agregar múltiples cuentas al dispositivo para sincronización de correos y contactos.
- Soporte para intercambio de correo, con bandeja de entrada combinada para buscar correo desde múltiples cuantas en la página.
- Soporte para Bluetooth 2.1.
- Habilidad para tocar una foto de un contacto y seleccionar llamar, enviar SMS o correo a dicho contacto.
- Habilidad para gestión del almacenamiento de todos los mensajes SMS y MMS, con eliminación de los mensajes más antiguos en una conversación automáticamente cuando se alcanza un límite definido.
- Nuevas características para la cámara, incluyendo soporte de flash, zoom digital, modo escena, balance de blancos, efecto de colores y enfoque macro.
- Mejorada la velocidad de escritura en el teclado virtual, con diccionario inteligente que aprende el uso de palabras e incluye nombres de contactos como sugerencias.
- Renovada interfaz de usuario del navegador con imágenes en miniatura de marcador, zoom de toque-doble y soporte para HTML5.

- Vista agenda del *Calendario* mejorada, que muestra el estado asistiendo a cada invitado y la capacidad de invitar a nuevos usuarios/invitados a los eventos.
- Optimización dela velocidad en *Hardware* y GUI renovada.
- Soporte para más tamaños y resoluciones de pantallas, con mejor ratio de contraste.
- Mejorado Google Maps 3.1.2.
- Clase *MotionEvent* mejorada para rastrear eventos multi-touch y pantallas multitáctil.
- Adición de fondos de pantalla animados, permitiendo la animación de imágenes de fondo de la pantalla de inicio para mostrar movimiento.

Poco tiempo después, el 3 de Diciembre de 2009, se lanzó la versión Android 2.0.1 (Nivel de API 6) con cambios menores en la API, cambios en el comportamiento del *framework* y solución de bugs y problemas menores.

También se lanzó bajo ese mismo nombre de *Eclair*, el 12 de enero de 2010 la versión Android 2.1 (Nivel de API 7) que se considera una actualización menor, al estar englobada en el mismo nombre, que tuvo modificaciones menores en la API y solución de bugs.

8. Android 2.2 / 2.2.3 – Froyo
   - Nivel de API 8

*Ilustración 66: Logo Android 2.2 Froyo*

Android 2.2 *Froyo* (Yogur helado) se liberó el 20 de Mayo de 2010. Esta versión está basada en el núcleo Linux 2.6.32. Es una de las versiones más conocidas de la historia de Android y todavía sigue teniendo algún usuario. Ofreció un importante lavado de cara y un amplísimo listado de novedades. Tuvo varias subversiones que comentaré. Sus principales características son:

- Optimizaciones de velocidad, memoria y rendimiento.
- Mejora en la velocidad de ejecución y rendimiento delas aplicaciones gracias al nuevo compilador *Just-in-time* (*JIT*) de la máquina *Dalvik (*ejecución del código de la CPU de 2 a 5 veces más rápido que en la versión 2.1 de acuerdo a varios *benchmarks)*.
- Soporte para la versión 2.0 de OpenGL ES.
- Soporte a Wi-Fi IEEE 802.11n.
- Integración del motor de JavaScript V8 de Chrome en el navegador.
- Soporte para el servicio *Android Cloud to Device Messaging* (C2DM), habilitando notificaciones *push*.
- Soporte para *Microsoft Exchange* mejorado, incluyendo políticas de seguridad, auto-descubrimiento, consulta a la *Global Access List* (GAL), sincronización de calendario y borrado remoto.
- Mejoras en la aplicación del lanzador con accesos directos de las aplicaciones del teléfono y navegador web.

- Funcionalidad de anclaje de red por USB y Wi-Fi hotsot, esto nos ofrece la posibilidad de compartir la conexión a Internet con otros dispositivos (*tethering*).
- Agregada la opción para deshabilitar el acceso de datos sobre la red móvil.
- Actualización de la aplicación *Market* con características de grupo y actualizaciones automáticas.
- Cambio rápido entre múltiples lenguajes de teclado y diccionario.
- Dictado por voz. Facilita que las aplicaciones interaccionen con el reconocimiento de voz y que terceras partes proporcionen nuevos motores de reconocimiento.
- Intercambio de contactos por *Bluetooth*.
- Soporte para *docks Bluetooh* habilitado para automóviles y de escritorio.
- Soporte para contraseñas numéricas y alfanuméricas.
- Soporte para subida de archivos en la aplicación del navegador.
- Soporte para la instalación de aplicaciones en la memoria externa. Se puede preguntar al usuario si desea instalar una aplicación en un medio de almacenamiento externo (como una tarjeta SD), como alternativa a la instalación en la memoria interna del dispositivo.
- Soporte para Adobe Flash.
- Soporte para pantallas de alta resolución (alto número de PPI, 320ppi), como 4" y 720p.
- La Galería permite a los usuarios ver pilas de imágenes mediante un gesto de zoom.
- Servicio para copias de seguridad de datos que se puede realizar desde la propia aplicación para garantizar al usuario el mantenimiento de sus datos.
- Soporte para definir modos de interfaz de usuario (“automóvil” y “noche”) para que las aplicaciones se configuren según el modo seleccionado por el usuario.

Esta versión tuvo tres subversiones, todas ellas dentro del mismo Nivel de API y fueron las siguientes:

- Android 2.2.1: Liberada el 18 de Enero de 2011. Realizó actualizaciones de seguridad, corrección de errores y mejoras de rendimiento.
- Android 2.2.2: Liberada el 22 de Enero de 2011. Arregló fallos menores, incluyendo problemas de routeo de SMS que afectaron al Nexus One.
- Android 2.2.3: Liberada el 21 de Noviembre de 2011. Instaló dos parches de seguridad.

9. Android 2.3.x – Gingerbread
    - Nivel de API 9-10

*Ilustración 67: Logo Android 2.3 Gingerbread*

Android 2.3 *Gingerbread* (Pan de Jengibre) fue lanzado el 6 de Diciembre de 2010. Está basado en el núcleo Linux 2.6.35. Puede ser considera como la versión 2.3, 2.3.0 o 2.3.1 ya que la primera actualización fue la 2.3.2. Es una versión importante dentro de la historia de Android

y actualmente cuenta con algunos usuarios. Tuvo varias subversiones con un cambio de Nivel de API. Sus principales novedades y características fueron:

- Actualización del diseño de la interfaz de usuario con incrementos en velocidad, simpleza y mejorando la experiencia del usuario. Dentro de esta actualización y mejoras de la interfaz de usuario destacamos la mejora de la funcionalidad de "cortar, copiar y pegar" y un teclado en pantalla con capacidad multitáctil.
- Soporte para mayores tamaños de pantalla y resoluciones (WXGA y superiores) debido al éxito que tiene Android entre las tabletas.
- Soporte nativo para SIP y telefonía por Internet VoIP.
- Entrada de texto del teclado virtual más rápida e intuitiva, con mejoras en precisión, texto sugerido y entrada por voz.
- Mejoras en la funcionalidad de copiar/pegar, permitiendo a los usuarios seleccionar una palabra al presionar y mantener, copiar y pegar.
- Soporte para *Near Field Communitacion* (NFC), permitiendo al usuario leer la etiqueta NFC incrustada en los objetos como anuncios publicitarios, *stickers*, pósters, etc.
- Nuevos efectos de audio tales como reverberación, ecualizador, virtualización de auriculares y aumentos de bajos.
- Nuevo *Gestor de Descargas*, mejorado para descargas largas y grandes, que da al usuario fácil acceso a cualquier archivo descargado del navegador, correo electrónico o cualquier otra aplicación.
- Soporte para múltiples cámaras en el dispositivo, incluyendo cámara frontal, si está disponible. Esto se hace pensado en la segunda cámara (cámara frontal) usada en videoconferencia. La incorporación de esta segunda cámara ha propiciado la inclusión de reconocimiento facial para identificar el usuario del terminal.
- Soporte para la reproducción de vídeo por WebM/VP8.
- Codificación de audio por AAC.
- Mejoras en la administración de la energía, con mayor rol activo en aplicaciones de administración que se mantienen activas en el dispositivo por mucho tiempo.
- Mejorado el soporte para el desarrollo en código nativo (NDK).
- Cambio del sistema de archivos de *YAFFS* a *ext4* en dispositivos nuevos.
- Mejoras y facilidades en audio, gráficos y entradas para los desarrolladores de juegos.
- Soporte nativo para más sensores como giroscopios y barómetros.
- Nuevo recolector de basura de la máquina Dalvik, ahora es concurrente para incrementar el rendimiento. Esto sirve para minimizar las pausas de las aplicaciones, ayudando a garantizar una mejor animación y el aumento de la capacidad de respuesta en juegos y aplicaciones similares. Se trata de corregir así una de las lacras de este sistema operativo móvil, que en versiones previas no ha sido capaz de cerrar bien las aplicaciones en desuso.

  En lo referente a las subversiones que hubo, fueron las siguientes:

- Android 2.3.2: Liberada el 9 de Enero de 2011. Nivel de API 9. Realizó mejoras y arreglos en la API.
- Android 2.3.3: Liberada el 9 de Febrero de 2011. Nivel de API 10. Corrección de fallos menores.
- Android 2.3.4: Liberada el 28 de Abril de 2011. Nivel de API 10. Añadió el soporte para chat de vídeo o voz con *Google Talk*, rebajo la seguridad de *SSL* al usar protocolos de cifrado inseguros y dio soporte a la biblioteca Open Accessory. La biblioteca Open Accessory fue introducida en 3.1 (Honeycomb) pero la biblioteca Open Accessory Library a la que se dio soporte en esta versión 2.3.4 agregó el soporte cuando un periférico USB es conectado con software compatible y una aplicación compatible en el dispositivo.
- Android 2.3.5: Liberada el 25 de Julio de 2011. Nivel de API 10. Incluyó mejoras en el sistema, mejoras en el rendimiento de red del dispositivo *Nexus S*, se arregló un fallo en el Bluetooth de los dispositivos *Samsung Galaxy S*, mejoras en la aplicación de correo electrónico, animación de sombras al deslizar por listas, mejoras en el software dela cámara y mejoras en el rendimiento dela batería.
- Android 2.3.6: Liberada el 2 de Septiembre de 2011. Nivel de API 10. Arreglo un fallo en las búsquedas por voz y tuvo como efecto secundario que disminuyo la funcionalidad de Wi-Fi *hotspot* en determinados dispositivos que se solucionó a finales de Septiembre.
- Android 2.3.7: Liberada el 21 de Septiembre de 2011. Nivel de API 10. Añadió el soporte para *Google Wallet*.

10. Android 3.x – Honeycomb
    - Nivel de API 11-13

*Ilustración 68: Logo Android 3.x Honeycomb*

Android 3.0 *Honeycomb* (Panal de Miel) fue lanzado el 22 de Febrero de 2011. Está basada en el núcleo de Linux 2.6.36. Fue la primera actualización exclusiva para tabletas, lo que quiere decir que sólo es apta para tabletas y no para teléfonos Android, se hizo esto para mejorar la experiencia del usuario en dispositivos con pantallas grandes. La nueva interfaz de usuario ha sido completamente rediseñada con paradigmas nuevos para la interacción y navegación. Entre las novedades introducidas destacan los *fragments*, con los que podemos diseñar diferentes elementos del interfaz de usuario. El primer dispositivo con esta versión fue la tableta Motorola Xoom, lanzado el 24 de febrero de 2011. Más tarde, además de para tabletas, se amplió a otros dispositivos de grandes tamaños como televisores Google TV.

A pesar de que la nueva interfaz gráfica está optimizada para tabletas, es compatible con las aplicaciones creadas para versiones anteriores. Esto se consigue gracias a la introducción de librerías de compatibilidad que pueden ser utilizadas en versiones anteriores a la 3.0.

No obstante, a pesar del relativo fracaso de esta versión, quizá motivada por su separación de las versiones para teléfonos inteligentes, *Honeycomb* se mantuvo durante un año con actualizaciones periódicas que introdujeron múltiples y notables cambios. Los más representativos son los relacionados a la interfaz, ya que ésta era muy diferente en una tableta respecto de un teléfono inteligente, si bien también existieron nuevas funcionalidades pensadas precisamente el nuevo formato. En total ocho actualizaciones, que detallaré después de la versión

inicial, que con dos Niveles de API diferentes aunque todas ellas englobadas bajo el nombre de *Honeycomb*, aunque tuvieran versiones numéricas diferentes.

Las características de esta versión incluyen:

- Soporte optimizado para tabletas, con una nueva y "virtual" interfaz de usuario holográfica.
- Agregada la barra de sistema con características de acceso rápido a notificaciones, estados y botones de navegación suavizados, disponible en la parte inferior de la pantalla.
- Añadida la barra de acción (*Action Bar* en inglés), entregando acceso a opciones contextuales, navegación, *widgets* u otros tipos de contenido en la parte superior de la pantalla.
- Multitarea simplificada, tocando Aplicaciones recientes en la barra del sistema permite a los usuarios ver instantáneas de las tareas en curso y saltar rápidamente de una aplicación a otra.
- Teclado rediseñado, permitiendo una escritura rápida, eficiente y acertada en pantallas de gran tamaño.
- Interfaz simplificada y más intuitiva para copiar/pegar.
- Las pestañas múltiples reemplazan las ventanas abiertas en el navegador web, además de la característica de auto completado de texto y un nuevo modo de “incógnito” permitiendo la navegación de forma anónima.
- Acceso rápido a las características de la cámara como la exposición, foco, flash, zoom, cámara frontal, temporizador u otras.
- Habilidad para ver álbumes y otras colecciones de fotos en modo pantalla completa en la galería, con un fácil acceso a vistas previas de las fotografías.
- Nueva interfaz de contactos de dos paneles y desplazamiento rápido para permitir a los usuarios organizar y reconocer contactos fácilmente.
- Nueva interfaz de correo de dos paneles para hacer la visualización y organización de mensajes más eficiente, permitiendo a los usuarios seleccionar uno o más mensajes.
- Soporte para videochat usando *Google Talk*.
- Aceleración de hardware.
- Soporte ara microprocesadores multinúcleo. Se optimiza la máquina virtual Dalvik para permitir multiprocesado, lo que permite una ejecución más rápida de las aplicaciones, incluso aquellas que son de hilo único
- Mejora del uso de los dispositivos en un entorno empresarial. Entre las novedades introducidas destacamos las nuevas políticas administrativas con **encriptación** del almacenamiento, caducidad de contraseña y mejoras para administrar los dispositivos de empresa de forma eficaz.
- Mejoras en el uso de HTTPS con *Server Name Indication* (SNI).
- Soporte para *Filesystem in Userspace* (*FUSE kernel module*).

- Mejora en la conectividad tales como nuevas APIS de Bluetooth A2DP y HSP con *streaming* de audio o la posibilidad de conectar teclados completos por USB o Bluetooth.
- Mejoras en los gráficos 2D/3D gracias al renderizador OpenGL acelerado por hardware. Aparece el nuevo motor de gráficos Rederscript, que saca mayor rendimiento al hardware e incorpora su propia API. Se incorpora un nuevo motor de animaciones mucho más flexible, conocido como animación de propiedades.
- Mejoras multimedia, como listas de reproducción M3U a través de *HTTP Live Sreaming*, soporte a la protección de derechos musicales (DRM) y soporte para la transferencia de archivos multimedia a través de USB con los protocolos MTP y PTP.

En lo que referente a las actualizaciones, fueron las siguientes:

- Android 3.1: Liberada el 10 de Mayo de 2011. Nivel de API 12. Añadió soporte para teclados externos y dispositivos punteros, soporte para *joysticks* y *gamepads*, conectividad para accesorios USB, lista expandida de aplicaciones recientes, refinamiento de la interfaz de usuario, soporte para reproducción de audio con FLAC, soporte para *proxy* HTTP para cada punto de acceso Wi-Fi conectado, bloqueo de Wi-Fi de alto rendimiento manteniendo conexiones Wi-Fi de alto rendimiento cuando la pantalla del dispositivo está apagada y Protocolo de transferencia de fotos y vídeo (PTP/MTP) y de tiempo real (RTP).
- Android 3.2: Liberada el 15 de Julio de 2011. Nivel de API 13. Realizó mejoras en el soporte de hardware, incluyendo optimizaciones para un amplio rango de tabletas, incrementó la capacidad delas aplicaciones para acceder a archivos delas tarjetas SD (por ejemplo para sincronización), modo de vista de compatibilidad para aplicaciones que no han sido optimizadas para resoluciones de pantalla de tabletas, nuevas funciones de soporte de pantalla, dando a los desarrolladores un mayor control sobre la apariencia dela pantalla en diferentes dispositivos.
- Android 3.2.1: Liberada el 20 de Agosto de 2011. Nivel de API 13. Realizó correcciones de errores menores y mejoras de seguridad, estabilidad y red Wi-Fi, actualización del Android Market con actualizaciones del texto de términos y condiciones de fácil lectura, mejoras en el soporte de *Adobe Flash* del navegador, mejoras en la predicción de escritura a mano en chino y actualización de *Google Books*.
- Android 3.2.2: Liberada el 30 de Septiembre de 2011. Nivel de API 13. Realizó arreglos de fallos y otras mejoras menores para el Motorola Xoom 4G.
- Android 3.2.3: Liberada el 1 de Diciembre de 2011. Nivel de API 13. Añadió soporte para "Pay as You Go" para tabletas 3G y 4G .
- Android 3.2.4: Liberada el 10 de Enero de 2012. Nivel de API 13. Realizó arreglados de problemas de conectividad de datos en modo avión en la versión estadounidense 4G del Motorola Xoom.
- Android 3.2.5: Liberada el 13 de Febrero de 2012. Nivel de API 13. Realizó correcciones de errores menores y ampliación de compatibilidades con terminales.
- Android 3.2.6: Liberada el 28 de Febrero de 2012. Nivel de API 13. Realizó correcciones en problemas con la conectividad de datos y otros fallos menores.

11. Android 4.0.x – Ice Cream Sandwich
    - Nivel de API 14-15

*Ilustración 69: Logo Android 4.0 Ice Cream Sandwich*

Android 4.0.0 *Ice Cream Sandwich* (Sándwich de Helado), se lanzó el 12 de Octubre de 2011. Está basado en el núcleo de Linux 3.0.1. Gabe Cohen de Google declaró que Android 4.0 era "teóricamente compatible" con cualquier dispositivo Android 2.3 en producción en ese momento, pero sólo si su procesador y memoria *RAM* lo soportaban. La cualidad más importante es que se unifican las dos versiones anteriores (2.x para teléfonos y 3.x para tabletas) en una sola compatible con cualquier tipo de dispositivo, aunque convivió durante varios meses con *Honeycomb* (Android 3.x). También ser renovó completamente la interfaz y la interacción con el usuario, mejorando de forma notable. Estuvo vigente durante otros diez meses más y al contrario que las versiones pasadas no recibió tantas actualizaciones, lo cual nos indica que **Android había empezado a entrar en la madurez**, aunque si recibió alguna actualización con cambio de Nivel de API incluido, como comentaré un poco más adelante.

Esta versión incluye numerosas novedades, entre ellas:

- Botones software (en pantalla), heredados de Android 3.x, están ahora disponibles para usar en los teléfonos móviles.
- Separación de *widgets* en una nueva pestaña, listados de forma similar a las aplicaciones.
- Facilidad para crear carpetas, con estilo de arrastrar y soltar.
- Lanzador personalizable.
- Buzón de voz mejorado con la opción de acelerar o retrasar los mensajes del buzón de voz.
- Funcionalidad de *pinch-to-zoom* en el calendario.
- Captura de pantalla integrada (manteniendo presionado los botones de bloqueo y de bajar volumen).
- Corrector ortográfico del teclado mejorado.
- Habilidad de acceder a aplicaciones directamente desde la pantalla de bloqueo.
- Funcionalidad copiar-pegar mejorada.
- Mejora del reconocimiento de voz con mayor integración de voz y dictado de texto en tiempo real continuo, se puede empezar a hablar sin esperar a la conexión con el servidor.
- Nueva API de reconocimiento facial, permite al usuario el desbloqueo facial, característica que permite a los usuarios desbloquear los equipos usando software de reconocimiento facial.
- Nuevo navegador web con pestañas bajo la marca de Google Chrome, permitiendo hasta 15 pestañas.
- Sincronización automática del navegador con los marcadores de Chrome del usuario.
- Nueva tipografía para la interfaz de usuario.

- Nuevo gestor para el **uso de datos por Internet,** que permite al usuario ver de forma gráfica el consumo de datos y poner avisos cuando se acerca a cierto límite de uso, y desactivar los datos cuando se ha excedido dicho límite para evitar sobre costes por parte de las operadoras por ejemplo.
- Capacidad para cerrar aplicaciones que están usando datos en segundo plano.
- Aplicación de la cámara mejorada sin retardo en el obturador, ajustes para el *time-lapse*, modo panorámico y la posibilidad de hacer zoom durante la grabación.
- Editor de fotos integrado.
- Nuevo diseño de la galería, organizada por persona y localización.
- Aplicación *People* actualizada con integración en redes sociales, actualización de estados e imágenes en alta resolución.
- Mejora de la API de comunicaciones NFC con *Android Beam*, una característica de Near Field Communication (NFC) que permite el rápido intercambio de corto alcance de enlaces web favoritos de un navegador de Internet, información de contactos, direcciones, vídeos de *YouTube* y otros datos.
- Soporte para el formato de imagen *WebP*.
- Aceleración por hardware de la interfaz de usuario.
- Soporte para Wi-Fi Direct.
- Grabación de vídeo a 1080P para dispositivos con Android de serie.

Respecto a las actualizaciones que recibió, fueron las siguientes:

- Android 4.0.2: Liberada el 29 de Noviembre de 2011. Nivel de API 14. Es una versión de mantenimiento que soluciona problemas menores y *bugs*.
- Android 4.0.3: Liberada el 16 de Diciembre de 2011. Nivel de API 15. Es una actualización importante, aunque no revolucionaria, que trajo consigo numerosas optimizaciones y correcciones de errores, mejoras en gráficos, bases de datos, correcciones de ortográfica, funcionalidades bluetooth, nueva API para desarrolladores que incluía una API de actividad social, mejoras en el calendario, cambios en la cámara para mejorar la estabilidad en los vídeos y resolución QVGA y mejoras de accesibilidad.
- Android 4.0.4: Liberada el 8 de Marzo de 2012. Nivel de API 15. En esta actualización, se introdujeron mejoras en la estabilidad, mejoras en el rendimiento de la cámara, rotación de la pantalla más fluida y mejoras en el reconocimiento de los números en el teléfono.

12. <u>Android 4.1 – Jelly Bean</u>
    - Nivel de API 16

*Ilustración 70: Logo Android 4.1 Jelly Bean*

Android 4.1 *Jelly Bean* (Gominola), se lanzó el 9 de Julio de 2012. Está basado en el núcleo de Linux 3.0.31. Esta versión fue una actualización incremental con el enfoque primario de mejorar la funcionalidad y el rendimiento de la interfaz de usuario, que eran los puntos débiles de Android. La mejora del rendimiento involucró el "*Proyecto Butter*", el cual usa anticipación táctil (para aumentar la velocidad del procesador al tocar la pantalla), triple *buffer*, latencia vsyns extendida y un arreglo en la velocidad de cuadros de 60 fps para crear una fluida y suave interfaz de usuario. El primer dispositivo en correr *Jelly Bean* fue el Nexus 7 (una tableta), que fue lanzado el 13 de julio de 2012.

Sus principales características fueron:

- Soporte de Bluetooth de baja energía.
- Soporte para OpenGL ES 3.0.
- Soporte para resolución 4K.
- Nuevos idiomas como hebreo, árabe, hindi...
- Posibilidad de instalar nuevos teclados.
- Texto bidireccional.
- Mejoras en las notificaciones con un sistema de información expandible personalizado.
- Los *widgets* de escritorio pueden ajustar su tamaño y hacerse sitio de forma automática al situarlos en el escritorio.
- El dictado por voz puede realizarse sin conexión a Internet (de momento, solo en inglés).
- Sistema de localización Wi-Fi en segundo plano.
- Mejoras en la seguridad y encriptación.
- Mejoras en *Google Search*, potenciando la búsqueda por voz.
- Mejoras en *Google Now* que permiten utilizar información de posición, agenda y hora en las búsquedas.
- Nuevos perfiles con acceso restringido.
- Función auto-completado en el *Dial Pad* para facilitar el marcado de números.
- Asistente personal *Google Now* que proporciona información contextualizada.
- Mayor calidad en las APIs *DRM*.
- Mejoras en la compilación de aplicaciones.
- Monitorización de notificaciones.

Tuvo una actualización a la versión Android 4.1.2, conservando el mismo Nivel de API 16 en Octubre de 2012 para solventar *bugs*.

13. Android 4.2 – Jelly Bean
    - Nivel de API 17

*Ilustración 71: Logo Android 4.2 Jelly Bean*

Android 4.2 *Jelly Bean – Gummy Bear* (Oso de gominola), se lanzó el 13 de Noviembre de 2012.

Se introdujeron numerosas mejoras y actualizaciones entre las que destacan:

- Posibilidad de tener múltiples cuentas de usuario en el mismo dispositivo, donde cada cuenta tendrá sus aplicaciones y configuración. Esto solo es posibles en tabletas.
- Los Widgets de escritorio pueden aparecer en la pantalla de bloqueo.
- Nuevo teclado predictivo deslizante.
- Posibilidad de conectar el dispositivo y una TVHD mediante Wi-Fi (*Miracast*).
- Mejoras en las notificaciones.
- Nueva aplicación en la cámara que incorpora la funcionalidad *Photo Sphere* para hacer fotos panorámicas inmersivas, es decir, en 360º.

Tuvo dos actualizaciones dentro de este mismo Nivel de API que fueron:

1. Android 4.2.1: Se lanzó el 27 de Noviembre de 2012 y sirvió para corregir un fallo en la aplicación *People* y se agregó *Bluetooth* para *gamepads* y *joysticks* como dispositivos HID soportados.
2. Android 4.2.2: Se lanzó el 11 de Abril de 2013 y sirvió para corregir diversos fallos como en el *streaming* de audio por *Bluetooth* o fallos del *flash* con la cámara, proporcionó mejoras en el rendimiento, incorporación de nuevas notificaciones de descarga para indicar el porcentaje descargado y el tiempo estimado para la finalización entre otros aportes.

14. Android 4.3 – Jelly Bean

- Nivel de API 18

*Ilustración 72: Logo Android 4.3 Jelly Bean*

Android 4.3 *Jelly Bean – Michel*, se lanzó el 13 de Noviembre de 2012.

Se introdujeron numerosas mejoras y actualizaciones entre las que destacan:

- Mejoras en el modo de conexión externa y de desarrollador (para actualizaciones vía cable USB).
- Mejoras en la seguridad.
- Ya no es necesario pulsar el icono del micrófono para realizar una búsqueda de voz. Solo hay que decir "OK Google" y en seguida ordenar al equipo lo que se necesite.
- Permite enviar a una impresora fotos, documentos y páginas web desde el dispositivo, de manera inalámbrica, si la impresora está conectada a *Google Cloud Prin*t.
- Cuando reciba una llamada de un número de teléfono no están en la agenda, el teléfono buscará coincidencias de las empresas con una lista local de *Google Maps*.
- Nueva máquina virtual de ejecución experimental, *ART*.
- Añade soporte completo para *Chromecast*.

- Optimizado para funcionar tan solo con 512MB de RAM (*Project Svelte*).

Tuvo una actualización para corregir problemas menores. Dicha actualización fue la versión Android 4.3.1. Conservó el mismo Nivel de API se lanzó en Octubre de 2013.

15. Android 4.4 – Kit Kat
    - Nivel de API 19-20

*Ilustración 73: Logo Android 4.4 KitKat*

Android 4.4 *Kit Kat* (Kit Kat), se lanzó el 31 de Octubre de 2013. Google se alió con Nestlé para ponerle nombre a esta versión. Con esta versión llegamos ya a un presente bastante cercano, y es que Android 4.4 es hoy en día **la versión más utilizada de Android en todo el mundo**, algo motivado por sus cuatro actualizaciones que han logrado llevar estabilidad al sistema, primero veremos la versión principal y después las actualizaciones.

Esta versión introdujo muchos cambios, tanto a nivel estético como a nivel interno y de funcionamiento, con las vistas puestas en el futuro.

Sus principales características fueron:

- Adaptación de Android a una gama aún más amplia de dispositivos, incluyendo aquellos con tamaños de memoria RAM de solo 512 MB. Para ello, todos los componentes principales de Android han sido recortados para reducir sus requerimientos de memoria, y se ha creado una nueva API que permite adaptar el comportamiento de la aplicación en dispositivos con poca memoria.
- Se introducen transparencias en la barra de estado y navegación
- Cambios estéticos en el reloj.
- Sustitución de elementos de la interfaz de azul a blanco.
- Introducción del modo inmersivo, en el que la barra de estado y la barra de navegación se ocultan en determinadas aplicaciones para una visualización a pantalla completa.
- Optimización y rendimiento en dispositivos de especificaciones técnicas comedidas, así como la implementación de *zRAM*.
- *WebViews* basadas en el motor de *Chromium*.
- Nuevo marco de transiciones y efectos visuales.
- Implementación de manera opcional y para desarrolladores de la máquina virtual *ART*.
- Desactivado el acceso a las estadísticas de batería a aplicaciones de terceros.
- Los monitores de actividad de red y señal son desplazados al menú de ajustes rápidos.
- Facilitación del acceso de las aplicaciones a la nube con un nuevo marco de almacenamiento. Este marco incorpora un tipo específico de *content provider* conocido como *document provider.*
- Nuevas intenciones para abrir y crear documentos y una ventana de dialogo que permite al usuario seleccionar ficheros.

- Añade un *content provider* para gestionar SMS.

Las diferentes actualizaciones que sufrió esta versión de Android fueron:

1. **Android 4.4.1** (5 de diciembre de 2013): Tuvo las siguientes características:

- Corrección de un error que había con el *widget* reproductor que aparecía en la pantalla de bloqueo cuando estábamos reproduciendo contenido multimedia con una aplicación compatible. Al mantener pulsado el botón pausa nos permite retroceder o avanzar la canción o vídeo, pero en Android 4.4 no funcionaba correctamente y nos podía cambiar de pantalla de bloqueo.
- Se aumenta de 4 a 7 el número de dispositivos a los que se pueden conectar mediante el soporte de *Bluetooth Smart (Low Energy)*.
- Mejora del control de volumen único para dispositivos con *Bluetooth*.
- Mejora del rendimiento del sistema.
- Corrección de fallos con la sincronización de cuentas de correo *Exchange*.
- Mejoras en la alineación de los iconos de la barra de estado.
- Corrección de problemas del volumen con las aplicaciones.
- Mejora del runtime ART para que funcione correctamente con más aplicaciones como *Whatsapp*.
- Corrección del *widget* de acceso rápido a Ajustes que hasta ahora permitía cambiar entre activar y desactivar la localización, ahora se puede configurar también para cambiar los modos de ahorro de energía.
- Implementación del acceso directo a Fotos desde Cámara, siendo éste un paso más hacia la integración completa como galería por defecto.
- Corrección del *bug* que impedía que la barra fuese translucida en la pantalla de bloqueo, así como se mejoró la alineación de los iconos de conexiones, cobertura, batería y reloj.
- Posibilidad de ocultar el teclado simplemente pulsando en una parte vacía de la pantalla. Una pequeña muestra de cómo van puliendo la interfaz.
- Corrección de un *bug* que hacía que algunas aplicaciones que incluyen sonido sonasen más alto que otras por el altavoz, aunque todas ellas tengan el volumen al máximo. Este error se corrige y ahora todas alcanzan el máximo volumen permitido.

2. **Android 4.4.2** (9 de diciembre de 2013): Tuvo las siguientes características:

- Solución de un fallo que simulaba la opción “elegir siempre” cuando aparecía la opción de compartir con distintas aplicaciones, escogiéndose así aplicaciones favoritas sin que el usuario lo hiciese.

- Solucionado el fallo de seguridad que afectaba a los SMS de tipo flash, que permitía mediante el uso de ese tipo de mensajes bloquear, reiniciar e incluso dejar al terminal sin conexión de red.
- Corrección de un fallo que impedía mantener pulsado el botón de Pause en el *widget* del reproductor de música en el *lockscreen* (o cualquier otro reproductor que soportara esta acción) para que apareciese una barra de estado de la canción.
- Desaparición de la pantalla de *App Ops*, que permitía gestionar los permisos y notificaciones de las aplicaciones que estaban instaladas en el dispositivo. Esto molestó a algunos usuarios, pero esta función estaba pensada únicamente para fines de desarrollo y nunca para ser accesible por el usuario final.
- Solución de un problema de batería que provocaba el desgaste de esta muy rápidamente.

3. **Android 4.4.3** (2 de Junio de 2013): Esta versión no tuvo grandes cambios de cara al usuario sino que se enfocó a la corrección del sistema operativo corrigiendo *bugs*. Aunque si tuvo varios cambios para mejorar la experiencia del usuario. Tuvo las siguientes características:

- Arreglos en la conexión de datos.
- Optimización del servicio *mm-qcamera-daemon*.
- Arreglos de enfoque de cámara en los modos HDR y normal.
- Arreglos de "*wakelock*" en el gestor de batería.
- Múltiples correcciones en el soporte *Bluetooth*.
- Solución de reinicios aleatorios.
- Solución de la desaparición de accesos directos de algunas aplicaciones tras su actualización.
- Arreglos de seguridad en la depuración USB.
- Arreglos de seguridad en los accesos directos de las aplicaciones.
- Solución en la conexión automática WI-FI.
- Ajustes en MMS, Email/Exchange, Calendario, Contactos, DSP, IPv6 y VPN.
- Solución del atasco en la pantalla de activación.
- Arreglo del LED en las llamadas perdidas.
- Arreglo del gráfico de uso de datos.
- Arreglos en VoIP.
- Corrección para conformidad de la FCC.
- Nueva Interfaz del marcador.
- Corrección de subtítulos.

4. **Android 4.4.4** (19 de Junio de 2013): Lo razón principal de esta actualización fue eliminar una vulnerabilidad de *man-in-the-middle* en *OpenSSL* arreglando CVE-2014-0224.

Además de las actualizaciones mencionadas, hubo otras actualizaciones que conllevaron un cambio en el Nivel de API, siendo el nivel 20 y se centraron en los dispositivos *wearables*. Fundamentalmente *smartwaches* o relojes inteligentes. Estas actualizaciones fueron:

1. **Android 4.4W** (25 de Junio de 2014): Es la versión inicial de la plataforma Android Wear para dispositivos *wearables* como *smartwatches* (relojes inteligentes). Es igual que Android 4.4 *KitKat*, pero con añadiendo extensiones para los dispositivos portátiles *wearables*.
2. **Android 4.4W.1** (2 de Septiembre de 2014): Se actualizó la Interfaz de usuario para utilizar la navegación con *Google Maps* y las alarmas.
3. **Android 4.4W.2** (21 de Octubre de 2014): Se añadió el soporte para GPS y la reproducción de música sin conexión.

16. Android 5.0 – Lollipop

- Nivel de API 21-22

*Ilustración 74: Logo Android 5.0 Lollipop*

Android 5.0 *Lollipop* (Piruleta), se lanzó el 3 de Noviembre de 2014. Inicialmente tiene el Nivel de API 21, recibió varias actualizaciones y hasta legar a la versión Android 5.1 donde se dio el salto al Nivel de API 22. Ahora nos centramos en la primera versión que trajo muchos cambios, sobre todo en la interfaz gráfica que se renovó por completo para incluir el denominado "*Material Design*", el cambio de la máquina virtual, la unificación de todos los sistemas, etc. Para concretar, las principales características fueron:

- Extensión y unificación de Android a todos los dispositivos como tabletas, teléfonos inteligentes, dispositivos *wearables*, Android TV, Android Auto, dispositivos empotrables, etc.
- ***Material Desing***: Ofrece un nuevo diseño intrépido y colorido, una sensible interfaz de usuario para que las experiencias sean más coherentes e intuitivas en todos los dispositivos. Tiene movimientos de respuesta naturales, iluminación y sombras realistas, familias de elementos visuales que hacen que sea más fácil de navegar en el dispositivo. Nuevos colores vivos, tipografía e imágenes de ayuda de borde a borde de enfocar su atención.
- **Notificaciones**: Dispone de nuevas formas de controlar cuándo y cómo se reciben mensajes, se facilita que el usuario sólo pueda ser interrumpido cuando él quiere. Permite ver y responder a mensajes directamente desde la pantalla de bloqueo. Incluye la capacidad de ocultar contenido sensible para estas notificaciones. Se puede programar el tiempo durante el cual sólo las notificaciones de prioridad aparecen. También, las llamadas entrantes no interrumpen lo que estés haciendo. Se puede optar por responder a la llamada o simplemente seguir haciendo lo que se esté haciendo.

Clasificación más inteligente de notificaciones. Permite ver todas las notificaciones en un solo lugar tocando la parte superior de la pantalla.

- **Batería**: Aparece una característica de ahorro de batería que extiende el uso de la batería de los dispositivos hasta 90 minutos. El tiempo estimado de batería restante aparece cuando el dispositivo está enchufado. El tiempo restante de batería antes de tener que cargar el dispositivo de nuevo ahora se puede encontrar en la configuración de la batería.
- Se utiliza definitivamente la máquina virtual *ART* con anticipación de tiempo (*AOT*) con mejoras en la compilación y en la recogida de basura.
- Soporte para *CPUs* de 64 bits en procesadores ARM, x86, y MIPS.
- Soporte para *OpenGL ES 3.1* y *Android Extension Pack* (AEP) en configuraciones de GPU soportadas.
- Pantalla de actividades recientes con tareas en lugar de aplicaciones, hasta un máximo configurado de tareas por aplicación.
- Mejoras en los dibujos Vectoriales para que se puedan escalar sin perder definición.
- Soporte para vistas previas de impresión.
- Cambio en la Pantalla de bloqueado, ya no soporta *Widgets,* pero proporciona accesos directos a aplicaciones y configuraciones de notificación.
- Renovadas la Bandeja de notificaciones y el desplegable de configuraciones rápidas.
- *Project Volta* para mejorar la duración dela batería.
- Mejoras en las búsquedas, que ahora se pueden realizar dentro de la configuración del sistema para un acceso más rápido a los ajustes particulares.
- Los inicios de sesión de usuarios y múltiples cuentas de usuario están disponibles en más dispositivos, como en los teléfonos inteligentes y no solo en las tabletas.
- Entrada y salida de audio a través de dispositivos USB.
- Las aplicaciones de terceros recuperan la capacidad para leer y modificar los datos ubicados en cualquier lugar del almacenamiento externo, como tarjetas SD.
- Fijación de pantalla de una aplicación para la actividad restringida de usuario. Es decir, se introduce un modo de bloqueo que impide al usuario salir de una aplicación y bloquea las notificaciones. Esto podría utilizarse, por ejemplo, para que mientras un usuario realiza un examen, no pueda ver las notificaciones, acceder a otras aplicaciones, o volver a la pantalla de inicio.
- Las aplicaciones utilizadas recientemente se recuerdan incluso después de reiniciar el dispositivo.
- Actualización de *WebViews* de forma independiente a través de *Google Play* por razones de seguridad, en logar de depender de actualizaciones del vendedor de todo el sistema.
- Se añaden quince nuevos idiomas.

- *Tap and Go* permite a los usuarios migrar rápidamente a un nuevo dispositivo Android, el uso de *NFC* y *Bluetooth* para transferir Detalles de la cuenta Google, ajustes de configuración de datos del usuario y las aplicaciones instaladas.
- Se incluye una aplicación de linterna, que funciona en los dispositivos compatibles con un flash de cámara.
- Prioridades personalizables por el usuario para las notificaciones de aplicación.
- Muchas aplicaciones del sistema (*Chrome*, *Gmail*,…) se han incorporado en código nativo para una ejecución más rápida.
- Se incorporan nuevos sensores como el de pulso cardíaco, el de inclinación (para reconocer el tipo de actividad del usuario) y sensores de interacción compuestos para detectar ciertos gestos.

Esta versión, sin variar el Nivel de API, recibió dos actualizaciones que fueron:

1. **Android 5.0.1**: Se lanzó el 2 de Diciembre de 2014. Sirvió para realizar algunas correcciones de errores, incluyendo las ediciones de resolución con reproducción de vídeo y manipulación de errores de contraseñas.
2. **Android 5.0.2**: Se lanzó el 19 de Diciembre de 2014. Sirvió para corregir un error con soporte TRIM y para cambiar la forma en que las alarmas despiertan la *CPU* y como las alarmas compiten por los recursos del sistema.

Después de esas actualizaciones, llego la versión **Android 5.1** que mantenía el nombre de *Lollipop* pero cambió el Nivel de API, situándose en el Nivel 22. Apareció el 9 de Marzo de 2015 y sus características fueron:

- Capacidad para unirse a redes *Wi-Fi* y de control emparejado dispositivos *Bluetooth* desde Ajustes Rápidos.
- Soporte para múltiples tarjetas SIM.
- Protección de dispositivos, si un dispositivo se pierde o es robado permanecerá bloqueado hasta que se inicie sesión con una cuenta de Google, incluso si el dispositivo se restablece a la configuración de fábrica.
- Llamadas de voz de Alta Definición, llamadas clara entre dispositivos con Android 5.1 compatibles.
- Mejoras de estabilidad y rendimiento.

Seguida a esta versión, apareció una actualización que fue la versión **Android 5.1.1**, seguía manteniendo el nombre de *Lollipop* y el Nivel de API 22, se lanzó el 19 de Abril de 2015. Sirvió para realizar correcciones y mejoras en la seguridad, la velocidad y la estabilidad.

17. Android 6.0 – Marshmallow

- Nivel de API 23

*Ilustración 75: Logo Android 6.0 Marshmallow*

Android 6.0 *Marshmallow* (Nubes), se lanzó el 5 de Octubre de 2015. Tiene el Nivel de API 23 tanto la versión inicial como su actualización. Estéticamente conserva el *Material Desing* de la versión anterior. Es la versión actual de Android.

Las principales características:

- Administrador de permisos: El usuario puede decidir que permisos conceder o quitar a cada aplicación por lo que se incrementa la seguridad y privacidad del usuario, como permisos de calendario, contactos, cámara, micrófono, SMS, sensores...
- *Google Now on Tap*: es la expansión de *Google Now* a todo nuestro dispositivo. Con una pulsación prolongada nos aparecerá una tarjeta con información referente a lo que está apareciendo en pantalla. Por ejemplo, si estamos leyendo un correo de un amigo que nos propone ir al cine a ver una película, al pulsar "*Now on Tap*" nos aparecerá la ficha de esa película.
- Soporte nativo para huellas dactilares, elemento fundamental para añadir una dosis de seguridad al proceso.
- *Android Pay*: Plataforma de pagos de Google con dispositivos móviles que combina *NFC* y *Host Card Emulation*.
- Android realizará restauraciones y copias de seguridad de datos completas y automáticas de nuestras aplicaciones para que tras cambiar de dispositivo o tras restablecerlo de fábrica para continuar con todos nuestros datos y aplicaciones.
- *Direct Share*: una forma de compartir contenido más simplificada.
- Nuevo gestor de baterías *Doze*: nuevo sistema que intentará minimizar los *wakelocks* cuando el dispositivo no se está usando de forma activa. Realiza un uso más eficiente de los recursos, con lo que podemos obtener dos horas extras de autonomía
- Soporte oficial para tarjetas SD y USB. Ahora podemos utilizar parte de un dispositivo de almacenamiento externo, para que sea usado como almacenamiento interno. Podemos fragmentar, formatear y encriptar una tarjeta SD para ser usada como memoria interna. También podemos montar y extraer lápices de memoria USB de forma nativa.
- Compatibilidad con lápices bluetooth.
- Pantalla de bloqueo mejorada.
- Controles de volumen simplificados.
- Mejoras en el modo silencio y modo prioridad.
- Opción experimental para modificar partes de la IU del sistema.
- *Direct Links*: podemos vincular cada una de nuestras aplicaciones con direcciones URL, para que determinados enlaces siempre se abran con sus respectivas aplicaciones.
- Explorador de archivos nativo.

- Mejoras en el apartado de memoria RAM.
- Mejoras en la selección de texto.
- Soporte de Hots.
- Mejoras de posicionamiento utilizando redes WiFi y dispositivos Bluetooth.
- Soporte de forma nativa a pantallas 4K.

Tuvo una actualización a la versión **Android 6.0.1**, conservaba el mismo Nivel de API y nombre. Se lanzó el 7 de Diciembre de 2015 y contenía las siguientes características:

- Cambios en la interfaz para tabletas y en el modo de no molestar.
- Nuevos *Emojis* disponibles.
- Conectividad mejorada.
- Doble toque en el botón de encendido para la cámara.
- Mejoras en la rapidez y estabilidad.
- Menú de copiar y pegar más rápido.
- Solucionado el problema con Direct Share.

18. Android 7.0 – Nougat
    - Nivel de API 24

*Ilustración 76: Logo Android 7.0 Nougat*

Es la futura versión de Android. Recibe el nombre de Android *Nougat* (Turrón), en vez de Nutella como se había especulado. El 15 de Junio de 2016 se lanzó la previa para desarrolladores para que puedan ir adaptando sus aplicaciones a esta nueva versión. Esta versión está enfocada principalmente al rendimiento, seguridad y productividad.

Algunas de sus características más destacadas son:

- Soporte para multiventana. Permite mostrar varias aplicaciones en la pantalla a la vez. El sistema operativo mostrará pantalla dos aplicaciones, mostrándolas lado a lado o una encima de la otra. El usuario puede arrastrar la línea divisoria que separa los dos para hacer una aplicación más grande y la otra más pequeña. Los fabricantes decidirán si implementar el modo libre para cambiar el tamaño de cada aplicación en sus dispositivos. Esta característica permitirá a los dispositivos Android TV mostrar una pantalla de *PIP*.
- Notificaciones: Hay numerosas modificaciones en cómo se muestran las notificaciones y las opciones que hay con ellas, por ejemplo responder desde la propia notificación, mostrar las notificaciones agrupadas por aplicaciones, etc. También se modifica la barra de notificaciones que ahora estará en dos capas, una inicial con cinco iconos y una segunda con todos los iconos, entre otros cambios.

- Nuevos *Emojis*.
- Mejoras en el gestor de energía *DOZE*, los dispositivos ahora empezaran a ahorrar energía en cuanto la pantalla esté apagada en vez de cuando la pantalla esté apagada y el equipo en reposo, parado sobre una mesa, como ocurría antes. Reducirá los *wakelocks* de la mayoría de las aplicaciones excepto de las que tengan notificaciones prioritarias.
- Reducción del consumo dela memoria RAM con mejoras en el *Project Svelte*, introducido en la versión Android 4.4 *KitKat*.
- Soporte para Realidad Virtual.
- Nueva *Vulkan* API, destinada a mejorar el rendimiento en juegos fundamentalmente al optimizar el cálculo de gráficos 3D.
- Soporte para Java 8 con *OpenJDK*.

*Ilustración 77: Evolución de Android*

Hasta aquí las versiones de Android que ha habido a lo largo de la historia.

# ANEXO II: INSTALACIÓN DE ANDROID STUDIO Y PRIMER PROGRAMA

*Android Studio* es un entorno de desarrollo integrado o IDE (del inglés *Integrated Development Environment*) creado y soportado por Google para facilitar a los desarrolladores la creación de aplicaciones para Android de forma sencilla. Está disponible para diferentes sistemas operativos como Windows, MAC o Linux, y en todos ellos su funcionamiento es el mismo.

Inicialmente para desarrollar aplicaciones para Android había que usar *Eclipse* junto con el kit de desarrollo de software *SDK* (del inglés *Software Development Kit*) que ofrecía Google, pero un tiempo después de la aparición de *Android Studio*, Google dejo de dar soporte al *SDK* para *Eclipse*.

Para que *Android Studio* pueda funcionar, lo primero es tener instalado el Kit de Desarrollo de Java o *JDK* (del inglés *Java Development Kit*), para ello hay que acceder a la página web de Java (http://www.oracle.com/technetwork/es/java/javase/downloads/index.html) y descargar e instalar la última versión disponible.

Una vez que tenemos instalado *JDK*, tenemos que descargar e instalar *Android Studio*, para ello accedemos a su página web (https://developer.android.com/studio/index.html) y mediante el botón que aparece damos a descargar el archivo.

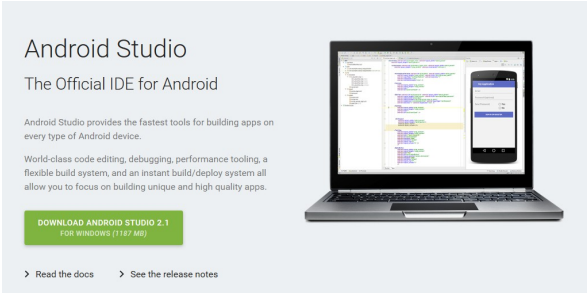


*Ilustración 78: Descargar Android Studio*

Se nos abrirá una ventana donde deberemos leer y aceptar los términos y condiciones y procederá a comenzar la descarga.

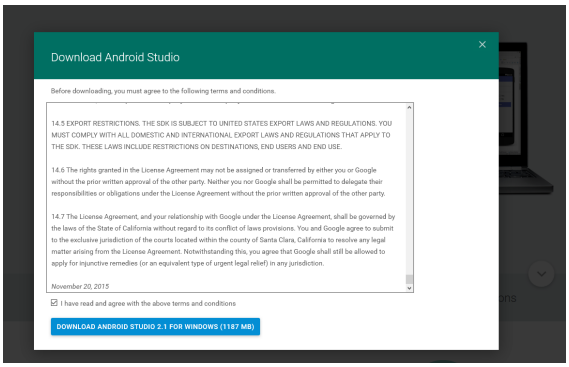


*Ilustración 79: Términos y condiciones de Android Studio*

Una vez que damos a guardar el archivo y mientras se lleva a cabo la descarga, veremos una pantalla con las instrucciones para instalación, con un vídeo incluido, podremos elegir la

versión del sistema operativo que tenemos, indicando que tenemos que instalar el JDK que mencioné anteriormente.

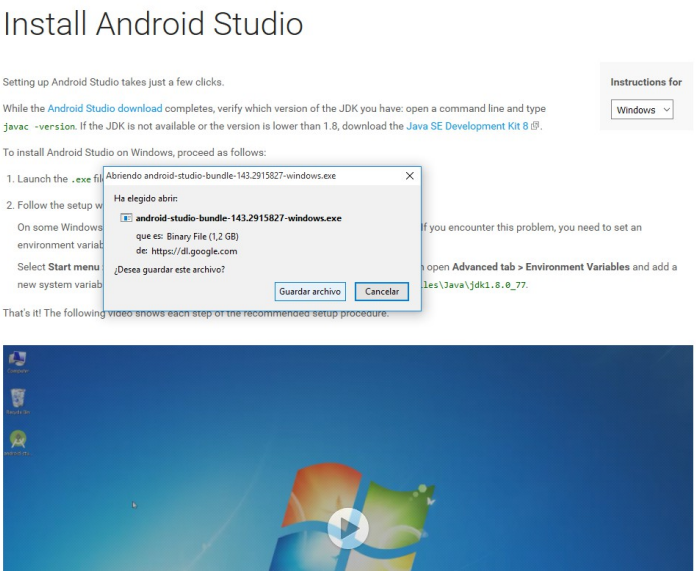


*Ilustración 80: Instrucciones en pantalla de la instalación de Android Studio*

Una vez que se termina de descargar el archivo ejecutable, lo ejecutamos para comenzar con la instalación. Primeramente nos aparece la ventana de presentación donde hay que presionar en el botón "N*ext*".

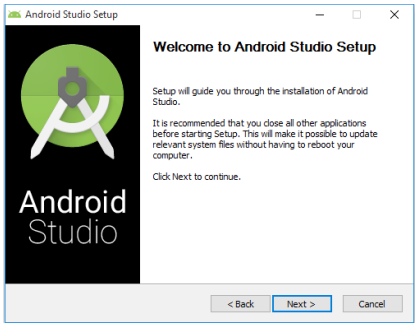


*Ilustración 81: Instalación Android Studio Paso 1*

Seleccionamos los componentes que queremos instalar, *Android Studio* es obligatorio, seleccionamos también los opcionales para poder usar la emulación de dispositivos.

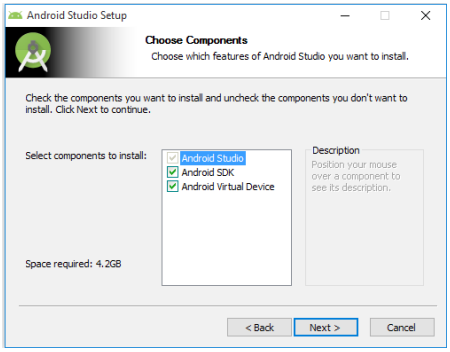


*Ilustración 82: Instalación Android Studio Paso 2*

Leemos y aceptamos la licencia de uso, botón "*I Agree*".

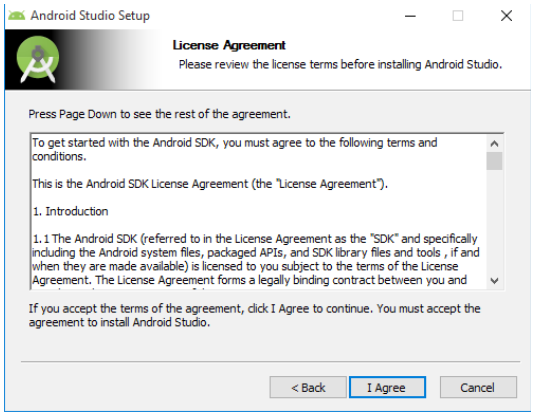


*Ilustración 83: Instalación Android Studio Paso 3*

Seleccionamos las carpetas donde se instalarán *Android Studio* y *Android SDK*, dejamos las que vienen por defecto y presionamos el botón "*Next*".

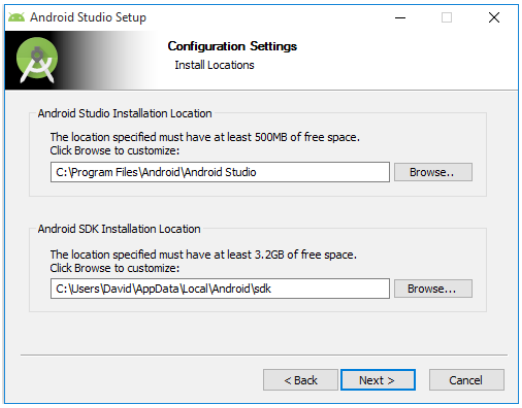


*Ilustración 84: Instalación Android Studio Paso 4*

Elegimos el nombre de la carpeta para el menú inicio de Windows, dejamos el que viene por defecto, y presionamos el botón "Install".

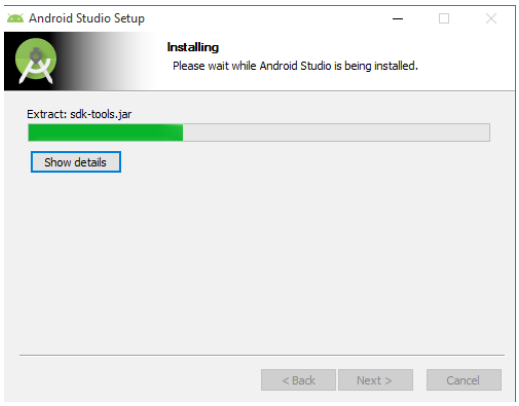


*Ilustración 86: Instalación Android Studio Paso 6*

Comenzará el proceso de instalación.

Cuando finalice el proceso, aparecerá "*Completed*" encima de la barra de progreso, entonces presionamos el botón "*Next*".

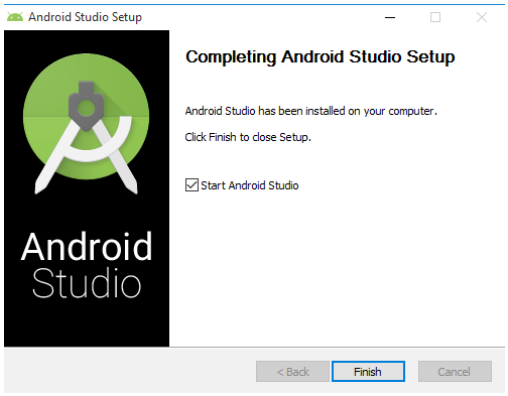


*Ilustración 88: Instalación Android Studio Paso 8*

Aparece una pantalla informándonos que ha finalizado la instalación, dejamos marcada la casilla "*Start Android Studio*" para abrir el programa y presionamos en "*Finish*".

Con esto ya está instalado el programa y ahora se abrirá, vamos con este proceso, ya que al ser la primera vez nos preguntara por una serie de opciones de configuración. Lo primero que aparece es el logotipo del programa que aparece en varios momentos mostrando el proceso de inicio.


*Ilustración 89: Abriendo Android Studio por primera vez Paso 1*

Después de este logotipo, nos preguntará si queremos importar la configuración de una versión anterior, le diremos que no marcando la opción "*I do not have a previous version of Studio or I do not want import my settings*" y presionaremos el botón "*OK*".

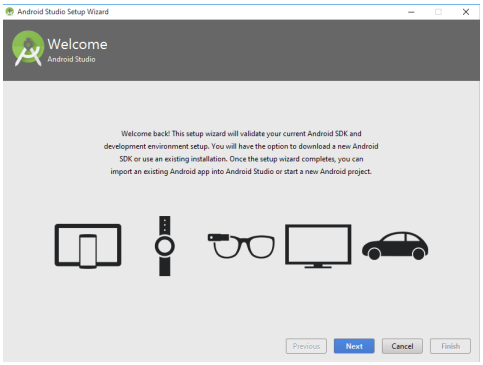


*Ilustración 90: Abriendo Android Studio por primera vez Paso 3*

Llegaremos a una pantalla que nos irá guiando por los pasos de la configuración. Presionamos en el botón "Next".

Seleccionamos la configuración "*Standard*" y presionamos el botón "*Next*".

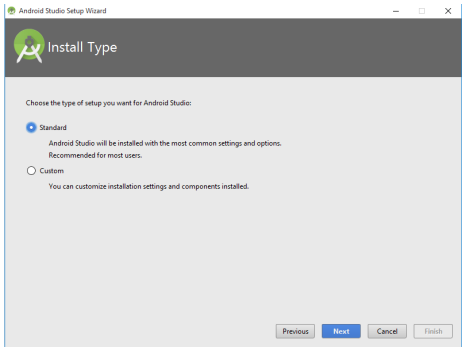


*Ilustración 92: Abriendo Android Studio por primera vez Paso 4*

Ahora de las opciones disponibles, seleccionamos "*Android SDK Plataform*", sin seleccionar el nivel de API que nos deja elegir ya que queremos uno menor para nuestra aplicación, "*Performance (Intel HAXM)*" y "*Android Virtual Device*" y presionamos el botón "*Next*".

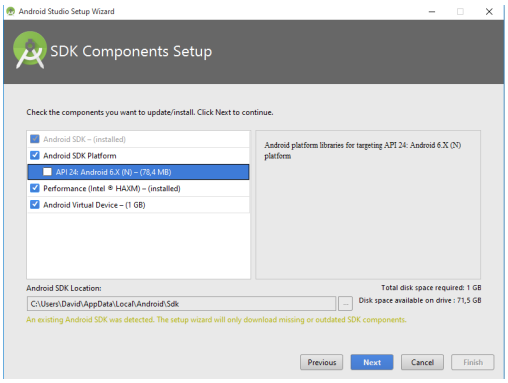


*Ilustración 93: Abriendo Android Studio por primera vez Paso 5*

En la pantalla que nos muestra el resumen de lo que vamos a instalar, presionamos el botón “Finish” y comenzará el proceso de instalación.

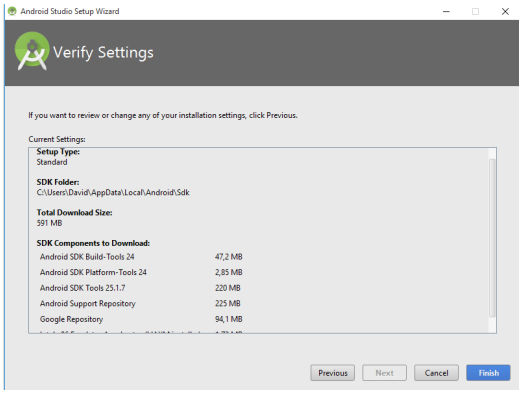


*Ilustración 94: Abriendo Android Studio por primera vez Paso 6*

Esperamos a que concluya el proceso de instalación

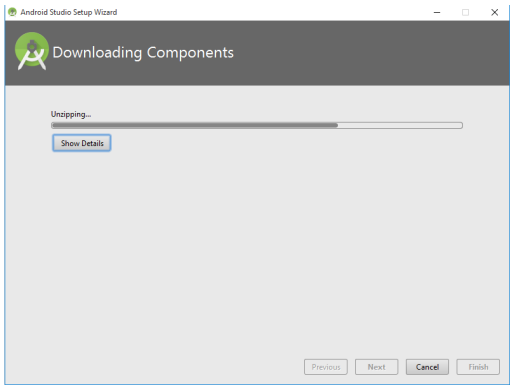


*Ilustración 95: Abriendo Android Studio por primera vez Paso 7*

Una vez concluido, nos muestra todos los procesos que ha llevado a cabo, presionamos el botón “*Finish*”.

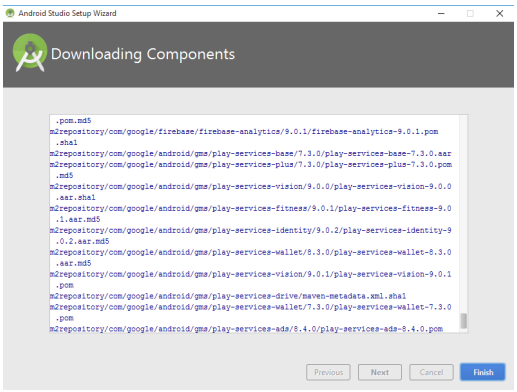


*Ilustración 96: Abriendo Android Studio por primera vez Paso 8*

Se nos presentará una pantalla con varias opciones para que escojamos lo que queremos realizar, como queremos empezar con nuestro primer programa, presionamos sobre la opción “Star a new Android Studio Project”.

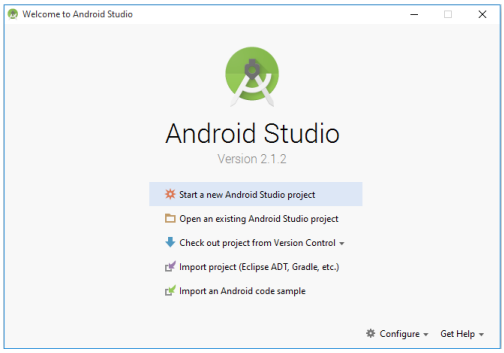


*Ilustración 97: Abriendo Android Studio por primera vez Paso 9*

Se nos presenta en pantalla el proceso para crear un nuevo proyecto, lo primero es seleccionar el nombre, llamaremos a la aplicación "*HolaMundo*" y presionaremos el botón "*Next*".

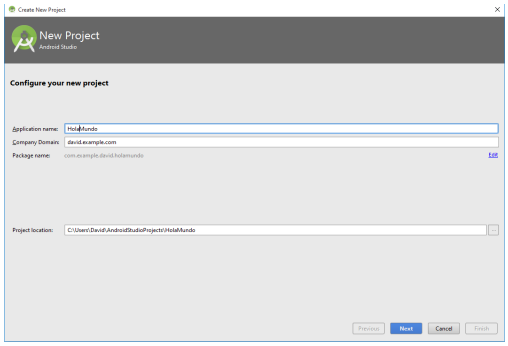


*Ilustración 98: Crear Hola Mundo Paso 1*

Ahora tenemos que seleccionar el tipo de dispositivo al que va dirigida la aplicación y el nivel de API a usar. Seleccionamos "*Phone and Tablet*" con "*Minimum SDK: API 14: Android 4.0 (Ice Cream Sandwich)*" y presionamos el botón "*Next*".

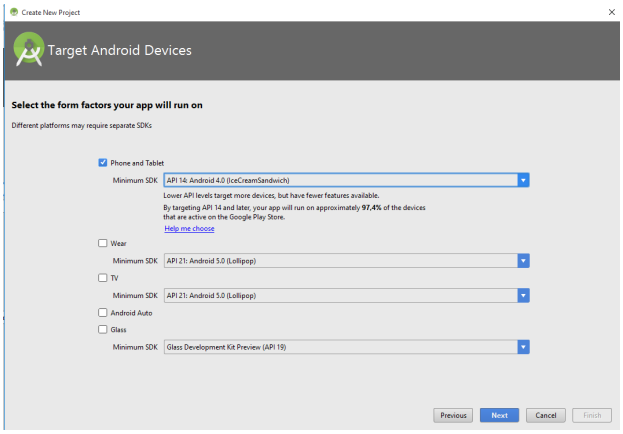


*Ilustración 99: Crear Hola Mundo Paso 2*

Seleccionamos el tipo de actividad que será la actividad principal, en este caso elegimos "*Empty Activity*" y presionamos el botón "*Next*".

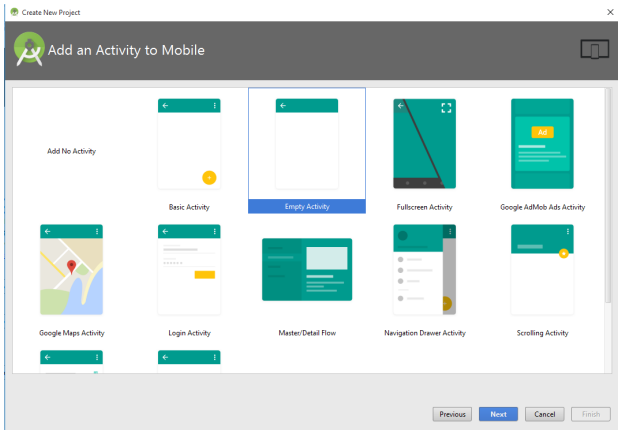


*Ilustración 100: Crear Hola Mundo Paso 3*

Elegimos el nombre de la actividad principal, dejamos el que viene por defecto como "*MainActivity*" y presionamos el botón "*Finish*".

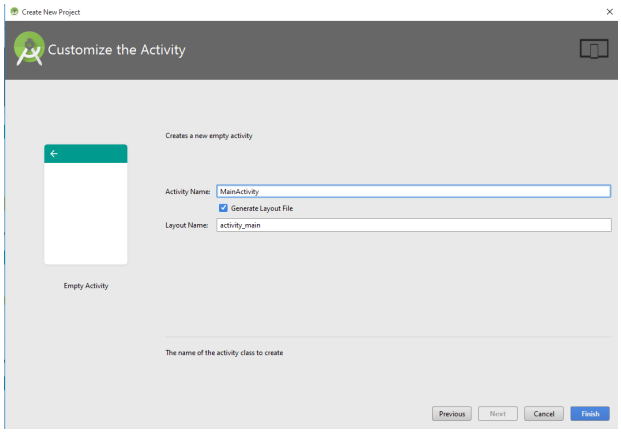


*Ilustración 101: Crear Hola Mundo Paso 4*

Comenzará el proceso de creación de la aplicación.

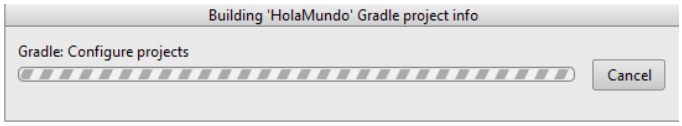


*Ilustración 102: Crear Hola Mundo Paso 5*

Cuando finalice el proceso, nos aparecerán la pantalla del IDE, habrá una ventana de "Tips" que habrá que cerrar y ya tendremos nuestra primera aplicación "Hola Mundo" creada, ya que viene hecha por defecto.

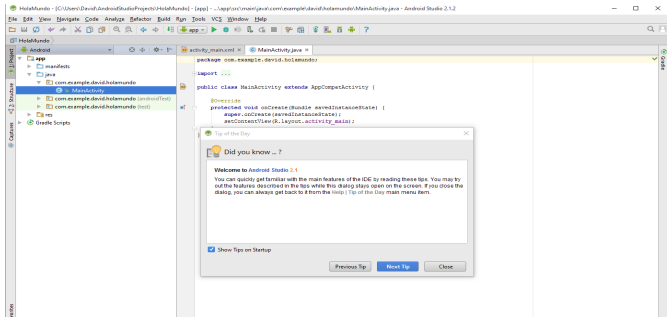

*Ilustración 103: Crear Hola Mundo Paso 6*

Para comprobar el funcionamiento de la aplicación, tenemos que crear un dispositivo virtual para ejecutar la aplicación en el. Para crear el dispositivo, presionamos el botón "*AVD Manager*" del menú.

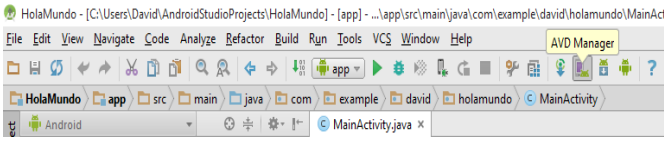


*Ilustración 104: Crear Dispositivo Virtual Paso 1*

Nos aparecerá en pantalla el proceso guiado para crear un dispositivo. Presionamos el botón "*Create Virtual Device*".

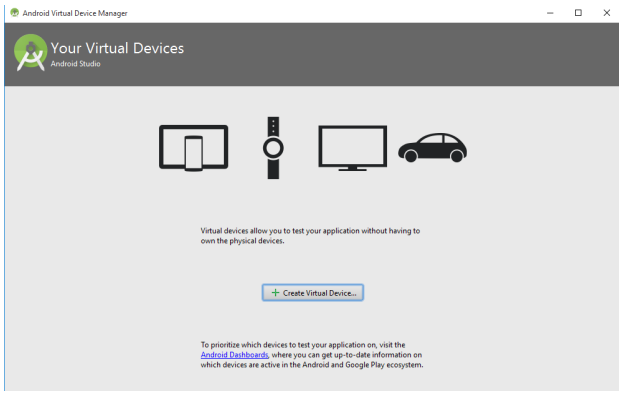


*Ilustración 105: Crear Dispositivo Virtual Paso 2*

Seleccionamos el tipo de dispositivo que queremos, entre los que aparecen en la lista simulando equipos reales y pulsamos en el botón "*Next*".

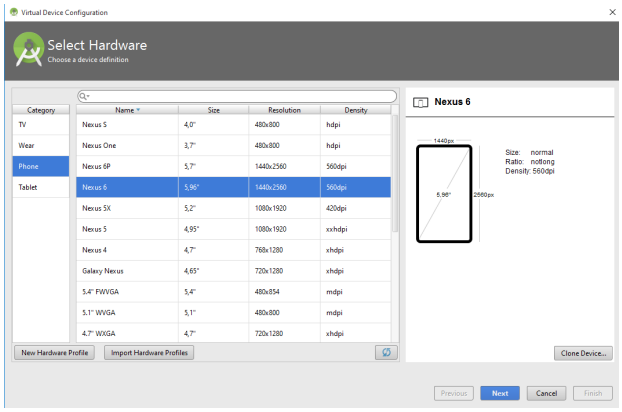


*Ilustración 106: Crear Dispositivo Virtual Paso 3*

En el siguiente paso del proceso, vamos a la pestaña superior "*Other Images*", buscamos la versión de Android *Ice Cream Sandwich 4.0* y presionamos sobre el enlace que pone "*Download*" que esta a su derecha para instalar dicha versión. Comenzará el proceso de descarga.

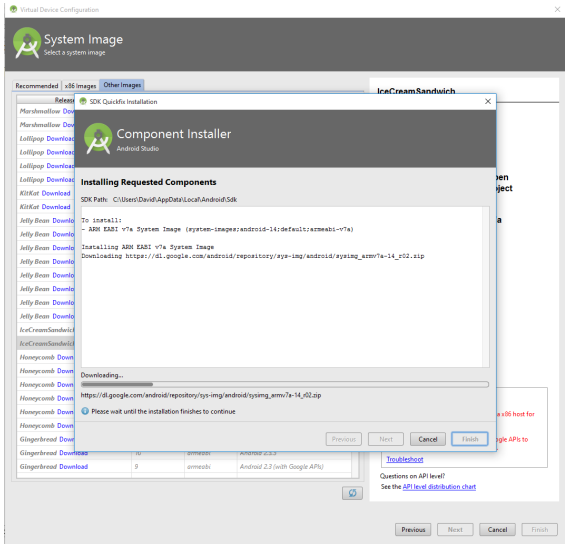


*Ilustración 107: Crear Dispositivo Virtual Paso 4*

Cuando termine la descarga presionamos el botón “Finish”.

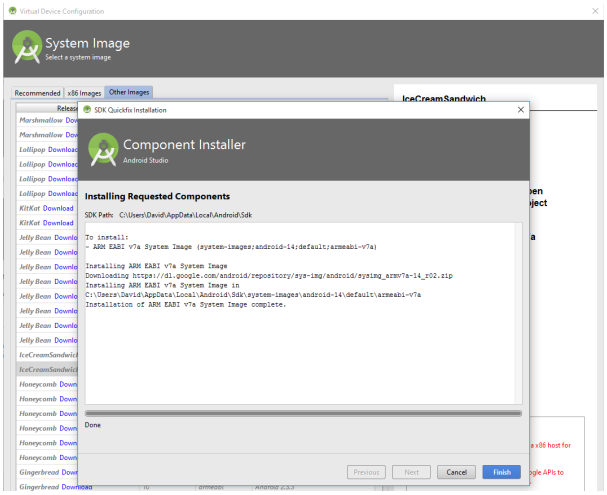


*Ilustración 108: Crear Dispositivo Virtual Paso 5*

Ahora que ya está instalada la versión de Android que queremos, la seleccionamos y damos al botón “*Next*”.

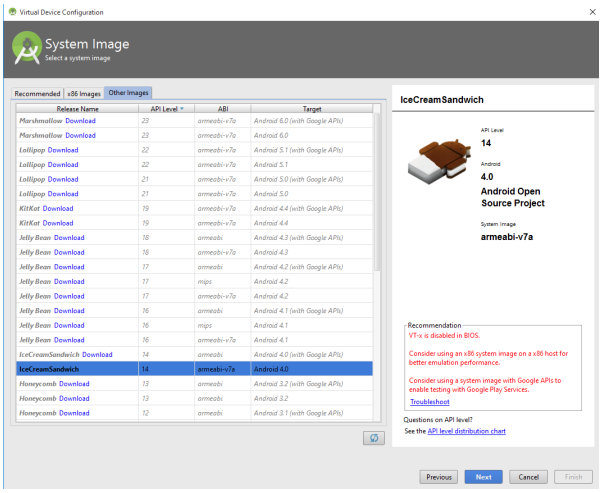


*Ilustración 109: Crear Dispositivo Virtual Paso*

Tendremos que darle un nombre al dispositivo y selecciona algunas opines y presionar el botón "*Finish*".

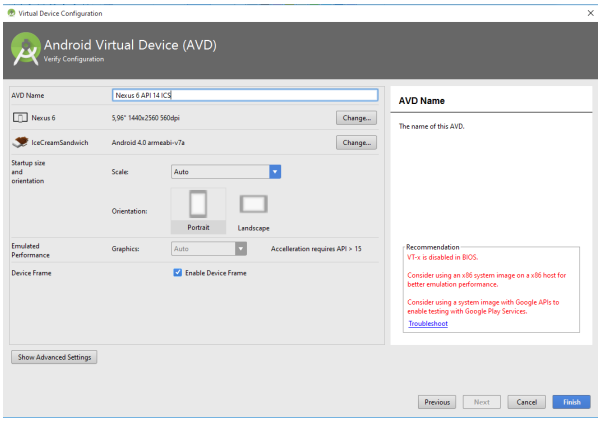


*Ilustración 110: Crear Dispositivo Virtual Paso 7*

Con eso el dispositivo estará creado, ahora tenemos que lanzarlo para poder ejecutar aplicaciones en el. Al terminar de crear el dispositivo, nos aparecerá la pantalla de "*Android Virtual Device Manager*" donde estará el dispositivo que hemos creado, nos vamos a la sección de "*Actions*" y presionamos el botón verde de ejecutar "*Launch this AVD in the emulator*", con ello comenzara a ejecutarse el dispositivo virtual.

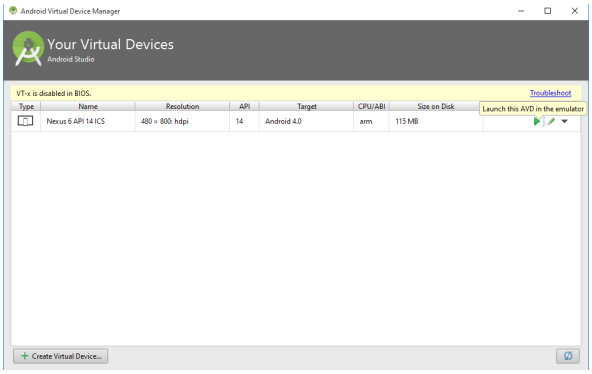


*Ilustración 111: Lanzar dispositivo virtual Paso 1*

Comenzará la carga del dispositivo virtual en el emulador.

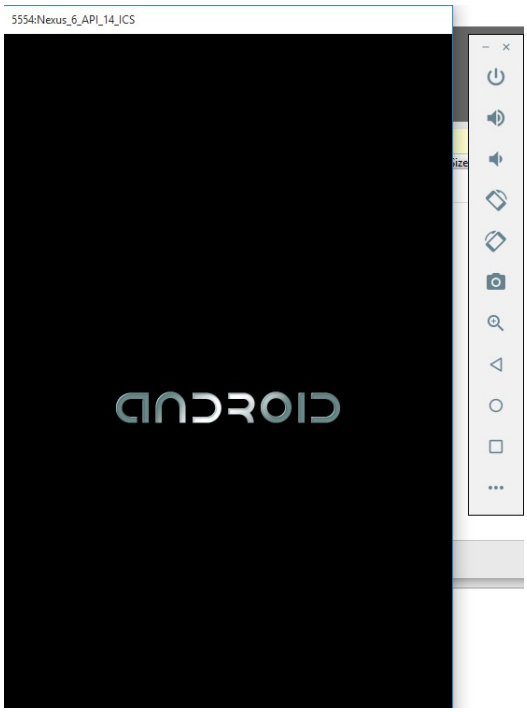


*Ilustración 112: Lanzar dispositivo virtual Paso 2*

Este es un proceso un poco lento, por lo que al cabo de un rato, el emulador habrá terminado de cargar la versión de Android y estará listo para usarse.

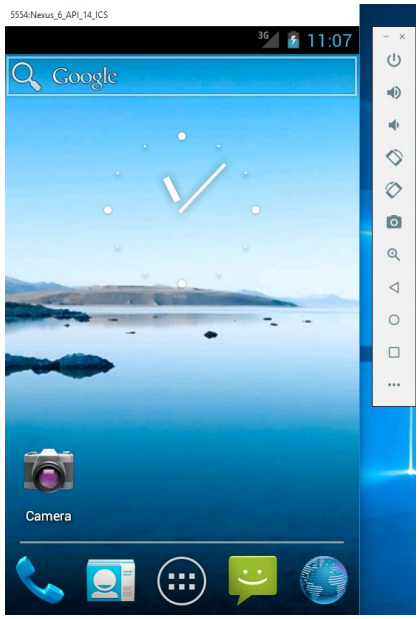


*Ilustración 113: Lanzar dispositivo virtual Paso 3*

Ahora ya estamos listos para ejecutar nuestra aplicación Hola Mundo en el dispositivo virtual. Para ello en la pantalla de *Android Studio*, vamos al menú "*Run*" y seleccionamos la opción "*Run app*".

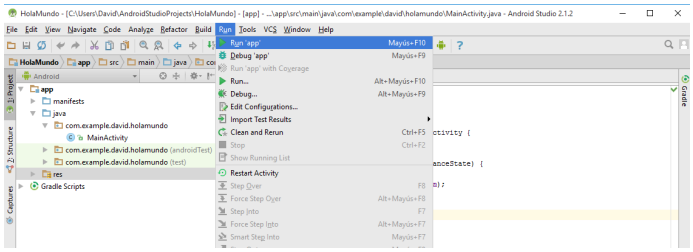

*Ilustración 114: Ejecutar aplicación Paso 1*

Se nos mostrará una pantalla donde tenemos que seleccionar en que dispositivo queremos que se ejecute la aplicación, en este caso solo aparecerá un dispositivo, ya que solo tenemos uno creado. Lo seleccionamos y pulsamos en el botón "OK".

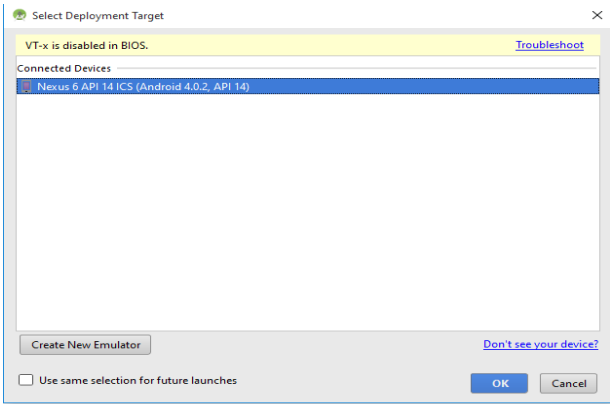


*Ilustración 115: Ejecutar aplicación Paso 2*

Comenzará a cargarse la aplicación en el emulador del dispositivo virtual. Podemos el proceso que se está llevando a cabo en la parte inferior de la pantalla de *Android Studio*.

*Ilustración 116: Ejecutar aplicación Paso 3*

Cuando finalice todo el proceso que lleva a cabo *Android Studio*, la aplicación estará instalada y ejecutándose en el dispositivo virtual.

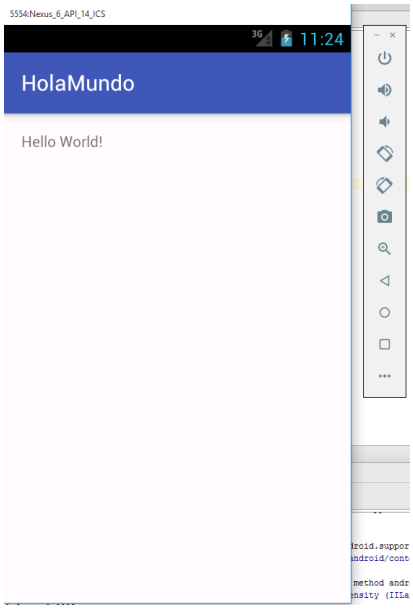


*Ilustración 117: Ejecutar aplicación Paso 4*

Con esto ya está completado todo el proceso de instalar *Android Studio*, crear una aplicación, crear un dispositivo virtual para lanzar las aplicaciones e instalar y probar las aplicaciones en dicho dispositivo virtual, que era el objeto de este Anexo.

Como conclusión, quiero destacar que el proceso de carga del dispositivo virtual es lento y costoso, por lo que lo recomendable es lanzar el dispositivo al principio y dejarlo en ejecución el tiempo que sea necesario en vez de cerrarlo y volverlo a lanzar cada vez que queramos probar la aplicación y sus cambios.