Tema - IMAGEN Y TELEVISION

Titulo - INTERACCION CON JUGUETES PARA USUARIOS CON DISCAPACIDAD USANDO EQUIPOS CON S.O. ANDROID
Autor –FERNANDO PRIETO MOYANO
Titulación - TELEMÁTICA
Tutor - JOSE MANUEL DIAZ LOPEZ
Departamento - DIAC
Presidente - JUAN JIMÉNEZ TRILLO
Vocal - JOSE MANUEL DIAZ LOPEZ
Vocal Secretario - JOSE LUIS RODRIGUEZ VAZQUEZ
Fecha de Lectura- 30 Septiembre 2013

Director -
Tribunal -

El proyecto realizado, trata de una aplicación desarrollada en la plataforma Android, orientada a personas con algún tipo de discapacidad sensorial o psíquica. Dicha aplicación fomenta el uso y la integración de dispositivos móviles en este tipo de sector de población. Está pensada, para que, un usuario con discapacidad, pueda interactuar con ella y de forma transparente a él, se ha creado un sistema mediante el cual, se registra todo el comportamiento que ese usuario ha tenido durante el tiempo de uso de la aplicación, con el fin de llevar un seguimiento del mismo; evaluar si existen cambios en él; determinar si son necesarios algunos cambios en la aplicación que favorezcan una mejoría en cuanto al uso y consecución de resultados en el paciente, etc.

Se ha combinado el uso de una aplicación instalada sobre un sistema operativo de libre distribución, concretamente Android, con un juguete de código abierto, como es Sphero. Eso ha permito el desarrollo de una aplicación perfectamente ajustada a los requisitos funcionales definidos, con una robustez y eficiencia similar a una aplicación de sistemas operativos móviles cerrados.

Es importarte remarcar, que la primera finalidad de este proyecto es ofrecer la posibilidad de usar un juguete, como Sphero, que está orientado a un sector de población sin discapacidad, haciendo uso de cualquier dispositivo móvil, a personas con diferentes grados de discapacidad sensorial, motora y psíquica. Siempre clarificando, que no existe la posibilidad de usar esta aplicación para cualquier tipo y grado de discapacidad, ya que, ello supondría, un proyecto de una envergadura enorme.

El hecho de usar un dispositivo móvil, es un derecho, que todos tenemos. Y por ello, se espera que tras la lectura y comprensión de este proyecto, se motive a los lectores a seguir desarrollando aplicaciones para que cualquier usuario, con discapacidad o no, tenga las mismas oportunidades de interactuación con dispositivos móviles.

The carried out project, is an application developed on the Android platform , aimed at people with some kind of sensory or mental disability . This application encourages the use and integration of mobile devices in this type of population sector. It is designed to be interacted by a disable person and transparently to that user, it has been created a system by which all behavior, that user has had during the time of use of the application, is recorded, that's to keep track of it ; assess whether there are changes in it, determine if changes are needed in the application that favor an improvement in the use and achieving patient outcomes , etc. .

It has combined the use of an application installed on an open source operating system, namely Android, an open source toy , as is Sphero . That has made it possible to develop an application perfectly adjusted to the functional requirements defined with a robustness and efficiency similar to a mobile OS application closed.
It is important to note, that the first aim of this project is to offer the possibility of using a toy, like Sphero , which is geared to a sector of the population without disabilities , using any mobile device , for people with different degrees of sensory impairment , motor and mental . Always clarifying that there is no possibility to use this application for any type and degree of disability, because the magnitude of the project would have been infinitely greater.

The fact of using a mobile device, is a right we all have. And so, it is expected that upon reading and understanding of this project, motivate readers to continue developing applications for any user , disabled or not, have the same opportunities for interaction with mobile devices .

Índice

Capitulo 1: Introducción

En el capítulo introductorio se presenta un enfoque general de los propósitos y objetivos de este proyecto. Los puntos principales del mismo son la motivación, los objetivos, los medios utilizados y los contenidos

1.1 Introducción

El proyecto que se presenta, trata de una aplicación desarrollada en la plataforma Android, orientada a personas con algún tipo de discapacidad sensorial o psíquica. Dicha aplicación fomenta el uso y la integración de dispositivos móviles en este tipo de sector de población. Está pensada, para que, un usuario con discapacidad, pueda interactuar con ella y de forma transparente a él, se registre todo el comportamiento que ese usuario ha tenido durante el tiempo de uso de la aplicación, con fin terapéutico.

Se ha elegido como sistema operativo Android, debido a que es un software de libre distribución, que permite a cualquier usuario desarrollar sus propias aplicaciones y a la vez ver y utilizar aportes de otros desarrolladores, con el fin de crear aplicaciones robustas e igual de eficientes que las que soportan sistemas operativos cerrados.

A pesar de ser un proyecto en el que se combina un dispositivo móvil, concretamente una Tablet PC, con un elemento lúdico, como Sphero; el fin realmente, es la obtención de resultados, para así tratarlos y favorecer una mejora en el usuario con discapacidad.

1.2 Objetivos

Los objetivos principales de esta aplicación, por una parte, engloban la interactividad del usuario con dicha aplicación y por otra, la obtención de resultados, posteriores al uso de la misma, que favorezcan una mejora en el paciente.

A continuación se enumeran los incluidos en ambas partes:

- Realización de cuatro mandos, con diferentes niveles de dificultad de manejo, dependiendo del grado de discapacidad del usuario; que permitan el uso por control remoto de un dispositivo Sphero.
- Realización de un menú con opciones avanzadas, que permitan configurar la aplicación particularizando al usuario con discapacidad que vaya a utilizarla.
- Implementar mecanismos que permitan al usuario tener el control del dispositivo Sphero en todo momento, sin que se vea comprometida su discapacidad.
- Realizar un sistema, por el cual, se consiga registrar todo el comportamiento del usuario, durante el tiempo de uso de la aplicación. Con el fin de llevar un seguimiento del mismo; evaluar si existen cambios en él; determinar si son necesarios algunos cambios en la aplicación que favorezcan una mejoría en cuanto al uso y consecución de resultados en el paciente, etc.

Capitulo 2: Sistema Operativo Android

En el capítulo se va a presentar el sistema operativo para el que ha sido diseñado este proyecto. Se tratará de realizar un acercamiento al lector, para que pueda llegar a comprender sin problemas el diseño e implementación de la aplicación desarrollada.

2.1 Introducción a Android

Android es el sistema operativo de Google orientado a dispositivos móviles, fue lanzado al mercado en su versión 1.0 en octubre de 2008 y está basado en una versión modificada del kernel de Linux 2.6, el cual utiliza para controlar servicios del núcleo del sistema como pueden ser la seguridad, gestión de memoria o gestión de procesos.
Actualmente, se puede encontrar presente en teléfonos móviles, PDAs, Tablets e incluso en Netbooks.

Es una plataforma de código abierto distribuida bajo la licencia Apache 2.0, por lo que su distribución es libre y posibilita el acceso y modificación de su código fuente.
Inicialmente fue desarrollado por Google, para más tarde unirse a la Open Handset Alliance (de la cual, Google también forma parte) que está integrada por T-Mobile, Intel, Samsung, HTC o Nvidia entre otros. Sin embargo, Google ha sido la compañía que ha publicado la mayor parte del código fuente bajo la licencia Apache.

En cuanto al ámbito del desarrollo de software para esta plataforma, a los programadores se les proporciona de manera gratuita el SDK y un plugin que se integra dentro del entorno de desarrollo de Eclipse, donde se incluyen todas las APIs necesarias además de un potente emulador integrado para facilitar las pruebas de la aplicación y que nos ofrece una gran cantidad posibilidades.

Sobre el campo de acción que tiene Android, hay que destacar que Google, siempre ha proclamado, que su objetivo con la plataforma no es que se convierta simplemente en un sistema operativo, lo que pretenden con Android, es reunir todos los elementos necesarios para que los desarrolladores tengan acceso a todas y cada una de las funciones que ofrece un dispositivo móvil de manera sencilla, es decir, se quiere estandarizar el desarrollo de aplicaciones para dispositivos móviles, con las ventajas que ello conlleva.

2.1.1 Historia de Android

Fue desarrollado por Android Inc., una pequeña compañía de California, la cual contaba con un gran equipo de gran experiencia en el diseño de aplicaciones, plataformas web, etc.

En 2005 fue comprada por Google, unos pocos meses después empezaron a surgir las primeras demos no oficiales y fotos de los primeros prototipos, pero dicha información no salió a la luz hasta el 5 de Noviembre de 2007, cuando dicha empresa se popularizó, gracias a la unión al proyecto de Open Handset Alliance, un consorcio formado actualmente por 84 empresas de desarrollo hardware, software y telecomunicaciones, que decidieron promocionar el software libre. Google liberó la mayoría del código de Android bajo la licencia Apache, una licencia libre y de código abierto.

El lanzamiento inicial del Android Software Development Kit, apareció en noviembre de 2007, oficialmente se inició el día 22 de octubre de 2008, en Estados Unidos, cuando fue lanzado el primer teléfono con Android a bordo, el G1 de T-Mobile, cuya interfaz de usuario

fue desarrollada con ayuda de The Astonishing Tribe. La cual, contaba con una ventana de notificación desplegable, widgets en la pantalla de inicio y por supuesto Android Market.

2.1.2 Evolución de Android

Android ha vivido una evolución radical, tanto en el sistema operativo mismo como en el ecosistema de la telefonía móvil, desde su nacimiento y desde su compra, por parte de Google, hasta el día de hoy. En este especial, se describirá su recorrido desde que Android fue producto Beta en noviembre del 2007. Sin embargo, no fue hasta casi un año después, en septiembre del 2008 cuando fuera lanzada la primera versión comercial y completa de Android, la versión Android 1.0 bajo el nombre en clave de Apple Pie.

Android actualmente es el sistema operativo móvil líder, tanto en cuota de mercado, como en innovación, y a falta de conocer la nueva versión Android 4.3, se va a exponer mediante un resumen de las versiones desarrolladas, que ayudarán a la comprensión de cómo poco a poco, ha llegado a ser un referente en cuanto a modelo de sistema operativo.

Android Beta

Android Beta se hizo público el 5 de noviembre del 2007. Esta primera versión era más un concepto que un producto acabado, que en cada nueva versión del SDK, había numerosos cambios y radicales modificaciones en su interfaz, funcionamiento y accesos directos. Realmente el lanzamiento más destacado de Android Beta fue el establecimiento del propio núcleo del sistema operativo.

Android 1.0 Apple Pie

Android 1.0 Apple Pie fue la primera versión de Android comercial lanzada. Supuso un gran impulso en el mercado y marcó el antes y el después en cuanto al ecosistema de telefonía móvil que, en ese momento era dominado por Symbian, junto el nuevo sistema operativo iOS que se iba abriendo camino junto con BlackBerry y Windows Mobile.

Ésta fue la versión de Android con un mayor número de novedades e incorporaciones de todas las vividas en la evolución y crecimiento de Android, pues sentaba las bases para el comienzo de un nuevo sistema operativo a nivel comercial.

Las novedades e incorporaciones más destacadas de Android 1.0 fueron:

- Incorporación de un mercado para compra y descarga de aplicaciones bajo el nombre de Android Market.
- Un navegador web con soporte multiples ventanas y capaz de abrir páginas web en HTML y XHTML.
- Soporte básico para cámara de fotos.
- Posibilidad de crear carpetas e introducir iconos de aplicaciones en ellas desde el escritorio.

- Acceder a servidores de correo electrónico por web soportando los protocolos POP3, IMAP4 y SMTP.
- Sincronización con los productos de Google: Gmail, Google Calendar y Google Contacts.
- Incorporación de productos de Google: GTalk, Google Maps, YouTube, Google Sync y Google Search.
- Marcación por voz.
- Soporte para fondos de pantalla y Widgets.
- Conectividad para WiFi y Bluetooth.

Android 1.1 Banana Bread

Android 1.1 Banana Bread, fue una pequeña actualización publicada el 9 de febrero de 2009, además, esta actualización resolvía pequeños errores detectados, mejorando y cambiando la API y añadiendo una serie de nuevas características.

Nuevas características:

- Añadidos detalles y reseñas sobre lugares y negocios en Google Maps.
- Cambio en la pantalla en llamada así como nueva pantalla en caso de uso de manos libres y posibilidad de mostrar-ocultar el teclado numérico.
- Posibilidad de guardar archivos adjuntos en los mensajes.

Android 1.5 Cupcake

Cupcake, para muchos fue la primera versión Android de la que se hizo uso en España. Su lanzamiento oficial fue el 30 de abril de 2009. Esta versión estaba basada en el núcleo de Linux 2.6.27 y el tema de escritorio incorporado, sería incorporado también en futuras versiones de Android.

Esta fue una gran actualización e incorporó un gran número de novedades e incorporaciones a Android como fueron:

- Habilidad de subir fotos a Picasa y vídeos a YouTube
- Posibilidad de incorporar teclados de terceros con predicción de textos y palabras personalizadas de los usuarios.
- Posibilidad de grabación y reproducción de vídeos en formatos de MPEG-4 y 3GP.
- Soporte para widgets en el escritorio de diferentes tamaños, con animaciones y con refresco automático.
- Incorporada la posibilidad de copiar y pegar en el navegador web.
- Autosincronización y soporte para Bluetooth estéreo.
- Avatares y fotos de contactos para ser mostrados en la sección de contactos “favoritos”.
- Añadido el efecto animado durante el inicio.
- Opción de autorotación según la posición del móvil.
- Transición de pantallas animadas.

- Fecha y hora mostradas en el registro de llamadas así como posibilidad de acceder al contacto seleccionando su tarjeta desde el registro de llamadas.

Android 1.6 Donut

Android 1.6 Donut fue otra gran actualización liberada el 15 de septiembre de 2009. Esta actualización vino repleta de novedades y mejoras sobre sus versiones anteriores:

- Considerable mejora en la búsqueda por entrada de texto y voz incluyendo en la indexación del buscador el historial de favoritos, contactos y web.
- Se mejora la búsqueda en Android Market y se incorporan capturas de imágenes de las aplicaciones.
- Motor multi-lenguaje para síntesis de voz y permitiendo que aplicaciones de terceros puedan hacer uso de ellas para poder usarla.
- Mejora en la galería de imágenes, cámara y cámara de vídeo así como acceso directo a las diferentes funciones como grabación o captura, incorporación de mejoras para la cámara e incremento de velocidad de la misma.
- Posibilidad de seleccionar y eliminar varias fotos de la galería de una sola vez.
- Framework de gestos ampliado y una nueva herramienta de desarrollo GestureBuilder.
- Soporte para CDMA/EVDO, 802.1x, VPNs y un motor text-to-speech.
- Soporte para resoluciones de pantalla WVGA.

Android 2.0/2.1 Eclair

El 26 de octubre de 2009 se lanzó otra gran actualización para Android en la que, tanto por sus mejoras, como por las incorporaciones de fabricantes que apostaron por Android lanzando terminales con este sistema operativo, se consiguió un gran éxito en ventas.

Si se atiende a las fechas de lanzamiento, a pesar de las grandes mejoras que se incorporaban en las nuevas versiones de Android, se creaba un problema y era el de fragmentación. Donut todavía estaba llegando a muchos teléfonos mientras que otros continuaban con Cupcake sin fecha prevista para actualizar a Donut, sin embargo se presentó Eclair donde los cambios en Android eran radicales tanto en su arquitectura como en su interfaz y uso:

- Con Eclair fue posible agregar múltiples cuentas en el teléfono para sincronización de correo y contactos.
- Creación de una bandeja combinada para buscar correo desde múltiples cuentas en la página.
- Soporte Bluetooth 2.1 e incorporada la posibilidad de transferir y compartir archivos a través de este protocolo.
- Posibilidad de hacer click sobre el avatar o foto de un contacto para llamarlo, enviar un SMS o un correo directamente.

- Habilidad de eliminar SMS y MMS más antiguos en caso de haber llenado la memoria de esta mensajería e indexar estos mensajes en el buscador.
- Grandes mejoras en la cámara como soporte de flash, zoom digital, modo escena, balance de blancos, efecto de colores y enfoque macro.
- Mejora en el teclado virtual aumentando su velocidad y añadiendo un diccionario inteligente capaz de incorporar palabras nuevas así como mostrar los nombres de nuestros contactos como sugerencia.
- Gran mejora del navegador de Android incorporando imágenes en miniatura de los favoritos, hacer zoom con toque-doble, soporte para HTML5 y soporte de geoocalización de usuario para mostrar información relacionada con su proximidad.
- Mejora en el calendario mostrando el estado de cada invitado y posibilidad de invitar a nuevos.
- Mejora en la velocidad y proceso en el hardware y una GUI renovada.
- Incorporación de Google Maps 3.1.2
- Soporte para más tamaños y resoluciones de pantallas.
- Incorporación de fondos de pantalla animados capaces de interactuar con los gestos del usuario.
- Soporte para Exchange.

Android 2.2.X Froyo

Android 2.2 Froyo, al igual que Android 2.0 incorporó un gran número de novedades y mejoras en el sistema operativo Android orientado tanto a la mejora en cuanto a experiencia del usuario y mejora de velocidad, características, opciones y rendimiento de los teléfonos con Android.

Esta versión fue lanzada el 20 de mayo de 2010 en la segunda keynote del Google I/O de ese mismo año y presentaba las siguientes novedades y características sobre las versiones anteriores:

- Optimizaciones en velocidad, memoria y rendimiento del sistema.
- Mejoras en el rendimiento de las aplicaciones gracias al nuevo proceso rendimiento de compilación Just-in-time (JIT)
- Posibilidad de poder ver imágenes apiladas en la galería mediante un gesto de zoom.
- Soporte para Adobe Flash
- Soporte para contraseñas numéricas y alfanuméricas
- Integración del motor de JavaScript V8 de Chrome en el navegador
- Compartir internet con otros dispositivos mediante anclaje de red por USB y Wi-Fi hotspot.
- Soporte para pantallas de gran resolución.
- Posibilidad de instalar aplicaciones o trasladarlas a la tarjeta SD.
- Soporte para el servicio Android Cloud to Device Messaging (C2DM) haciendo posible la recepción de notificaciones push.

- Posibilidad de subir archivos mediante el navegador.
- Actualización de Android Market con nuevas características y actualizaciones automáticas.
- Cambio rápido entre múltiples lenguajes de teclado y diccionario
- Intercambio de contactos mediante Bluetooth.
- Mejora de la compatibilidad con Microsoft Exchange, se añaden nuevas políticas de seguridad, sincronización de calendario, y borrado remoto.

Android 2.3.x Gingerbread

El 6 de diciembre de 2010 se presentó la nueva versión de Android con la Gingerbread o 2.3. Gingerbread es la que más subversiones ha contado en todos los lanzamientos de Android, tanto con pequeñas mejoras o corrección de errores, como grandes actualizaciones desde la 2.3.3 hasta la 2.3.7.

Características:

- Nuevo gestor de descargas. Los usuarios acceden de una manera rápida y sencilla a cualquier archivo que haya descargado a través del navegador, correo u otra aplicación.
- Soporte para reproducción de video por el codec de Google para vídeo WebM/VP8 y codec de audio por AAC.
- Soporte nativo para más sensores (tales como giroscopio y barómetro)
- Mejoras en la interfaz de usuario con mejoras de la velocidad y mayor agilidad.
- Soporte para tamaños y resoluciones de pantalla extra-grandes (WXGA y superiores).
- Mejoras en el audio, representación gráfica y entrada para el desarrollo de juegos.
- Cambio desde YAFFS a ext4 en nuevos terminales.
- Mejoras en la administración de la energía mejorando la autonomía del terminal.
- Soporte para múltiples cámaras, por ejemplo cámara frontal permitiendo la videoconferencia.
- Soporte para Near Field Communication (NFC)
- Mejoras en la opción de copiar y pegar al permitir seleccionar una palabra al mantener presionado sobre ella.
- Soporte nativo para SIP y telefonía por internet VoIP.
- Mayor velocidad y precisión en la entrada de texto del teclado virtual con sugerencias de palabras y posibilidad de insertar palabras por voz.
- Mejoras en el sonido y posibilidad de añadir ecualizador y mejoría de graves.

Android 3.x Honeycomb

El Android 3.X, también llamado Honeycomb, fue un Android específico para tablets, aunque más adelante tanto el Android para smartphones, como para tablets, se fusionarían con la versión 4.0. Fue lanzado el 22 de febrero de 2011 y contó con 6 subversiones con diferentes cambios y solución de errores.

Entre las características de Android 3.X Honeycomb se encontraban:

- Soporte e interfaz orientada para su uso en tabletas.
- Una barra de sistema, con acceso rápido a notificaciones, estados y botones de navegación que se encontraba en la parte inferior de la pantalla.
- Aceleración de hardware.
- Soporte de videoconferencia para Google Talk
- Soporte para procesadores multi-nucleo.
- Multitarea simplificada
- Un teclado diseñado para su uso en pantallas grandes.
- Mejoras en el uso de HTTPS con Server Name Indication (SNI).
- Posibilidad de encriptación de los datos de usuario.
- Filesystem in Userspace (FUSE; kernel module).
- Nueva interfaz para escritorio, visualización de contactos, galería y cámara de fotos.
- Autocompletado de texto, nueva forma de visualizar pestañas y posibilidad de navegación de incógnito en el navegador.

El 10 de Mayo del 2011 se lanzaría una actualización con la versión 3.1 que añadía mejoras en Honeycomb también orientadas exclusivamente a las tabletas como:

- Mejora y suavización de la interfaz de usuario.
- Posibilidad de conectar otros dispositivos vía USB como teclados, almacenamientos externos, Joysticks o ratones.
- Lista expandida de aplicaciones abiertas recientemente.
- Widgets redimensionables.
- Soporte de audio para el formato FLAC.
- Bloqueo de conectividad WiFi de alto rendimiento permitiendo continuar con la conexión aunque la pantalla esté bloqueada.
- Soporte proxy para cada uno de los puntos WiFi al que esté conectado.

Android 3.2 saldría poco después con nuevas mejoras e incorporando nuevas características el 15 de julio del 2011:

- Mejoras de soporte de hardware ampliando la posibilidad de instalar el sistema operativo en tabletas de otros fabricantes o arquitecturas.
- Añadía la capacidad para que las aplicaciones pudieran acceder a la tarjeta SD con posibilidad de sincronización de datos o archivos que en ella se encontrasen.
- Compatibilidad para aplicaciones que no fueron diseñadas especialmente para tabletas.
- Nuevas funciones de soporte de pantalla y facilidad a los desarrolladores para simular la pantalla y diseñar aplicaciones para tablets.

Android 4.0.x Ice Cream Sandwich

El Ice Cream Sandwich fue otra enorme actualización y cambio de look de Android. Fue publicado el 16 de octubre del 2011 y el código fuente publicado el 14 de noviembre de

2011 y se incorporaron varias mejoras que ya se vieron en Honeycomb, pero esta vez, no solo para tablets, sino también para teléfonos móviles. Junto a su lanzamiento, también vio la luz el tercer terminal Nexus de Google y de nuevo fue con Samsung con el Galaxy Nexus, como primer terminal en llevar esta nueva versión de Android.

Novedades:

- Los botones táctiles que aparecieron en Honeycomb se incorporaron también para smartphones.
- Separación de Widgets y Aplicaciones en el listado de aplicaciones.
- Creación de carpetas unificando diferentes aplicaciones en un mismo espacio del escritorio.
- Galería de imágenes rediseñada y segmentada por fotos, localizaciones y personas.
- Soporte para el formato de imagen de Google, WebP.
- Aceleración de la interfaz de usuario mediante hardware.
- WiFi Direct
- Buzón de voz mejorado con la opción de acelerar o retrasar los mensajes del buzón de voz (para Google Voice)
- Añadida la funcionalidad de pinch-to-zoom en el calendario nativo.
- Grabación de video a 1080p para dispositivos con Android de serie
- Incorporado Android Beam que permite transmitir de un dispositivo a otro con NFC archivos, imágenes o vídeos de YouTube.
- Actualización de Contactos con integración social como estados o imágenes en alta resolución.
- Nueva tipografía para la interfaz de usuario, Roboto.
- Editor de fotos integrado.
- 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.
- Captura de pantalla nativa sin hacer uso de aplicaciones de terceros o rootear el terminal.
- Mejora de predicción y corrección de texto del teclado.
- Acceso a aplicaciones o widgets desde la misma pantalla de bloqueo del teléfono.
- Cierre manual de aplicaciones que se estén ejecutando en un segundo plano.
- Mejora en la función de copiar y pegar texto.
- Mejora en reconocimiento de voz y dictado de texto.
- Reconocimiento y desbloqueo facial.
- Sustitución del navegador nativo por Google Chrome así como posibilidad de sincronizar favoritos con el Chrome de escritorio.
- Control y monitorización de datos consumidos así como posibilidad de bloqueo de los mismos una vez alcanzado el límite.

Android 4.1 Jelly Bean

Android 4.1 (Jelly Bean) fue presentado en la Google I/O el 27 de junio de 2012 y su principal objetivo fue impulsar y mejorar la funcionalidad, fluidez y rendimiento de la interfaz de usuario de Android gracias al "Proyecto Butter" en el cual se unían la anticipación táctil, triple buffer, latencia vsync extendida y un aumento de la velocidad de cuadros de 60 frames por segundo, creando una increíble mejora en la fluidez de la interfaz en comparación a Ice Cream Sandwich.

El primer dispositivo en correr Jelly Bean, fue el Nexus 7, el primer tablet en la familia Nexus y fabricado por Asus que lanzado el 13 de julio de 2012.

Entre sus novedades se encontraban las siguientes:

- Exponencial mejora en la fluidez de la interfaz de usuario.
- Mejoras en la accesibilidad.
- Widgets expandibles.
- Soporte de texto bidireccional y traducción de Android en otros idiomas.
- Capacidad de desactivar notificaciones de una/s determinada/s aplicación/es
- Reorganización y redimensión automática de Widgets y accesos directos en el escritorio.
- Mapas de teclado
- Dictado de voz sin conexión.
- Mejoras en la aplicación de búsqueda de Google y en la cámara.
- Incorporación del Bluetooth como forma de transferencia de archivos con Android Beam.
- Fotos de contactos en alta resolución para el directorio de contactos a través de Google+.
- Google Now
- Audio multicanal y Audio USB.
- Encadenamiento de audio (reproducción sin pausas)
- Posibilidad de añadir widgets en otros escritorios o launchers sin necesidad de root.

Android 4.2 Jelly Bean

La presentación de Android 4.2 Jelly Bean, tenía que haberse realizado en un evento el día 9 de octubre de 2012 en Nueva York pero el azote del Huracán Sandy provocó que este fuese cancelado ante la decepción de medio mundo.

Sin embargo, lejos de retrasar el evento o emitirlo en vivo, Google anunció la nueva versión mediante un comunicado de prensa bajo el eslogan "A new flavor of Jelly Bean".

Las novedades que se incorporaron en la última versión de Android disponible fueron:

- Creación de fotos panorámicas en 360º con Photo Sphere.

- Teclado con escritura deslizable (estilo Swype)
- Cambios en la pantalla de bloqueo como soporte para widgets y posibilidad de deslizar para acceder directamente a la cámara.
- VPN siempre activa.
- Confirmación a la hora de enviar SMS Premium.
- Nuevo método de seguridad con SELinux.
- Acceso directo a notificaciones desde barra de notificaciones o deslizando esta con dos dedos hacia abajo.
- Protector de pantalla "Daydream" cuando el terminal está conectado a un Dock o inactivo.
- Múltiples cuentas de acceso (para tablets)
- Soporte para pantallas inalámbricas (Soporte Miracast)
- Nuevo reloj con reloj mundial, cronómetro y cuenta atrás.
- Nuevos gestos y accesibilidad para invidentes.
- Integración de la misma interfaz independientemente del dispositivo o del tamaño de la pantalla.
- Información de las notificaciones extensibles sin necesidad de entrar en la aplicación para visualizar las novedades.

El pasado 11 de febrero de 2013 se hizo pública la actualización 4.2.2 que incluía una serie de mejoras y correcciones:

- Solucionado un error en streaming de audio por Bluetooth.
- Solucionado un error con el flash de la cámara.
- Acceso directo a apagar y encender bluetooth y WiFi manteniendo presionado sobre su icono en la barra de notificaciones – Quick Settings.
- Nueva notificación de descarga mostando el porcentaje descargado y descargas activas.
- Nuevos sonidos para batería baja y carga inalámbrica.
- Carga más rápida de la galería de imágenes
- Se elimina la opción de mostrar todas las llamadas en la lista de llamadas.
- Mejoras generalizadas y mayor estabilidad.

2.1.3 Android en el mercado

A finales de agosto de 2013, Google anunció que ya se habían activado más de mil millones de dispositivos Android en todo el mundo, y ahora el equipo de Android actualiza los datos de distribución de versiones Android.

Para septiembre de 2013 hay de nuevo novedades a la hora de contabilizar las versiones. Desde ahora, dejan de contabilizar las versiones anteriores a Android 2.2 porque no son compatibles con la aplicación de Google Play. A partir de abril dijeron que sólo iban a contar los dispositivos que acceden a Google Play, pero dejaron un margen de tiempo con las versiones anteriores a Android 2.2 y para esos dispositivos seguían teniendo en cuenta los dispositivos que se identificaban en los servidores de Google.

En agosto de 2013, Android 1.6 (Donut) tenía el 0,1%, Android 2.1 (Eclair) el 1,2% y el resto de versiones antiguas desde hace tiempo no aparecían en las gráficas por tener menos de un 0,1% de cuota.

Version	Codename	API	Distribution
2.2	Froyo	8	2.4%
2.3.3 - 2.3.7	Gingerbread	10	30.7%
3.2	Honeycomb	13	0.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	21.7%
4.1.x	Jelly Bean	16	36.6%
4.2.x		17	8.5%

Figura2.1: Cuota de mercado actual (fuente: Xatakaandroid.com)

En septiembre de 2013, han recogido los datos durante varios días, terminando el 4 de septiembre. La versión más usada es Android 4.1 (Jelly Bean) con el 36,6%, seguida de Android 2.3 (Gingebread) con el 30,7%.

De forma global el 45,1% de dispositivos Android llevan Jelly Bean, el 36,6% la versión 4.1 y el 8,5% la versión 4.2. La versión más reciente de Android, la 4.3, todavía no aparece en los datos por no llegar al 0,1%.

Ice Cream Sandwich es la tercera versión más usada con el 21,7%. Froyo tiene el 2,4% de penetración y Honeycomb el 0,1%. Sólo las versiones de Jelly Bean han ganado cuota de mercado, el resto van perdido mes a mes.

2.2.1 Arquitectura

Android es un sistema diseñado por capas. Utiliza el kernel de Linux 2.6 que le da acceso a la parte hardware de los dispositivos a la par que le permite ser compatible con muchos de los drivers creados para Linux.
Esta arquitectura está perfectamente plasmada en el diagrama que de la siguiente figura:

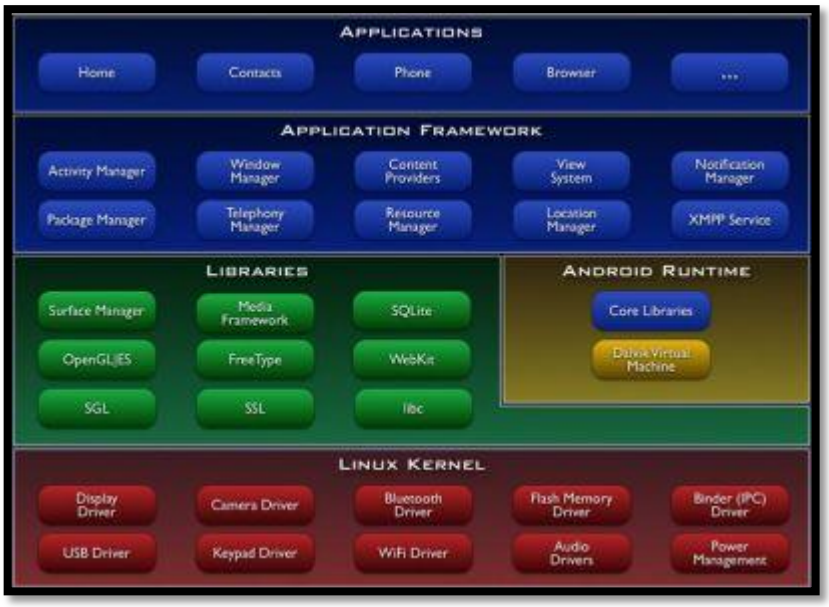


Figura2.2: Arquitectura del sistema operativo Android

- **Aplicaciones**

Todas las aplicaciones, tanto las incluidas en el propio Android como las creadas por desarrolladores, están escritas en lenguaje Java y pueden estar compuestas por 5 bloques de los que hablaremos más adelante: Activity, Intent, Broadcast, Services y Content Providers, pero no tienen que aparecer todos ellos por obligación, simplemente se utilizarán los necesarios para llevar a cabo los objetivos para los que fue diseñada dicha aplicación. Entre las aplicaciones ya instaladas en el sistema se pueden encontrar un cliente de correo, navegador web, gestor de contactos, mapas o calendario entre otras.

- **Framework**

El marco de aplicaciones da acceso completo a los programadores a las mismas APIs utilizadas por las aplicaciones básicas. La arquitectura está diseñada para que la reutilización de componentes sea sencilla, ya que cualquier aplicación puede dar un perfil público a sus datos o capacidades para que otra aplicación haga uso de esas cualidades. Todo esto es posible, partiendo de la base de que las normas de seguridad impuestas por el framework no se vean comprometidas, de esta manera se consigue incorporar un importante grado de reutilización del código. En la figura **Figura2.2** se pueden ver algunas de las librerías más importantes de esta capa.

- **Librerías**

El sistema incluye un conjunto de librerías en C y C++ que proporcionan la mayor parte de las funcionalidades presentes y que son utilizadas por varios de los componentes del sistema Android. Estas capacidades son expuestas a los desarrolladores a través del framework descrito anteriormente para que puedan ser utilizadas.

- **Android Runtime**

Siguiendo el diagrama de la arquitectura mostrado en la **Figura2.2**, al mismo nivel que las librerías de Android, se sitúa el Android runtime o entorno de ejecución, compuesto por librerías que proporcionan la mayor parte de las funcionalidades de Java además de incluir la máquina virtual. Al ejecutarse, cada aplicación crea su propio proceso con su propia instancia de la máquina virtual Dalvik, que ha sido escrita de manera que un mismo dispositivo, pueda lanzar múltiples instancias de dicha máquina virtual de forma eficiente.

- **Kernel de Linux**

La capa más cercana al hardware del dispositivo corresponde al núcleo de Android que está basado en el kernel de Linux 2.6. Android utiliza este kernel como una abstracción del hardware disponible en cada terminal conteniendo los drivers necesarios para utilizar cualquier parte de este hardware a través de las llamadas correspondientes. Android también depende de este kernel para gestionar los servicios base del sistema como pueden ser la seguridad, gestión de memoria, de procesos, la pila de red y el modelo de driver.

Como apunte, cabe decir que la versión 2.6 del kernel de Linux fue elegida por diversas razones. Una de ellas fue su naturaleza de código abierto, ya que utilizar cualquier otra opción comercial no habría encajado con el uso de la licencia Apache. Además existen algunos cambios importantes en esta versión como pueden ser:

- Respuesta más ágil
- El kernel puede ser interrumpido (solo en ciertos casos)
- Renovación del kernel en cuanto al manejo de hilos.

2.2.2 Dalvik

Los requisitos hardware inherentes a los dispositivos móviles obligan a que los recursos sean bastantes limitados. Aunque existen estudios que auguran la equiparación de estos aparatos con sus homólogos de escritorio en cuanto a capacidad de cómputo y memoria en un futuro no muy lejano, en la actualidad existen ciertas restricciones relacionadas con estos aspectos. La memoria RAM se encuentra en torno a los 192 MB y la memoria interna disponible para la ejecución de aplicaciones es bastante limitada.

Dalvik es una máquina virtual intérprete que ejecuta archivos en el formato Dalvik Executable (*.dex), un formato optimizado para el almacenamiento eficiente y ejecución mapeable en memoria. Su objetivo fundamental, es el mismo que cualquier máquina virtual, permite que el código sea compilado a un bytecode independiente de la máquina en la que se va a ejecutar, y la máquina virtual interpreta este bytecode a la hora de ejecutar el programa. Una de las razones por las cuáles no se optó por utilizar la máquina virtual de Java es la necesidad de optimizar al máximo los recursos y enfocar el funcionamiento de los programas hacia un entorno dónde los recursos de memoria, procesador y almacenamiento son escasos.

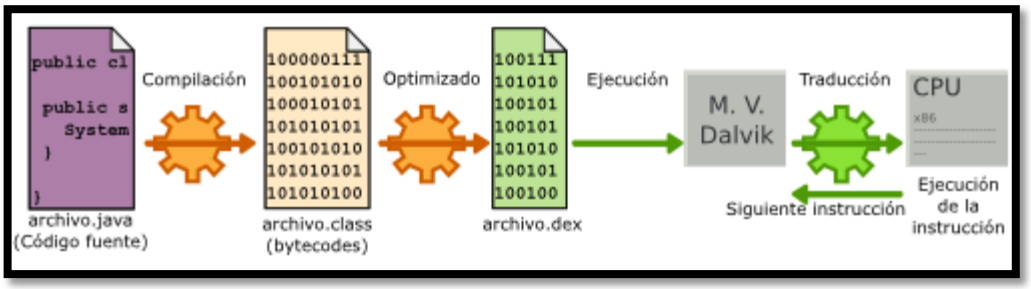


Figura 2.3: Mecanismo de compilación y traducción de una instrucción

Dalvik está basada en registros y puede ejecutar clases compiladas por un compilador Java y que posteriormente han sido convertidas al formato nativo usando la herramienta "dx". El hecho de que corra sobre un kernel Linux le permite delegar las tareas relacionadas con la gestión de hilos y memoria a bajo nivel.

Otra característica importante de Dalvik es que ha sido optimizada para que múltiples instancias de ella puedan funcionar al mismo tiempo con un impacto muy bajo en el rendimiento de la memoria del dispositivo. El objetivo de esto, es proteger a las aplicaciones, de forma que el cierre o fallo inesperado de alguna de ellas no afecte de ninguna forma a las demás.

- ## Diferencia con la Máquina Virtual de Java

La máquina virtual de Java, que podemos encontrar en casi todas las PC's actuales, se basa en el uso de las pilas. De modo contrario, Dalvik utiliza los registros, ya que los teléfonos móviles están optimizados para la ejecución basada en los mismos.

Aunque se utiliza el lenguaje Java para programar las aplicaciones Android, el bytecode de Java no es ejecutable en un sistema Android. De igual forma, las librerías Java que utiliza Android son ligeramente distintas a las utilizadas en Java Standard Edition (Java SE) o en Java Mobile Edition (Java ME), guardando también características en común.

- **Dalvik y la optimización en aplicaciones Android**

El uso de Dalvik permite reducir bastante el tamaño del programa buscando información duplicada en las diversas clases y reutilizándola.

Lo que se suele llamar en Java como "recolectar basura", que libera el espacio en memoria de objetos que ya no son utilizados en nuestros programas, ha sido perfeccionada en Android con el fin de mantener siempre libre la máxima memoria posible.

De igual forma, el hecho de que Android haga un uso extenso del lenguaje XML para definir las interfaces gráficas y otros elementos, implica que estos archivos deben ser linkados a la hora de compilar y para que su conversión a bytecode pueda mejorar el rendimiento de las aplicaciones.

2.2.3 Componentes

En este apartado se hablará sobre los componentes o bloques que pueden encontrarse en cualquier aplicación de Android. No es obligatorio implementarlos todos, por lo que podría decirse que las aplicaciones están compuestas por una combinación libre de los bloques o componentes descritos a continuación. Todos estos componentes deben ser declarados de forma explícita en el fichero *AndroidManifest.xml*, donde se encuentran definidos otros datos importantes como los permisos, siendo un fichero básico en cualquier aplicación.

- **Activity**

Las actividades representan el componente principal de la interfaz gráfica de una aplicación Android. Se puede pensar en una actividad como el elemento análogo a una ventana en cualquier otro lenguaje visual. Una actividad refleja una determinada actividad llevada a cabo por una aplicación, lleva asociada la interfaz de usuario representada por la clase View y sus derivados. Se implementa mediante la clase del mismo nombre Activity.

La mayoría de aplicaciones están compuestas de varias pantallas con diferentes objetivos, por lo tanto, estarán compuestas de varias actividades. El paso de una pantalla a otra se consigue mediante la inicialización de una nueva actividad, cuando una nueva ventana se abre, la actividad anterior queda en pausa dentro de la pila pudiendo el usuario navegar entre actividades previamente abiertas.

El concepto de Intent se encuentra estrechamente ligado con la inicialización de las actividades. El concepto de intent se puede definir como la intención de realizar una acción (en este caso, comenzar una nueva actividad). Al lanzar un intent, una aplicación puede llamar a otra aplicación distinta que sea capaz de realizar la acción solicitada, como por ejemplo escribir un mensaje de texto, realizar una llamada o abrir una URL en el navegador web.

- ## Service

Los servicios son componentes sin interfaz gráfica que se ejecutan en segundo plano. En concepto, son exactamente iguales a los servicios o demonios presentes en cualquier otro sistema operativo. Los servicios pueden realizar cualquier tipo de acciones, por ejemplo actualizar datos, lanzar notificaciones no intrusivas, o incluso mostrar elementos visuales si se necesita en algún momento la interacción con del usuario para obtener una confirmación.

El ejemplo típico que se da para comprender el concepto de servicio es el reproductor de música que viene incluido en el sistema operativo. Esta aplicación está dotada de una interfaz gráfica (o Activity) donde se permite al usuario elegir entre una lista de canciones, generar listas de reproducción así como el resto de acciones típicas de estos reproductores. Sin embargo, en el momento en que el usuario comienza a reproducir un archivo de audio, puede seguir navegando entre las canciones e incluso comenzar a utilizar otras aplicaciones. Esto se debe a que la reproducción se realizar mediante un servicio y se ejecuta en background. De igual manera que las actividades, se implementa mediante la clase con su mismo nombre.

- ## Broadcast receiver

Un broadcast receiver es un componente destinado a detectar y reaccionar ante determinados mensajes o eventos globales generados por el sistema.

También es posible, mediante el uso de los intents, generar mensajes broadcast en la propia aplicación que estén dirigidos a cualquier Broadcast receiver que se encuentre activo.

Este componente tampoco tiene una interfaz gráfica asociada, pero pueden utilizar el API Notification Manager como se ha visto anteriormente para avisar de manera no intrusiva al usuario del evento que se ha producido a través de la barra de notificaciones presente en el sistema. Al igual que los compontes anteriores, los broadcast receivers, se implementan a través de la clase con su mismo nombre.

- ## Content provider

Un content provider, es el mecanismo que se ha definido en Android para compartir datos entre aplicaciones. Mediante estos componentes es posible compartir determinados datos de nuestra aplicación sin mostrar detalles sobre su almacenamiento interno, su estructura, o su implementación. De la misma forma, una aplicación podrá acceder a los datos de otra a través de los content provider que se hayan definido.

Existe una serie de content providers ya definidos que se encuentran implementados y que permiten compartir todo tipo de datos, información de los contactos, imágenes, video, mensajes de texto o incluso audio. Estos se encuentran dentro del paquete android.provider ofreciendo además la posibilidad de crear content providers propios.

Los content provider, son objetos de la clase ContentProvider, que se encuentra localizada en el paquete android.content. Cada objeto de esta clase lleva asociada una URI única que sirve para identificarlo y a través de la cual el resto de aplicaciones deben acceder a él. Cuando se crean un nuevo content provider en nuestra aplicación, los datos que se vayan a hacer públicos se almacenarán habitualmente como una tabla de una base de datos SQLite donde cada columna se refiere a un tipo de dato y cada fila representa un registro.

2.2.4 Ciclo de vida de una aplicación

Cada aplicación Android es ejecutada en su propio proceso y es el propio sistema operativo el encargado de lanzar y parar estos procesos, gestionar su ejecución y decidir qué hacer en función de los recursos disponibles y de las órdenes dadas por el usuario.

El número de procesos posibles serán tantos como permitan los recursos del dispositivo. El usuario desconoce este comportamiento. Sólo es consciente de que mediante un simple clic pasa de una a otra aplicación y puede volver a cualquiera de ellas en el momento que lo desee. No debe preocuparse sobre cuál es la aplicación que realmente está activa, cuánta memoria está consumiendo, ni si existen o no recursos suficientes para abrir una aplicación adicional.

Cada aplicación, está formada por una o varias pantallas, llamadas también componentes Activity o actividades, debido a que estas son implementadas mediante una extensión de la clase Activity. Esta clase permite formar las interfaces gráficas de usuario, que muestran información en pantalla y responden a las acciones del usuario.

El desarrollador puede controlar en cada momento en qué estado se encuentra una actividad.

2.2.5 Ciclo de vida del componente Activity

Existen tres estados:

- **Activo:** La actividad se encuentra en primer plano (Encima de la pila de tareas) e interactuando con el usuario.
- **Pausado:** La actividad sigue siendo visible para el usuario, pero ha perdido el foco. Por ejemplo que se haya mostrado un cuadro de dialogo delante de la actividad. Se debe guardar el estado de la interfaz y los datos de esta actividad antes de entrar en este estado, ya que se podrían perder si el sistema necesita más recursos de memoria.

Parado: La actividad no es visible para el usuario, queda a disposición del sistema para borrarla de la pila en caso de necesitar memoria.

La clase Activity dispone de métodos que se llaman cada vez que ésta cambia de estado, para permitir realiar tareas como guardar los datos antes de cambiar de estado, y cargar la actividad más rápido la proxima vez que se muestre. A continuación se muestra un diagrama con los distintos estados por los que pasa una actividad:

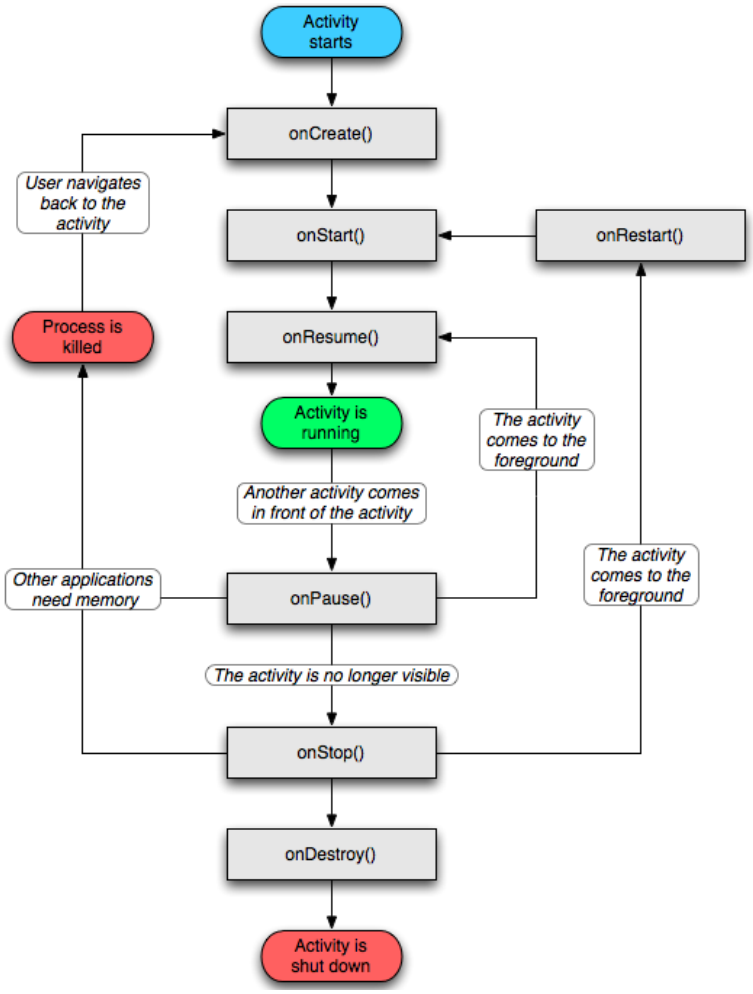


Figura 2.4: Ciclo de vida de un componente Activity

- **onCreate(Bundle savedInstanceState):** Este método se llama al crear la actividad. Siempre se sobrescribe para configurar la vista, crear adaptadores, rellenar los objetos con sus valores etc. Puede recibir como parámetro el estado anterior de la actividad para que se pueda ser restaurada.
- **onPause():** Es llamado justo antes de que se traiga a primer plano otra actividad. Aquí es donde se deben guardar los datos para no perder la información de la actividad si esta es sacada de la pila. Dentro de este método también se suelen parar las tareas pesadas que consuman CPU.
- **onStop():** Es llamado cuando la actividad se va a ocultar durante un largo periodo de tiempo. Si el sistema necesita recursos, puede que este método no sea llamado, por lo que es recomendable guardar los datos en el método onPause().

- **onDestroy:** Último en llamarse antes de destruir la actividad. Puede llamarse a través del método finish() o llamarlo el sistema para conseguir más memoria. Para saber quién lo llamó, se puede usar isFinishing().

2.2.6 Ciclo de vida de un Service

Los servicios se pueden usar de dos formas, dependiendo de cómo sean lanzados, su ciclo será uno u otro.

- Si se lanza con startService() se ejecurará hasta que termine. Los servicios se configuran en el método onCreate() y se liberan en el onDestroy(). Podemos terminar un servicio externamente con Context.stopService() o dentro del mismo servicio con Service.stopSelf() o Service.stopSelfResult().
- Si se lanza con Context.bindService(), se puede interactuar con él mediante una interfaz que el servicio debe exportar. Se terminará el servicio con Context.unbindService().

A continuación se muestra el diagrama con el ciclo de vida de los servicios:

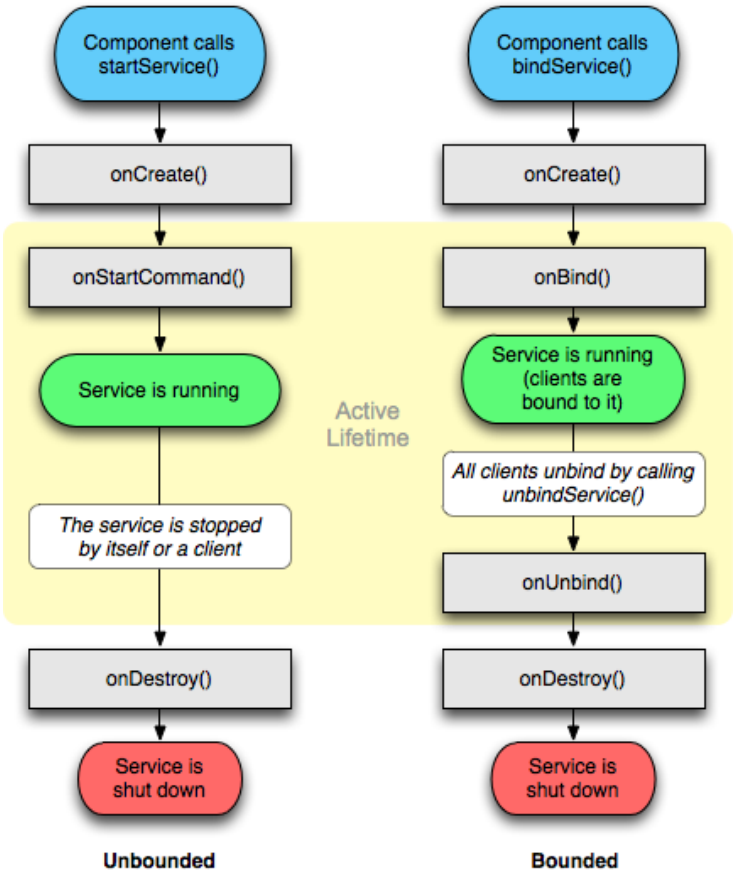


Figura 2.5: Ciclo de vida de un Service

Capitulo 3: Sphero

En el capítulo se va a presentar el dispositivo Sphero empleado en el proyecto. Es una parte fundamental, ya que junto con la aplicación Android, forman una herramienta de uso favorable para los usuarios con discapacidad.

3.1 Introducción a Sphero

Sphero es un dispositivo robótico creado por la compañía Orbotix en 2012. Dicha empresa, posee una filosofía de negocio muy particular, ofrecen al usuario un producto de código abierto, al cual, se le permite reprogramar tanto el hardware (modificación de parámetros en los sensores, límite de velocidad del motor, frecuencia de recepción bluetooth, etc.), como el software; ya que Orbotix ha desarrollado una serie de aplicaciones propietarias y pone a disposición del usuario un SDK, tutoriales, foros y servicio técnico para así, fomentar el desarrollo de software que promueva el uso de Sphero junto a él, siempre sin depender de mandos radiocontrol. Únicamente desde un dispositivo móvil con Bluetooth.

Es controlado por Bluetooth a través de un Smartphone o Tablet bajo sistema operativo Android o iOS. Aunque actualmente, existen proyectos de usuarios en los que se puede interactuar con un Sphero desde una XBOX360, Arduino, PhoneGap, Windows Phone 8, e incluso realizando envío de SMS con un cliente Twilio.

3.1.1 Características físicas de Sphero

- Posee un acelerómetro, el cual funciona, de forma similar al de un dispositivo móvil (Smartphone o Tablet PC), teniendo en cuenta los ejes X,Y,Z:

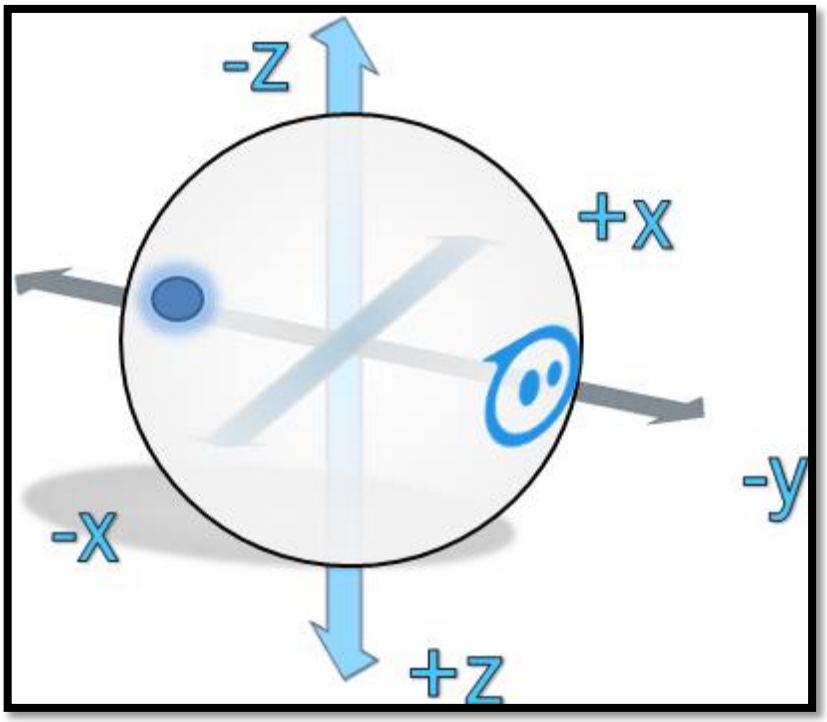


Figura 3.1: Representación de los ejes de dirección

- Posee un giroscopio, el cual permite colocar un Sphero en la posición que el usuario desee y eso va a determinar la dirección de desplazamiento. Cuando se describa la implementación de la aplicación, se apreciará verdaderamente el uso del *LEDTail*, que en la **Figura 3.2** aparece representado por una elipse iluminada.

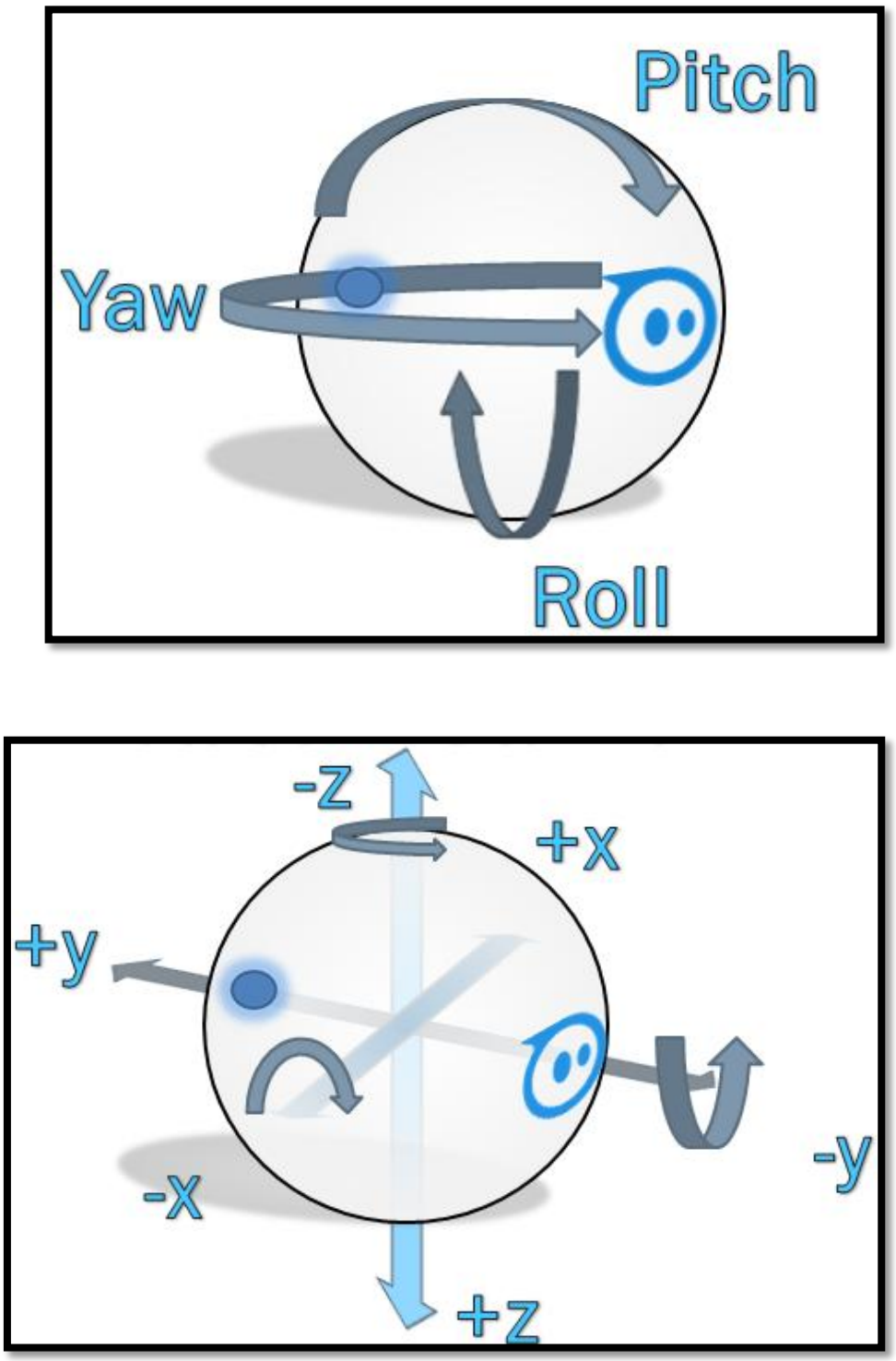


Figura 3.2: Representación del mecanismo de giro del giroscopio

- Posee un LED, al cual el fabricante llama *LEDTail* o LED de la Cola, que permite tener un punto de referencia sobre el que se encuentra un Sphero para realizar giros y desplazamientos. Cualquier mandato que se envíe a un Sphero vendrá dado en grados.

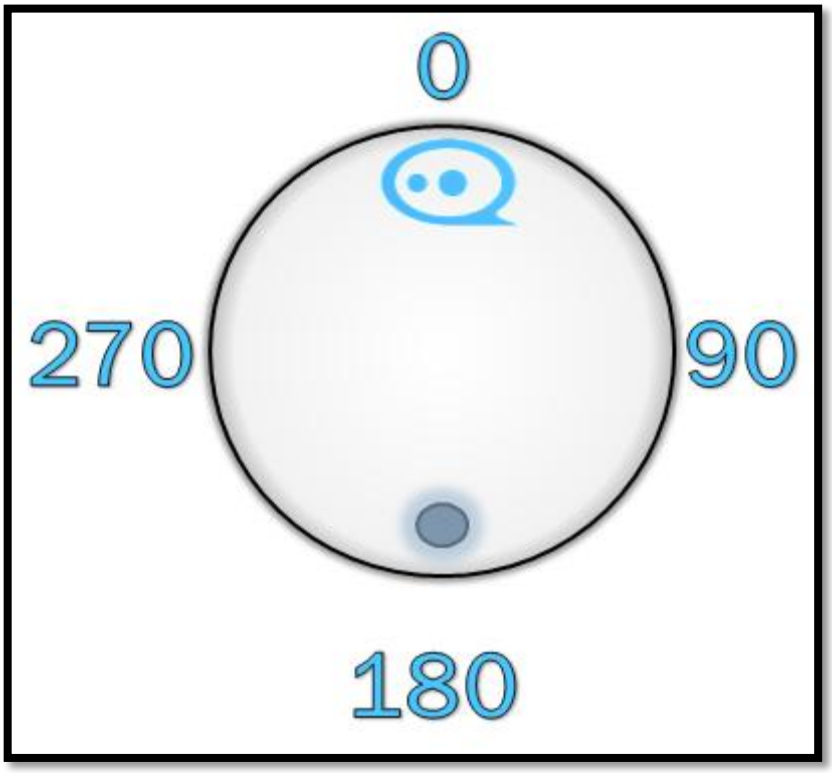


Figura 3.3: Representación de las direcciones en grados respecto a la cabeza

- Posee un sensor que informa de colisiones que se producen en un Sphero, cuando está siendo controlado por un usuario.
- Posee un magnetómetro, con el cual obtiene el índice de inducción electromagnética de una fuente cercana para ser utilizada para cargar su batería interna. Esto lo utiliza, para anular cualquier instrucción que esté llevando a cabo, si se le coloca sobre su base de carga, ya que detecta la fuente de inducción y comienza su estado de carga de batería.
- Posee un sensor de localización, el cual realiza los cálculos necesarios para indicar su posición en todo momento, tomando previamente una posición de referencia.
- Posee en su interior un conjunto de LEDs capaces de proyectar más de 16 millones de colores combinándose.

3.2 Motivos que llevaron a elegir Sphero

Después de un período de investigación, para encontrar un dispositivo destinado a este proyecto, con el que una persona con discapacidad pudiese interactuar y por supuesto, le ayude a obtener una serie de resultados que posibiliten una mejora sensorial; y tras evaluar todas las posibilidades que ofrecían los diferentes dispositivos radio control, Sphero fue la mejor opción ya que era el dispositivo más versátil.

Los otros modelos que se encontraron, para analizarlos y compararlos antes de elegir Sphero, fueron:

- **BERO (Be the Robot).** Dispositivo radio control por Bluetooth, que mediante una serie de cadenas de texto enviadas a dicho dispositivo, son interpretadas para realizar un movimiento u otra acción. El APK que ofrece el fabricante, así como su manual de usuario, son demasiado básicos y ofrece menos opciones de uso que Sphero.

- **Coche radio control Bluetooth.** Es un dispositivo que, en principio podría haber sido válido, pero del que no se obtuvo respuesta por parte del fabricante, cuando se le preguntó por la posibilidad de desarrollar una aplicación integrable en su dispositivo. Lo único que se consiguió saber es que, funcionaba mediante el envío de caracteres al juguete, para realizar los mandatos de movimiento. No existía ninguna API específica para dicho dispositivo.

- **Coche radio control, más completo que el anterior.** De igual forma que en el caso anterior, no se obtuvo respuesta por parte del fabricante, acerca del uso del propio código implementado por cualquier desarrollador. Presentaba varios sensores, que detectaban cercanía de cualquier obstáculo y un motor un poco más potente que el caso anterior.

Capitulo 4: Flurry Analytics

En este capítulo se va a presentar el servicio que ofrece Flurry Analytics, que permite el registro de eventos en su web, para luego tratar los datos de la forma que interese al usuario. Es un servicio que se ha utilizado en este proyecto y aunque al principio del mismo, no se pensó en incorporarlo, finalmente se integró en él, debido a su gran relevancia a la hora de obtener un exhaustivo seguimiento del uso de la aplicación.

4.1 Introducción a Flurry

Flurry es una librería para Android, iPhone y iPad (entre otras plataformas) que permite analizar cómo los usuarios interactúan con una determinada aplicación. Para ello, se distribuyen eventos en los métodos de las actividades que el programador crea conveniente analizar.

Una vez registrado un evento en una aplicación, Flurry ofrece una serie de estadísticas para saber el tipo de dispositivo que ha usado esta funcionalidad, la hora a la que se envió ese evento, la ubicación del usuario, así como cualquier parámetro que aporte información valiosa. Dichos parámetros son perfectamente personalizables, invocando un método llamado logEvent y pasándole una clave y valor, como se muestra a continuación:

FlurryAgent.logEvent(String arg0, Map<String, String> arg1)

```
intent.setClass(LaunchController.this, MJoystick.class);
FlurryAgent.logEvent("Usa mando Joystick", userParams);
startActivity(intent);
```

En este ejemplo puede observarse el registro de un String y pasándole un **Map<String, String>**. Esto se hace para asignar al evento “Usa mando Joystick” una serie de parámetros del usuario:

```
userParams.put("Usuario", nombreAlumno);
userParams.put("Estado de Usuario", "Registrado");
```

Posteriormente, es posible consultar los eventos registrados en https://dev.flurry.com autentificándose con un usuario dado de alta previamente:

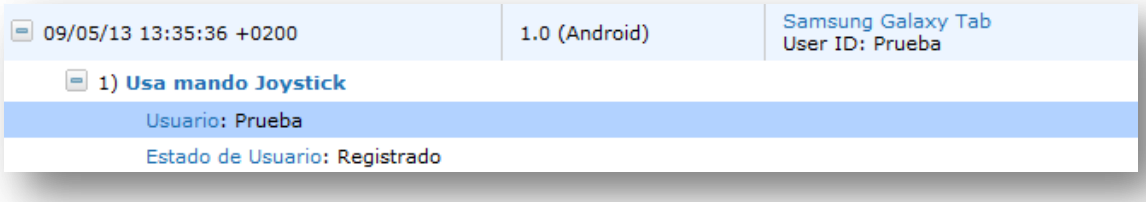


Figura 4.1: Evento registrado en la web de Flurry

4.2 Características principales

Flurry Analytics es un software gratuito orientado al ámbito profesional. Desde su lanzamiento en 2008, Flurry ha añadido continuamente características innovadoras y ha ampliado notablemente su servicio para dar soporte a multitud plataformas bajo diferentes sistemas operativos. Es muy intuitivo y apto para usuarios con un nivel medio o bajo en Android u otro sistema operativo.

Flurry Analytics proporciona una increíble lógica de comercio sobre cómo y dónde la gente está utilizando la aplicación. Pueden identificarse rápidamente a los usuarios que dan más uso a la aplicación y realizar mejoras en ella dependiendo del comportamiento que tengan dichos usuarios. Estos pueden ser agrupados según características clave tales como la demografía, ubicación, preferencias de idioma, y el uso de determinadas funciones en la aplicación.

También cuenta con características avanzadas para otro tipo de usuarios, incluyendo el Funnel Analysis, Segmentación Personalizada, Alertas Flurry.

- **Funnel Analysis**. Proporciona seguimiento de usuarios, ya que ejecutan una serie de pasos definidos, lo que permite ver cómo muchos de los usuarios que iniciaron un proceso dado lo completó. Para aquellos que no completaron el proceso, es registrable en qué paso salían. Toda esta información se puede utilizar para ajustar los aspectos de una aplicación para maximizar el número de personas que llegan y completan el último paso, comúnmente conocida como "la meta" o "conversión". Funnel Analysis de Flurry Analytics, cualquier caso que haya sido instalado en una aplicación se puede utilizar para definir un paso.Hay dos tipos de Funnels, "in-app", centrados en una sóla aplicación y "cross-app", que abarcan dos o más aplicaciones.

  El siguiente ejemplo, muestra un Funnel con el número y porcentaje de usuarios que completaron cada paso:

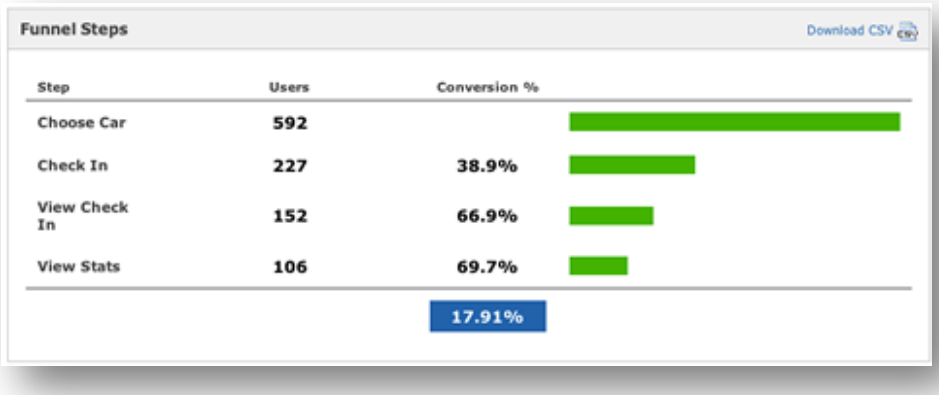


Figura 4.2: Representación de un Funnel Analysis

- **Segmentación Personalizada.** Permite analizar cómo los diferentes grupos de los usuarios de la aplicación varían en su uso y comportamiento. Por ejemplo, si se ha desarrollado un juego para un dispositivo móvil, es posible que se desee entender cómo los usuarios que alcanzaron el nivel 5 en los EE.UU. difieren de los usuarios que alcanzaron el nivel 5, por ejemplo, en Alemania. Se pueden examinar todas las métricas que se ofrecen en Flurry Analytics para cada segmento de forma independiente.

  Uno de los valores fundamentales de la segmentación es el proceso de perfeccionamiento de los grupos personalizados definidos para entender qué conjunto de usuarios es más valioso para el desarrollador enfocándolo al negocio. ¿Quién gasta más dinero dentro de la aplicación? ¿Qué lugar tiene la mayor retención, o la frecuencia de uso? La comprensión de composición de la audiencia permite planificar mejor una próxima campaña de adquisición (ya que es posible saber la clase de los consumidores a la que va a ir destinada y así desarrollar unas características específicas en la aplicación).

- **Alertas Flurry.** Esto permite monitorizar las métricas clave de aplicaciones si se ha iniciado una sesión en el servicio o no. Además de configurar alertas para supervisar cambios, como nuevos picos de usuarios o cuando demasiados usuarios no son capaces de conseguir avanzar de nivel en un videojuego.

  Si el usuario desarrollador, cuenta con un número amplio de aplicaciones, estas alertas, pueden ayudar a mantener un registro de los cambios que más importan, para realizar nuevas versiones de la aplicación.

4.3 Motivos que llevaron a su uso

Al comienzo del proyecto, se contó con el servicio que la API de Sphero incluye, para registrar eventos en una cuenta de SpheroWorld. Mediante la clase *AchievementManager,* los eventos deseados por el desarrollador, pueden ser registrados para, de forma posterior, analizar esos datos.

El problema fue, que una vez utilizados los métodos que la API de Sphero ofrecía, no se registraban de forma adecuada en la cuenta dada de alta. Por lo que, hubo que contactar con los propios desarrolladores de software de Orbotix. Ellos mismos, fueron los que informaron de errores en la web de Sphero World a la hora de registrar los eventos, dicho servicio estaba en reparación, debido a la cantidad de incidencias que habían tenido y propusieron la alternativa de usar Flurry Analytics. Esta, pertenece a una compañía independiente a Orbotix, pero ofrece un servicio más amplio y más robusto, ya que lleva más años desarrollándose, con más versiones en su SDK y con un grupo de usuarios inmensamente más amplio que el de Sphero.

Capitulo 5: Discapacidad

En este capítulo se va a explicar la comunicación, con los trastornos que impiden una comunicación óptima; para dar paso a los tipos de discapacidad, detallando cada tipo. Se concluirá con sistemas de comunicación enlazándolo con los juguetes adaptados y la accesibilidad en dispositivos móviles, ya que todo ello forma parte de este modelo de aplicación.

5.1 La comunicación

5.1.1 Definición

Un primer acercamiento a la definición de comunicación puede realizarse desde su etimología. La palabra deriva del latín *communicare*, que significa "compartir algo, poner en común". Por lo tanto, la comunicación es un fenómeno inherente a la relación que los seres vivos mantienen cuando se encuentran en grupo.

Cuando los seres humanos se comunican entre sí están compartiendo cuestiones, llevando a que las situaciones propias sean comunes con el otro y con las que este tenga. Por tanto, la comunicación es una actividad absolutamente humana y parte de la relación de las personas en cualquier ámbito y momento de la vida.

Si no fuese gracias a la comunicación no podría conocerse lo que nos rodea y además compartirlo con el propio entorno, pero al ser un hecho concreto y a disposición de todos, la comunicación facilita la obtención de información para conocer, expresarse y relacionarse con el resto de las personas.

El proceso de comunicación implica la emisión de señales, tales como sonidos, gestos o señas, con la única intención de dar a conocer un mensaje. Para que en la comunicación, el mensaje, al destinatario, será necesario que este cuente con las habilidades de decodificar e interpretar el mensaje en cuestión.

En tanto, en este proceso casi siempre se producirá un *feedback*, una ida y vuelta, porque una vez que el emisor emite su mensaje el proceso se revierte y el receptor al momento de responder se convertirá en el emisor, siendo el emisor original el receptor del proceso de comunicación.

Sus elementos, por tanto son:

- **Emisor**: Aquél que transmite la información (un individuo, un grupo o un dispositivo).
- **Receptor**: Aquél, individual o colectivamente, que recibe la información. Puede ser un dispositivo.
- **Código**: Conjunto o sistema de signos que el emisor utiliza para codificar el mensaje.
- **Canal**: Elemento físico por donde el emisor transmite la información y que el receptor capta por los sentidos corporales. Se denomina canal tanto al medio natural (aire, luz) como al medio técnico empleado (imprenta, telegrafía, radio, teléfono, televisión, ordenador, etc.) y se perciben a través de los sentidos del receptor (oído, vista, tacto, olfato y gusto).
- **Mensaje**: La propia información que el emisor transmite.
- **Contexto**: Circunstancias temporales, espaciales y socioculturales que rodean el hecho o acto comunicativo y que permiten comprender el mensaje en su justa medida.

La principal dificultad que puede presentarse dentro del proceso comunicativo es lo que se conoce como ruido, una perturbación que complicará el normal desarrollo del mensaje. Algunos ruidos comunes a la hora de comunicarse resultan ser: distorsión en el sonido, empleo de ortografía defectuosa o la disfonía del emisor.

La comunicación, en los seres humanos resulta ser una acción propia de la actividad psíquica, procediendo la misma del pensamiento, el lenguaje y las capacidades psicosociales de relación. La comunicación, ya sea verbal o no, le permitirá a los individuos influir en las decisiones de los demás y también ser influido por las que tenga el resto.

5.1.2 Factores que impiden la comunicación humana

El conjunto de factores que impiden la comunicación son considerados como barreras u obstáculos, que afectan sus objetivos, llegando a deformarlos y entre ellas podemos mencionar: *"Barreras psicológicas, semánticas, fisiológicas, físicas, administrativas".*

- ***Psicológicas***. Agrupan a varios procesos o factores mentales que no permiten la compresión de una idea, como por ejemplo; las suposiciones, preocupaciones o emociones externas al ámbito laboral, timidez, sobre valoración, etc. Estos procesos afectan a un receptor para que pueda aceptar o rechazar la comunicación de ideas, ya que para convencer, se debe explicar y las causas que pueden motivar a la formación de dichas barreras u obstáculos pueden ser; el estatus de poder, el sarcasmo, las criticas, el conocimiento exacto de un tema, habilidad lingüística, aspecto físico, interrupciones continuas, etc.
- ***Semánticas o verbales****. D*efinidas por no saber precisar un sentido, a través del significado de las palabras, generando distintas posibles interpretaciones entre emisor y receptor.
- ***Fisiológicas***. Cuando afectan los sentidos del emisor o el receptor, impidiéndole la compresión, como por ejemplo deficiencias foniátricas.
- ***Físicas***.se refieren a los factores ambientales, como el ruido, la distancia, visual, telecomunicación en sí.
- ***Administrativas***. son las que afectan a las estructuras organizacionales, por ser deficientes a nivel operacional de la empresa, así como inadecuadas o faltas de estrategia.

5.1.2 Trastornos en la comunicación y el lenguaje

Los trastornos que se pueden presentar en este ámbito, son muchos y variados, pueden afectar a uno, a varios o a todos los componentes del lenguaje, difieren en su etiología, en el pronóstico, en las necesidades educativas que generan y en la respuesta educativa que requieren.

Delimitar el concepto de trastorno, dependerá de dónde se ponga el límite de lo normal y lo patológico. Por lo tanto al ser subjetivo, dependerá del criterio del observador que va a emitir el juicio y de las normas sociales imperantes.

En general, se considera que un lenguaje normal, es aquel que tiene un uso preciso de las palabras según su significado, un vocabulario de calidad y cantidad, claridad de la articulación, una forma gramatical adecuada, un ritmo y velocidad apropiados, un volumen de voz audible, un tono adecuado a la edad y el sexo y una entonación de las frases en concordancia con su significado y sus necesidades expresivas.

Este canon de normalidad, sólo es aplicable al lenguaje adulto, ya que en el lenguaje infantil normal, todas o casi todas estas habilidades están en pleno proceso de desarrollo, sin que se considere un trastorno, sino propio del desarrollo evolutivo y que de forma natural o con intervención directa o indirecta, irá desapareciendo sin dejar secuelas.

No obstante, existe un pequeño grupo de niños y niñas que sí presentan verdaderos indicadores de trastornos. De aquí se deduce la importancia de conocer los parámetros evolutivos de la edad, para no incurrir en errores de considerar patológico lo que es normal en determinados momentos del proceso evolutivo.

5.1.2.1 Trastornos en el habla

Hablar es expresar a través de mecanismos físicos y fisiológicos todos los procesos de lenguaje interior (léxico-semánticos, morfológico-sintáctico, fonológico y pragmático). Para hablar, se necesita un flujo de aire en la espiración que al pasar por las cuerdas vocales, las hace vibrar y producir el sonido, las posiciones y movimientos de los órganos de la articulación (labios, mandíbula, lengua, paladar, etc.), así como, la forma en que se emite el aire (oral o nasal), y, cómo resuena en las cavidades orales y nasales, es lo que permite producir el habla.

Clasificación:

- Alteraciones que afectan a la articulación
    - Dislalias
    - Trastorno Fonológico
    - Disglosias
    - Disartrias
- Alteraciones que afectan a la fluidez verbal y el ritmo en la expresión
    - Disfemia
    - Taquilalia y farfulleo
    - Bradilalia
- Alteraciones de la voz
    - Disfonía

- Rinofonía

5.1.2.2 Trastornos del lenguaje

Los trastornos del lenguaje, se dan formando un continuo que iría, desde el retraso simple del lenguaje hasta la pérdida total de las capacidades lingüísticas en el caso de la afasia.

Otro importante aspecto a resaltar, es el diferente pronóstico que cada uno de los tipos de trastorno tiene. Mientras, en los Retrasos en la adquisición del lenguaje, el pronóstico es bueno, pudiendo normalizarse con la intervención, Rondal, (1985) en los casos de Trastorno Específico del Lenguaje o en Afasia, el pronóstico es negativo, ya que aunque mejoran, suelen dejar secuelas permanentes.

Clasificación:

- Retrasos en la adquisición y desarrollo del lenguaje
- Trastorno Específico del Lenguaje
- Afasias

5.1.2.3 Trastornos de la comunicación

A lo largo del siglo pasado, se puso el énfasis en el estudio de las alteraciones que presentaban los niños en su desarrollo del lenguaje. Inicialmente, se centraron en aquellos que perturbaban la voz y el habla. En la siguiente fase con el auge de la lingüística, el enfoque se centró en el estudio de estructuras semánticas, gramática y sintaxis. Al final del siglo, con el enfoque pragmático, se pretendió situar su desarrollo en el marco de comunicación y de las interacciones sociales.

Ante esta denominación, se estaría frente a trastornos del componente pragmático del lenguaje. Es un concepto reciente, en el que todavía no existe un claro consenso en la clasificación de los trastornos o síndromes que lo integran.

5.2 Discapacidades

5.2.1 Discapacidades psíquicas

El término "discapacidad intelectual" equivale a "retraso mental" que se define por la AAMR a la "discapacidad caracterizada por limitaciones significativas en el funcionamiento intelectual y en la conducta adaptativa que se manifiesta en habilidades conceptuales, sociales y prácticas. Esta discapacidad comienza antes de los 18 años."

El desarrollo intelectual estará relacionado, según este organismo, con las siguientes dimensiones:

- Capacidades intelectuales
- Conducta adaptativa (conceptual, social y práctica)
- Participación, interacciones y roles sociales
- Salud (salud física, salud mental, etiología)
- Contexto (ambientes y cultura).

Se puede realizar la siguiente clasificación atendiendo al índice del C.I., según una escuela Navarra (CREENA) especializada en gente con discapacidad psíquica:

- **Retraso mental/discapacidad intelectual ligera o leve**. Se incluyen en la misma las personas cuya medida en C.I., sin llegar a 55 – 50, se sitúa por debajo de 75 – 70 (unas 2 desviaciones típicas por debajo de la media, con un error de medida de aproximadamente 5 puntos). Acerca de ese tramo límite por arriba, se indica que se podría diagnosticar discapacidad ligera con un cociente intelectual entre 70 y 75 si existe déficit significativo en conducta adaptativa, pero no cuando no exista.

  Estas personas suelen presentar ligeros déficits sensoriales y/o motores, adquieren habilidades sociales y comunicativas en la etapa de educación infantil y, con frecuencia, no se diferencian de sus iguales por los rasgos físicos. A lo largo de la enseñanza básica, suelen llegar a adquirir aprendizajes instrumentales y algún grado de conocimientos académicos.

- **Retraso mental/discapacidad intelectual media o moderada**. Se incluyen en la misma las personas cuya medida en C.I. está entre 55 – 50 y 40 – 35. Lógicamente, con este nivel intelectual, las posibilidades adaptativas de estas personas suelen verse muy afectadas en todas las áreas de desarrollo. Como grupo suponen alrededor del 10 % de toda la población con discapacidad intelectual.

  A lo largo de la enseñanza básica, suelen desarrollar habilidades comunicativas durante los primeros años de la infancia y, durante la escolarización, pueden llegar a alcanzar algún grado de aprendizajes instrumentales. Suelen aprender a trasladarse de forma autónoma por lugares que les resulten familiares, atender a su cuidado personal con cierta supervisión y beneficiarse del adiestramiento en habilidades sociales.

- **Retraso mental / discapacidad intelectual severa o grave**. Se sitúa en el intervalo de C.I. entre 35 – 40 y 20 – 25 y supone el 3 – 4 % del total de la discapacidad. Las adquisiciones de lenguaje en los primeros años suelen ser escasas y a lo largo de la enseñanza básica pueden aprender a hablar o a emplear algún signo de comunicación alternativo. Las posibilidades adaptativas están muy afectadas en todas las áreas de desarrollo, pero es posible el aprendizaje de habilidades elementales de cuidado personal.
- **Retraso mental discapacidad profunda / pluridiscapacidad**. La mayoría de estas personas presenta una alteración neurológica identificada que explica esta discapacidad, la confluencia con otras (de ahí el término pluridiscapacidad que aquí se le asocia) y la gran diversidad que se da dentro del grupo. Ello condiciona el hecho de que uno de los ámbitos de atención prioritaria sea el de la salud física.

  El C.I. de estas personas queda por debajo de 20 – 25 y son el 1 – 2 % de la tipología. Suelen presentar limitado nivel de conciencia y desarrollo emocional, nula o escasa intencionalidad comunicativa, ausencia de habla y graves dificultades motrices. El nivel de autonomía, si existe, es muy reducido.

5.2.2 Discapacidades motoras

Una persona con discapacidad motora es aquella que presenta de manera transitoria o permanente alguna alteración de su aparato motor, debido a un deficiente funcionamiento en el sistema nervioso, muscular y /u óseo, o en varios de ellos relacionados, que en grados variables limita alguna de las actividades que puede realizar el resto de las personas de su edad.

Esta discapacidad implica la limitación del normal desplazamiento físico. Las personas que tienen este tipo de discapacidades pueden ser semiambulatorias o no ambulatorias. En el caso del primer tipo, se movilizan ayudadas por elementos complementarios, como ser muletas, bastones, andadores, etc. Las no ambulatorias sólo pueden desplazarse con silla de ruedas. Esto implica la fundamental importancia de estos elementos para las personas con discapacidad.

Las causas de esta discapacidad pueden ser por secuelas neurológicas, miopáticas, ortopédicas o reumatológicas. Las secuelas neurológicas se dividen en cerebrales (parálisis o hemiplejia) o medulares.

Para clasificar una deficiencia motora hay que atender a cuatro criterios:

Fecha de aparición

La deficiencia puede aparecer desde el nacimiento, después, en la adolescencia y a lo largo de toda la vida:

- Congénitas: cualquier tipo de malformaciones. Por ejemplo: espina bífida.

- Después del nacimiento: como la parálisis cerebral, cuya detección suele ser unas semanas más tarde del nacimiento (cuando no muestra los patrones motrices adecuados a su edad).
- Adolescencia: miopatías.
- A lo largo de toda la vida: accidentes, traumatismos (traumatismos craneoencefálicos, vertebrales, derrames, etc.) etc.

Etiopatología

- Transmisión genética: Miopatía de Duchenne de Boulogne, transmitida por una madre portadora
- Infecciones microbianas: tuberculosis ósea, poliomielitis, etc.
- Accidentes: pueden sobrevenir en el momento del parto o a lo largo de toda la vida
- Origen desconocido: como, por ejemplo: espina bífida, escoliosis ideopática o el origen de los tumores

Localización topográfica

- Parálisis: monoplejia, diplejia o parálisis bilateral, tetraplejia, paraplejia, triplejia y hemiplejia
- Paresia: es una parálisis más suave, ligera o incompleta. Tipos: monparesia, hemiparesia, tetraparesia, parapesia o paraparesia, triparesia y diparesia.

Origen

- Origen cerebral: parálisis cerebral, traumatismos cerebrales, tumores cerebrales, etc.
- Origen espinal: poliomielitis, espina bífida, lesiones medulares degenerativa que degeneran en ataxias y traumatismo medulares
- Origen Muscular: miopatías (en realidad son distrofias)
- Origen óseo-articular u osteoarticular: malformaciones congénitas, distrofias, enfermedades microbianas (osteomielitis o tuberculosis osteoarticular), traumatismos de la infancia, lesiones osteoarticulares por desviaciones del raquis (cifosis, escoliosis, etc.)

5.2.3 Discapacidades sensoriales

Una persona tiene un déficit sensorial cuando presenta una alteración o deficiencia que afecta a sus órganos sensoriales principales.

Dentro de la categoría de la discapacidad sensorial, se encuentra la discapacidad visual, la discapacidad auditiva y otros tipos de discapacidades relacionadas con disminución de algunos de los sentidos.

5.2.2.1 Discapacidad Visual

Según la Organización Mundial de la Salud una persona con deficiencia visual presenta una ausencia o mal funcionamiento del sistema óptico, causado por enfermedad, lesión o anormalidad congénita que, a pesar de la corrección, convierte a la persona en un sujeto oficialmente considerado como deficiente visual en el país en el que vive.

La expresión más grave de una patología ocular es la ceguera, que se puede definir como una pérdida de visión lo suficientemente grande como para evitar que una persona se mantenga por sí misma en cualquier ocupación, volviéndolo dependiente de otros medios o personas para poder subsistir.

En términos generales, la mayoría de la gente asocia el significado de ciego con una persona con ausencia total de visión, sin embargo entre los conceptos de ceguera total y la visión normal existen distintas categorías. Según la OMS, la clasificación de la agudeza visual y deterioro es la siguiente:

- **La agudeza visual baja**. Significa visión entre 20/70 y 20/400 con la mejor corrección posible, o una visual de campo de 20 grados o menos.
- **Ceguera**. Se define como una agudeza visual peor de 20/400, con la mejor corrección posible, o un campo visual de 10 grados o menos
- **Moderado deterioro visual o baja visión.** Se considera agudeza visual de 20/70 a 20/400 (inclusive).

5.2.2.1 Discapacidad Auditiva

Disfunciones o alteraciones cuantitativas en una correcta percepción auditiva además trae aparejadas otras alteraciones cuya gravedad vendrá condicionada por factores tan importantes como son la intensidad de la pérdida auditiva y el momento de aparición de la misma.

Teniendo en cuenta que los órganos sensoriales proporcionan informaciones importantes que inciden en un desarrollo evolutivo adecuado de la persona, hay que considerar que el aislamiento y la falta de información a que se ve sometida ésta por causa del déficit auditivo pueden representar implicaciones importantes para su desarrollo del lenguaje y las diversas modalidades comunicativas, así como en los campos cognitivo, cognoscitivo, emocional, comportamental, social y ocupacional. Ocurren cuando hay un problema en los oídos o en una o más partes que facilitan la audición.

Una persona con una deficiencia auditiva puede ser capaz de oír algunos sonidos o puede no oír nada en absoluto. La palabra deficiencia significa que algo no está funcionando correctamente o tan bien como debería. La gente también utiliza palabras como sordo, sordera o duro de oído para referirse a las pérdidas auditivas. Es una de las anomalías congénitas más frecuentes. Los problemas auditivos también se pueden desarrollar más tarde en la vida de una persona.

Una de las clasificaciones que se pueden hacer acerca de los tipos de deficiencia auditiva son:

- Hipoacusias:

    - Deficiencia auditiva ligera.
      - Pérdida auditiva de entre 20 y 40 db.

- Pequeñas dificultades articulatorias.
- No identifican totalmente todos los fonemas.

  - Deficiencia auditiva media.
    - Pérdida auditiva de entre 40 y 70 db.
    - Identificación sólo de vocales.
    - Articulación defectuosa.
    - Lenguaje productivo limitado.
  - Capacidad para la estructuración del pensamiento verbal.

- Sorderas:

  - Deficiencia auditiva severa.
    - Pérdida auditiva de entre 70 y 90 db.
    - Percepción de algunos sonidos, pero imposibilidad de adquisición espontánea del lenguaje.
    - Los afectados son llamados sordos medios.
  - Deficiencia auditiva profunda.
    - Pérdida auditiva superior a los 90 db.
    - No pueden adquirir el lenguaje oral.
    - Los afectados son llamados sordos profundos.

5.3 Sistemas aumentativos y alternativos de comunicación (SAAC)

Los Sistemas Aumentativos y Alternativos de Comunicación (SAAC) son formas de expresión distintas al lenguaje hablado, que tienen como objetivo aumentar y/o compensar las dificultades de comunicación y lenguaje de muchas personas con discapacidad.

La comunicación y el lenguaje son esenciales para todo ser humano, para relacionarse con los demás, para aprender, para disfrutar y para participar en la sociedad y hoy en día, gracias a estos sistemas, no deben verse frenados a causa de las dificultades en el lenguaje oral. Por esta razón, todas las personas, ya sean niños, jóvenes, adultos o ancianos, que por cualquier causa no han adquirido o han perdido un nivel de habla suficiente para comunicarse de forma satisfactoria, necesitan usar un SAAC.

Entre las causas que pueden hacer necesario el uso de un SAAC, se encuentran la parálisis cerebral, la discapacidad intelectual, los trastornos del espectro autista, las enfermedades neurológicas tales como la esclerosis lateral amiotrófica , la esclerosis múltiple, el párkinson, las distrofias musculares, los traumatismos cráneo-encefálicos, las afasias o las pluridiscapacidades de tipologías diversas, entre muchas otras.

La Comunicación Aumentativa y Alternativa, no es incompatible, sino complementaria a la rehabilitación del habla natural, y además puede ayudar al éxito de la misma cuando éste es posible. No debe pues dudarse en introducirla a edades tempranas, tan pronto como se observan dificultades en el desarrollo del lenguaje oral, o poco después de que cualquier accidente o enfermedad haya provocado su deterioro. No existe ninguna

evidencia de que el uso de CAA inhiba o interfiera en el desarrollo o la recuperación del habla.

5.4 Recursos utilizados en CAA

La Comunicación Aumentativa y Alternativa incluye:

- **Sistemas de símbolos**. Incluyen sistemas tanto gráficos (fotografías, dibujos, pictogramas, palabras o letras) como gestuales (mímica, gestos o signos manuales) y, en el caso de los primeros, requiere también el uso de productos de apoyo. Los diversos sistemas de símbolos se adaptan a las necesidades de personas con edades y habilidades motrices, cognitivas y lingüísticas muy dispares.
- **Productos de apoyo para la comunicación.** Incluyen recursos tecnológicos, como los comunicadores de habla artificial o los ordenadores personales y tablets con programas especiales, que permiten diferentes formas de acceso adaptadas algunas para personas con movilidad muy reducida, y facilitan también la incorporación de los diferentes sistemas de signos pictográficos y ortográficos, así como diferentes formas de salida incluyendo la salida de voz. También pueden consistir en recursos no tecnológicos, como los tableros y los libros de comunicación.
- **Estrategias y productos de apoyo para el acceso**. Son diversos instrumentos tales como los punteros, los teclados y ratones adaptados o virtuales y los conmutadores, que sirven para acceder a los ordenadores, comunicadores, tableros o libros de comunicación.

5.4.1 Sistemas de símbolos

Teniendo en cuenta una clasificación de los sistemas de símbolos para la CAA en gestuales y gráficos, se pueden encontrar una gradación desde sistemas muy sencillos, que se adaptan a personas con déficits cognitivos y lingüísticos de diversa consideración, hasta sistemas complejos que permiten niveles avanzados de lenguaje signado (basado en signos manuales) o asistido (basado en signos gráficos).

- **Símbolos gestuales**. Esta gradación abarca desde el uso de mímica y gestos de uso común hasta el uso de signos manuales, generalmente en el orden correspondiente al lenguaje hablado; es lo que se denomina lenguaje signado o bimodal. Las lenguas de signos utilizadas por las personas no oyentes no se consideran SAAC, ya que constituyen idiomas que se han desarrollado y se adquieren de forma natural, al igual que ocurre con el lenguaje hablado. El uso de signos manuales requiere disponer de habilidades motrices suficientes, como puede ser el caso de personas con discapacidad intelectual o TEA.
- **Símbolos gráficos.** Abarcan desde sistemas muy sencillos basados en dibujos o fotografías hasta sistemas progresivamente más complejos como los sistemas pictográficos o la ortografía tradicional (letras, palabras y frases). Gracias a los productos de apoyo para la comunicación y los diversos recursos para el acceso, los sistemas gráficos pueden ser usados por personas con movilidad reducida, incluso en casos de extrema gravedad. Por ello, además de ser usados, como en el caso

anterior, por personas con discapacidad intelectual o TEA, los usan también personas con discapacidades motoras.

- **Sistemas pictográficos.** Se aplican a personas que no están alfabetizadas a causa de la edad o la discapacidad. Tienen la ventaja de permitir desde un nivel de comunicación muy básico, que se adapta a personas con niveles cognitivos bajos o en etapas muy iniciales, hasta un nivel de comunicación muy rico y avanzado, aunque nunca tan completo y flexible como el que se puede alcanzar con el uso de la lengua escrita. Los sistemas pictográficos más usados en los diversos territorios del estado español son el sistema SPC (Sistema Pictográfico de Comunicación) y el sistema ARASAAC, desarrollado por este propio Portal Aragonés de CAA y que es de libre disposición con licencia Creative Commons.

5.4.2 Productos de apoyo para la comunicación

Pueden dividirse los productos de apoyo para la comunicación en básicos y tecnológicos. Los tableros de comunicación son productos de apoyo básicos que consisten en superficies de materiales diversos en las que se disponen los símbolos gráficos para la comunicación (fotografías, pictogramas, letras, palabras y/o frases) que la persona indicará para comunicarse. Cuando los símbolos se distribuyen en varias páginas se habla de de libros de comunicación.

Entre los productos tecnológicos se encuentran los comunicadores electrónicos especialmente diseñados para tal fin y los ordenadores portátiles o las tablets con programas especiales que los convierten en comunicadores. Los comunicadores electrónicos dedicados o emulados en ordenadores se personalizan con los símbolos gráficos que requiere cada persona y se caracterizan por ser portátiles y adaptarse a las formas de acceso apropiadas para cada persona (teclados, ratones, conmutadores, etc.). Disponen de una salida para los mensajes en forma de habla digitalizada o sintetizada, así como también, a menudo, de otras salidas como pantalla, papel impreso o incluso funciones de control del entorno.

5.4.3 Estrategias y productos de apoyo para el acceso

Gracias a las diferentes estrategias y productos de apoyo para el acceso, por muy restringida que se encuentre la movilidad de una persona, casi siempre es posible encontrar una solución para que pueda acceder a la comunicación, así como a otras actividades, tales como la movilidad asistida, el control del entorno o el acceso al ordenador para la escritura, el dibujo, el juego o la comunicación a través de la red.

Para indicar los símbolos gráficos en los comunicadores, tableros y libros de comunicación existen cinco estrategias fundamentales:

- **La selección directa.** Consiste en señalar o pulsar las teclas directamente, con el dedo, con la mirada o con otras partes del cuerpo, para indicar los pictogramas, palabras o letras que se quieren comunicar. Los punteros de distinto tipo son ejemplos de productos de apoyo que puede facilitar la selección o acceso directo.

- **La selección con ratón.** Solamente para productos electrónicos, consiste en acceder con un ratón a teclados o cuadrículas con símbolos para la comunicación en pantalla. Se pueden usar una gran variedad de ratones adaptados, en forma de joystick, trackball, así como el ratón facial (controlado con movimientos de la cabeza), el ratón controlado con la mirada o el multimouse, consistente en cinco teclas o conmutadores.
- **La exploración o barrido dependiente.** Solamente en tableros o libros, consiste en que el interlocutor vaya señalando, uno en uno o por grupos, filas y columnas, los símbolos o letras a comunicar, hasta que el hablante asistido indique con un gesto que se ha dado con el que quería comunicar.
- **La exploración o barrido independiente.** Solamente para productos electrónicos, en este caso es el comunicador u ordenador el que presenta las diferentes opciones a comunicar hasta que el hablante asistido selecciona la que le conviene pulsando un conmutador. Existen muchos tipos de conmutadores que se pueden activar con diferentes partes del cuerpo.
- **La selección codificada.** En este caso, cada símbolo o letra tiene un código, por ejemplo un número de dos o tres cifras o un color y un número, de manera que el hablante asistido indica de forma directa o por barrido este código para transmitir el símbolo o letra. De esta forma con pocas teclas o casillas puede acceder a un gran número de símbolos.

5.5 Juguetes adaptados

Se consideran juguetes adaptados aquellos juguetes estándar a los que se les ha hecho alguna modificación física que facilita el uso y el juego a un niño con discapacidad.

Estos consisten en un juguete con una o dos entradas de conmutador para poder ser accionado mediante pulsación. Hay muchos tipos de adaptaciones, desde muy sencillas, como poner velcro a fin de fijar el juego o juguete, hasta otras bastante más complicadas, como traducir los efectos sonoros a visuales o instalar un *jack* que facilite el uso del juguete.

Los criterios básicos que marcarán el tipo y grado de adaptación del juguete son la capacidad de movimiento del niño o niña, sus capacidades de manipulación y sus posibilidades de percepción sensorial. Las modificaciones que se apliquen deberán estar también en consonancia con la complejidad física, el grado de discapacidad y el nivel de comprensión del niño.

Ante cualquier tipo de adaptación hay que velar por las condiciones de seguridad y asegurarse de que las adaptaciones que se introduzcan no añadan ningún peligro para el niño o niña que lo usará o para los de su entorno.

Otro aspecto que debe tenerse en cuenta es el aspecto final del juguete una vez adaptado, velando para que éste siga siendo atractivo y deseable a los ojos de un niño con o sin discapacidad.

Actualmente la gama de juguetes adaptados no es muy amplia, ya que existen pocas empresas encargadas de dicha adaptación, los precios aumentan entre un treinta y un

cuarenta por ciento respecto del juguete sin adaptar y no todos los juguetes son adaptables para los diferentes tipos y grados de discapacidad.

A continuación se describen dos juguetes, en forma de pack, que han sido adaptados:

- **BJ-K02, Kit de Juego de Bolos.** Consiste en un juego de bolos infantil ligero, el cual incluye un conmutador el cual puede accionar una sopladora eléctrica, también incluida y que hará que se mueva la bola. Y además, incluye un comunicador Go Talk, con 20 plantillas con pictogramas relacionados con la actividad.

Figura 5.1: BJ-K02, Kit de Juego de Bolos

- **Kit pecera adaptada**. Kit que permite dar alimentos a los peces de la pecera mediante un brazo articulado y un alimentador adaptado, además incluye un pulsador, una cámara de fotos adaptada y además incluye un comunicador Go Talk, con 20 plantillas con pictogramas relacionados con la actividad.

Figura 5.2: Kit pecera adaptada

5.6 Accesibilidad orientada a dispositivos móviles

Actualmente existen mecanismos que posibilitan la accesibilidad en dispositivos para personas con algunos tipos de discapacidad. Es un derecho, que todos tenemos y por ello, es necesario que se sigan desarrollando para que cualquier usuario, con discapacidad o no tenga las mismas oportunidades de interactuar con dispositivos móviles.

Hoy en día, en el mercado existen soluciones para personas con discapacidades auditivas y visuales, también se han lanzado otros dispositivos que facilitan las funciones a otros usuarios que, por cuestión de edad o algún otro tipo de problema físico no pueden tener fácil acceso al móvil.

5.6.1 Accesibilidad para personas con discapacidad auditiva

Las personas que tienen una discapacidad auditiva pueden utilizar un dispositivo móvil que sea compatible con un accesorio adicional llamado “lazo de inducción”.

El lazo de inducción es un filtro de ruidos que ayuda a quienes tienen un audífono con bobina telefónica "T" (es un audífono especial que funciona con teléfonos) a eliminar las interferencias para mantener una conversación.

Este accesorio se cuelga en el cuello y permite contestar llamadas con una sola tecla desde el dispositivo, así como también es posible utilizarlo como manos libres. Con el lazo de inducción también se puede apagar o encender el dispositivo móvil.

5.6.2 Accesibilidad para personas con discapacidad visual

Para las personas con discapacidad visual se usan soluciones a partir de diferentes programas, los cuales traducen el texto del menú de la pantalla y de los mensajes en voz. En España los fabricantes de dispositivos móviles junto con la empresa Code Factory y la ONCE-CIDAT comercializan una aplicación diseñada para dispositivos móviles llamada Mobile Speak.

Mobile Speak funciona como un lector de pantalla para reconocer el texto y toda la información visual que aparece en el terminal y transformarla en sonido.

Esta aplicación hace posible que las personas con discapacidad visual puedan acceder a las funcionalidades más utilizadas en los dispositivos móviles. Hacer y recibir llamadas, mensajes de texto, registro de llamada y lista de contactos; navegar por internet, escuchar archivos de audio o conectarse con otros dispositivos por medio de Bluetooth o Wifi, son algunas de las funcionalidades que esta aplicación puede realizar.

5.6.3 Accesibilidad para personas mayores

Para las personas mayores o con alguna dificultar física o intelectual, la empresa austriaca Emporia comercializa dispositivos móviles pensados para utilizarse de manera sencilla, incluyendo en el terminal las funciones más básicas, como acceso a llamadas, mensajes de texto, agenda de contactos, entre otras cosas.

Dentro de la gama Emporia, existe uno de sus primeros dispositivos posee una gran pantalla con iluminación naranja para mejorar el contraste con el contenido, así como un volumen de timbre y altavoz adecuados para personas mayores, de forma que puedan escuchar con claridad sin que el sonido sea molesto. También incluye un botón rojo en la parte de arriba del dispositivo, en el cual se puede programar hasta cinco números de emergencia para tener acceso directo a ellos.

Actualmente la gama de teléfonos Emporia cuentan con servicios como llamada de emergencia con la que automáticamente se localiza la posición del usuario, gracias al GPS integrado. Cuentan con un nivel de vibración extrafuerte, botones de gran tamaño, un sensor de caídas, reloj de pulsera etc.

Capitulo 6. Aplicación para la accesibilidad Sphero

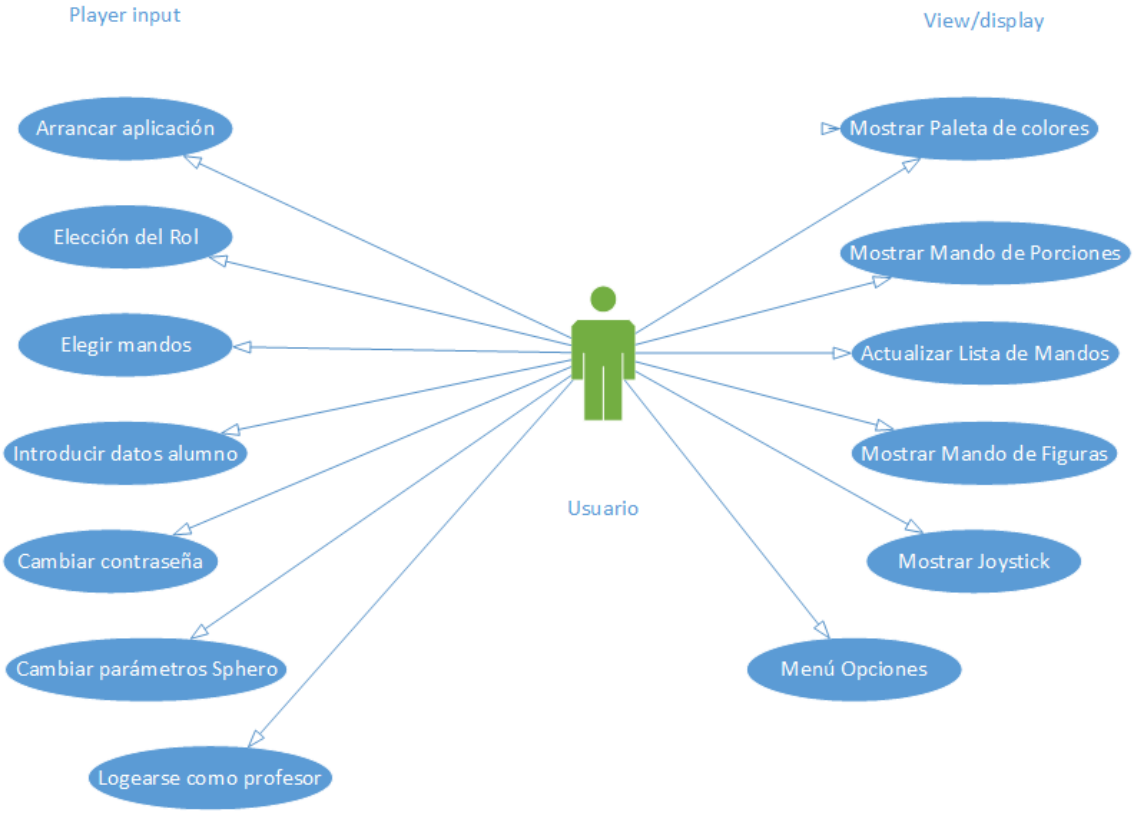
En este capítulo se va a presentar y explicar todo lo relacionado con la aplicación desarrollada sobre el sistema operativo Android. Se describirá su diseño e implementación, donde podrán encontrarse diagramas, imágenes y fragmentos de código para favorecer la comprensión del lector.

6.1 Diseño

6.1.1 Especificación de casos de uso

En este apartado se utilizan los casos de uso como una simplificación gráfica de las funcionalidades del sistema para su mejor comprensión. Un caso de uso, representa un uso típico del sistema, permitiendo dicha técnica capturar y definir los requisitos que se deben cumplir en una aplicación y la interacción que existe entre el usuario y el sistema. Es utilizada frecuentemente en el área de la Ingeniería del Software para mostrar al cliente de manera sencilla de qué será capaz su futuro sistema.

Sin embargo, aunque las posibilidades que ofrece esta técnica son importantes, no se va a profundizar en la misma, mostrando solo los casos de uso que resulten más ilustrativos al lector y que mejoren su comprensión global sobre la aplicación.



En el diagrama presentado, cada caso de uso consiste en un óvalo con un nombre descriptivo en su interior. Además se encuentra un actor, que es el que va a interactuar con todo el sistema. Existe un único actor de nombre “usuario” que es el que realiza todos los casos de uso.

A continuación, se ofrece una breve descripción del cometido de cada uno de los casos de uso mostrados en el diagrama:

- **Arrancar aplicación:** consiste en la ejecución del programa principal.
- **Elección de rol:** escoger entre rol de profesor o alumno.
- **Elegir mandos:** escoger entre una lista de mandos, los que el usuario crea convenientes.
- **Introducir datos del alumno:** escribir los datos necesarios para identificar al alumno que va a utilizar los mandos y así particularizar el comportamiento realizado.
- **Cambiar contraseña:** introducir una contraseña diferente a la existente, por motivos de seguridad.
- **Cambiar parámetros Sphero:** introducir una serie de valores que harán que el funcionamiento de Sphero sea diferente al que viene por defecto, y favorecerán su manejo.
- **Logearse como profesor:** consiste en introducir las credenciales pedidas para acceder al menú del usuario con rol profesor.

6.1.2 Interfaz de usuario

Resulta posible diseñar e implementar una interfaz gráfica con el propio lenguaje Java pero resulta demasiado costoso para el desarrollador, además de desaprovechar el modelo vista-controlador que ofrece Android. De esta forma, es posible dar aspecto a la aplicación mediante ficheros xml y por otro lado implementar toda la lógica con lenguaje Java.

Para realizar estas interfaces mediante el lenguaje xml, existen algunas herramientas disponibles en el mercado como podría ser los constructores GUI que se integran con Eclipse o constructores GUI independientes de Eclipse, que facilitan la labor.

Dentro de las opciones nombradas, para este proyecto, se ha optado por utilizar la herramienta que existe en Eclipse para desarrollar cada pantalla de la aplicación Android. No se ha trabajado directamente sobre el fichero xml en todos los casos, ya que, se han utilizado layouts en muchos casos, que contienen un gran número de elementos, para los que el desarrollador, necesita un punto de vista gráfico, para así distribuirlos de forma correcta en la pantalla.

Llegados a este punto, en caso de que el lector desee comprobar el aspecto final de la aplicación le remito al siguiente apartado de implementación, ya que por motivos ilustrativos se ha preferido incluir las capturas de pantalla en dicha sección.

6.2 Implementación

Una vez tomadas las decisiones en cuanto al diseño de la aplicación, en este apartado se realizará un estudio en profundidad de las funcionalidades implementadas dando una visión más detallada sobre el funcionamiento interno.

Para conseguir alcanzar el objetivo, en algunos de los siguientes apartados se recurrirá a la inclusión de fragmentos de código fuente para ilustrar las explicaciones.

6.2.1 Lenguaje de programación y entorno de desarrollo

Como ya se ha comentado en otros apartados, el lenguaje de programación utilizado para el desarrollo de aplicaciones para la plataforma Android es Java, aunque cabe destacar que las librerías que incluye el sistema están escritas en C y C++.

Para iniciar el desarrollo de una aplicación Android es condición indispensable descargar el SDK de la plataforma que el propio Google pone a disposición de cualquier programador. Este SDK contiene las bibliotecas necesarias para utilizar algunas funciones propias de la plataforma y que dan acceso a las partes hardware del dispositivo, como la vibración, la cámara o el GPS.

En cuanto al entorno de desarrollo integrado o IDE, se deja la elección a gusto del programador y aunque se ofrece soporte para otros entornos, Google recomienda utilizar su nuevo producto Android Studio pero, para este proyecto se ha utilizado Eclipse, ya que en el momento en el que se comenzó a desarrollar aún no existía el producto de Google. Con Eclipse existe un plugin para el mismo, que integra un completo emulador de la plataforma y facilita el trabajo de integración. Al crear un nuevo proyecto Android, dicho plugin genera el árbol de directorios necesario y crea algunos archivos básicos como el AndroidManifest.xml y R.java.

6.2.1.1 Configuración del entorno de desarrollo

En este apartado se va a presentar la instalación paso a paso de las herramientas necesarias para poder desarrollar una aplicación Android. Los principales elementos para el correcto funcionamiento de esta plataforma son: IDE Eclipse, SDK de Android y plugin ADT para Eclipse.

- **Instalación de Java Platform (JDK) desde la web de Oracle.** Conviene asegurarse de que el equipo con el que se va a trabajar tiene instalado y configurado correctamente el SDK.

Figura 6.1: Enlace de descarga de JDK

- **Instalación del IDE Eclipse.** Una vez instalado el JDK, se procederá a la instalación de Eclipse, desde su propia web. Existe la posibilidad de descargar la versión para sistema operativo de 32 bits o de 64 bits. En mi caso descargo la versión de 64 bits, una vez descargada simplemente debe ejecutarse el fichero ejecutable sin tener que instalar Eclipse en nuestro equipo previamente.

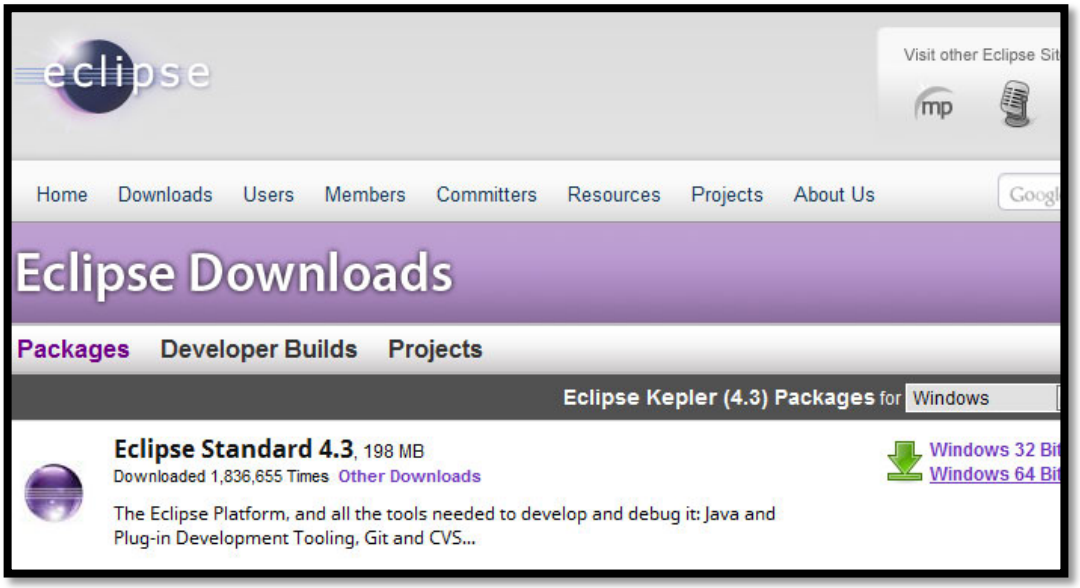


Figura 6.2: Enlace de descarga de de Eclipse

- **Instalación SDK Android.** Una vez instalado Eclipse, se procederá a instalar Android SDK, que quedará integrado en nuestro IDE una vez finalice su instalación.

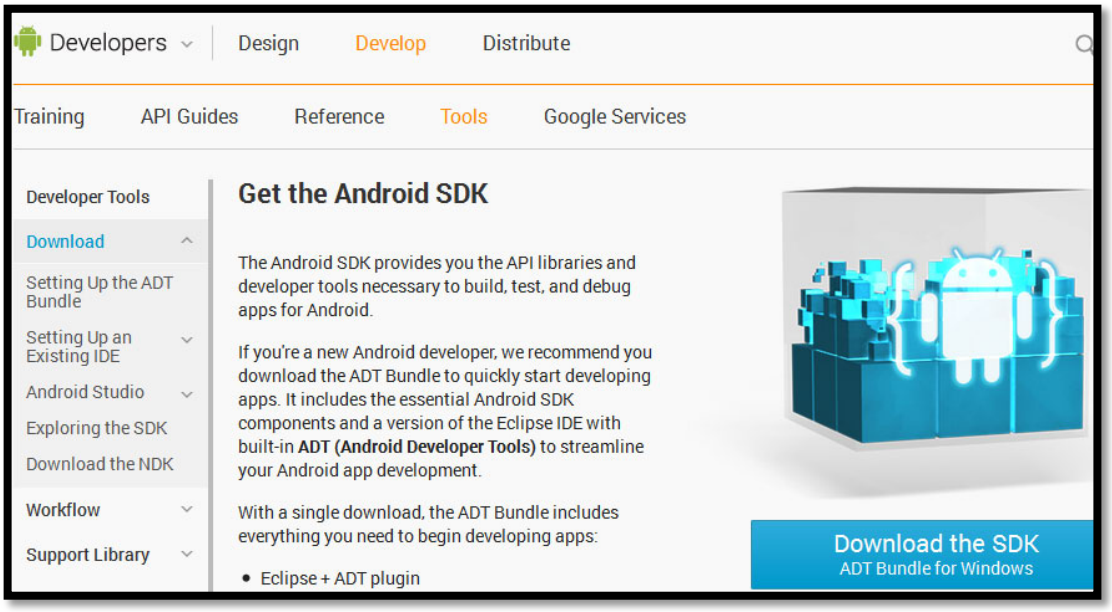


Figura 6.3: Enlace de descarga de SDK Android

- **Instalación plugin ADT de Eclipse.** Una vez que se tengan descargados tanto la plataforma Eclipse como el SDK Android, se deberá instalar el plugin ADT de Eclipse. Para ello, se accederá directamente desde la interfaz de Eclipse, a través de la opción "Install new software" de la opción "Help". Una vez que encontremos el paquete "Developer Tools" a través de la opción de búsqueda "Available Software Sites", deberá descargarse e instalarse. Posteriormente se deberá configurar en Eclipse el SDK indicándole la ruta de instalación de dicho plugin SDK, aplicar los cambios y ya quedará todo el entorno de desarrollo listo para comenzar a desarrollar la aplicación.

6.2.1.2 Integración del SDK Sphero en el proyecto de Eclipse

Suponiendo ya instalados, tanto Eclipse como el SDK de Android, para realizar una aplicación, en la que se vaya hacer uso de los métodos y atributos que Orbotix ha creado para controlar un Sphero, se debe descargar e integrar el SDK de Sphero en el proyecto creado en Eclipse.

Para ello, deberán seguirse los siguientes pasos:

- Se deberá acceder a la web oficial de Sphero y dentro del apartado "Developers" se deberá descargar SDK Sphero en el apartado SDK Quick Start.

Figura 6.4: Enlace de descarga de SDK Sphero para Android

- Una vez descargado, se deberán copiar las librerías que se proporcionan, en nuestro proyecto de Eclipse:

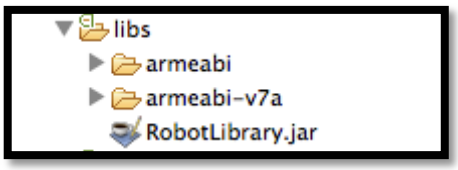


Figura 6.5: Librerías Sphero en el proyecto Eclipse

Las dos primeras carpetas, corresponden a las clases que Orbotix ha desarrollado para enviar eventos desde la aplicación que se desarrolle a su web. Actualmente, ese servicio está en reparación, por ello se ha optado por utilizar un servicio ajeno a Orbotix Sphero, llamado Flurry, que posibilita el mismo registro de eventos desde la aplicación Android desarrollada.

El fichero ".jar" corresponde a la librería que contiene todos los métodos y clases que se van a utilizar para interactuar con un Sphero y que Orbotix, el fabricante de este producto, pone a disposición del usuario, de forma gratuita y documentada.

- Llegados a este punto, se comprobará que están importadas las librerías de Orbotix en el buildpath del proyecto. Si es así, se podrán utilizar todos los atributos y métodos necesarios para controlar un Sphero.

6.2.1.3 Integración del SDK Flurry en el proyecto de Eclipse

Una vez instalados Eclipse, SDK de Android y SDK Sphero, para realizar una aplicación, en la que se vayan a enviar eventos de todo lo que el desarrollador desee, para posteriormente, tratar esos datos registrados, se deberá descargar e integrar el SDK de Flurry en Eclipse.

Para ello, deberán seguirse los siguientes pasos:

- Se deberá crear una cuenta en la web de Flurry, en la que se especifique el sistema operativo en el que va a correr la aplicación. Además deben indicarse, de forma opcional una serie de parámetros como categoría de la aplicación, nombre, etc.

- Una vez que ya esté creada, se deberá acceder a la cuenta:

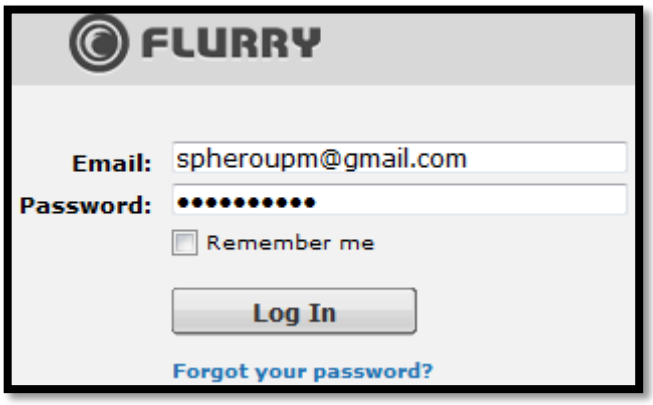


Figura 6.6: Pantalla de login en la web de Flurry

- Dentro del apartado descargas, se nos asignará un token único para la aplicación, y que estará ligado al usuario de la cuenta Flurry. Ese token será utilizado desde la aplicación que se desarrolle, para autenticar al usuario que va a enviar los eventos y así poder registrarlos en su cuenta.

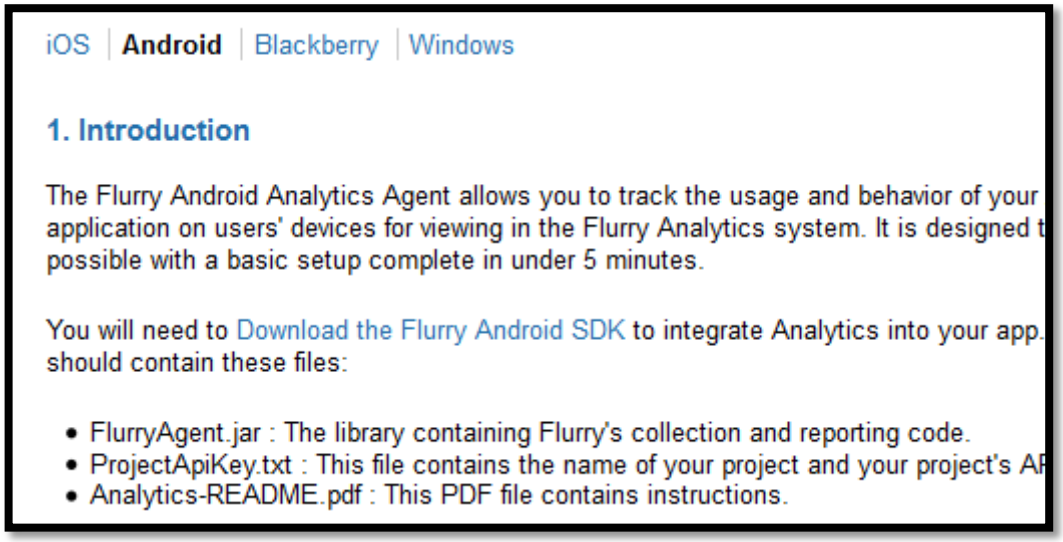


Figura 6.7: Enlace de descarga de SDK Flurry para Android

- Una vez descargado, deberá copiarse la librería Flurry a la carpeta libs del proyecto e importarse en el buildpath. Una vez realizado, se podrán invocar los métodos y utilizar los atributos de dicha librería.

6.2.2 Realización del Proyecto

Este proyecto, tiene una doble finalidad, por lo que es importante detallar ambas.

- Por un lado, es una aplicación que promueve la interacción, de usuarios con discapacidad, con un dispositivo externo radiocontrol; además favorece la integración del uso actual de dispositivos móviles con este tipo de usuarios.
  Es importarte remarcar, que a día de hoy, existen un grupo muy reducido de aplicaciones orientadas a la discapacidad. Bien por temas económicos, ya que, para realizar una aplicación de este tipo requiere seguir unos estándares de diseño y un comportamiento determinado y todo ello conlleva a utilizar un desarrollador específico, que conozca estas pautas a la hora de programar. Además conlleva un riesgo, y es que no todas las aplicaciones orientadas a la accesibilidad son manejables por cualquier usuario, hay que acotar un grado y el tipo de discapacidad. Ello conlleva a reducir enormemente el número de usuarios para la que va a ir destinada, y para muchas empresas de desarrollo de software, esto es algo poco viable, ya que los ingresos van a ser menores, que una aplicación que va a ir destinada a cualquier sector de población.

  Con ello quiero remarcar que la primera finalidad es ofrecer la posibilidad de usar un juguete, como Sphero, que está orientado a un sector de población sin discapacidad, haciendo uso de cualquier dispositivo móvil bajo sistema operativo Android, a personas con diferentes grados de discapacidad sensorial, motora y psíquica.

- Por otro lado, esta aplicación sirve para obtener resultados una vez que ella sido utilizada y que van a ayudar a beneficiar de forma terapéutica al usuario que ha hecho uso de ella. En estos resultados, van a ir incluidos todos los eventos que se producen al pulsar botones, tocar diferentes zonas de la pantalla, cambiar de mandos, etc. Y que van a posibilitar a una persona especializada en el tratamiento de personas con discapacidad, llevar un control del uso que hace el paciente y ver posibles mejoras que refleja el paciente a lo largo del tiempo.

Esta aplicación, consta de un número determinado de clases, algunas de ellas del tipo Activity, que fueron plasmadas y brevemente explicadas en el apartado *6.1.2 Modelo de clases.* A continuación se van a explicar de forma más detallada cada una de ellas, haciendo incapié en el requisito funcional que se pedía en la aplicación antes de desarrollarla y cómo se encontró la forma de implementarlo.

6.2.2.1 Clase Principal

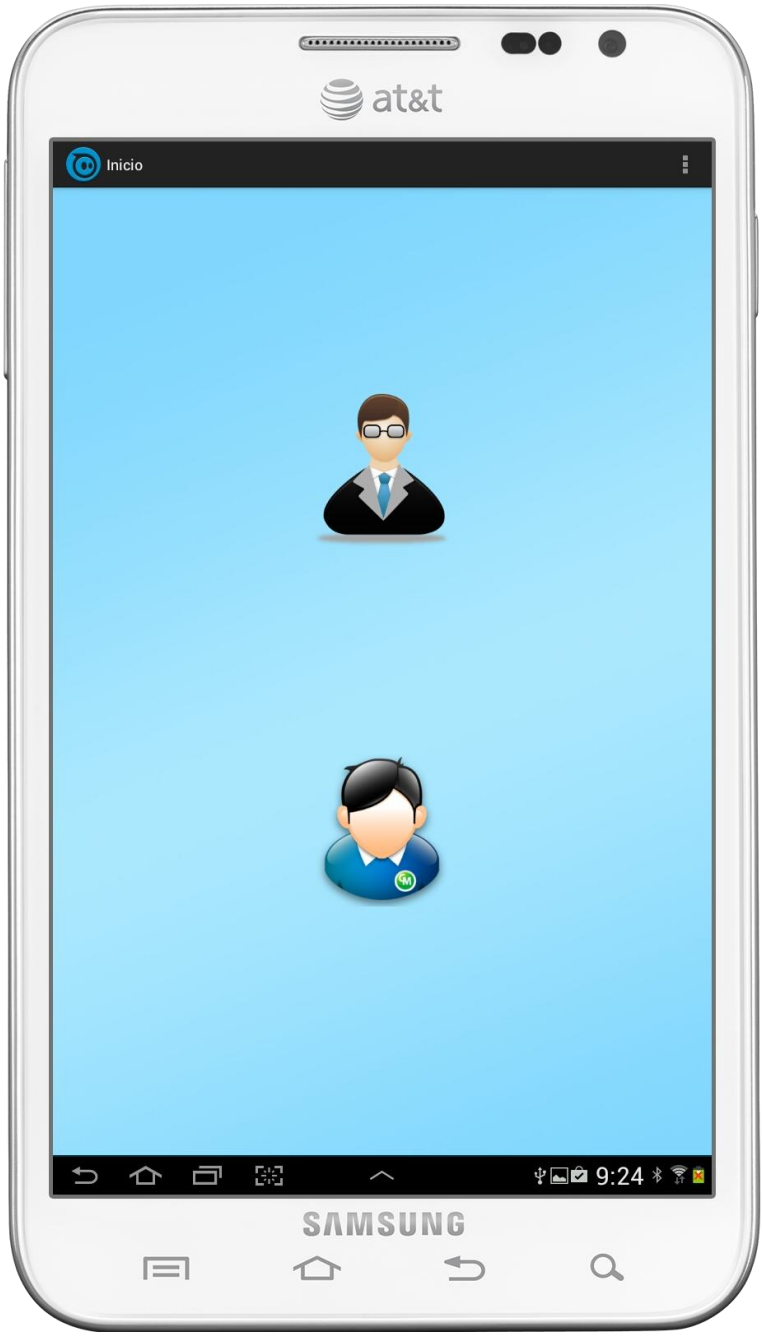


Figura 6.8: Pantalla correspondiente al Activity Main

La clase principal corresponde a la "Activity Main". Muestra dos roles, mediante iconos, uno que identifica al profesor y otro al alumno. Es la clase más sencilla, pues únicamente dependiendo de la opción que el usuario seleccione, saltará a la actividad correspondiente al rol de profesor o a la del rol alumno.

Por la parte de la vista, únicamente se utiliza un Relative Layout con dos botones en los que se carga una imagen y se les asigna un método al atributo onClick, que va a ejecutarse cuando se presione sobre cada uno de ellos.

6.2.2.2 Clase de acceso al Perfil de profesor

Figura 6.9: Pantalla correspondiente al Activity PasswordProfe

La clase corresponde a la "Activity PasswordProfe", es a la cual se accede después de que el usuario haya pulsado el icono del profesor en la actividad anterior (Main), asegura

que ningún alumno pueda acceder al menú de profesor, ya que podría des configurar algunos parámetros relevantes ya guardados.

Únicamente se pide una contraseña y se valida pulsando el botón aceptar. La contraseña se comprueba con un fichero guardado en una carpeta del dispositivo móvil, si el fichero no existe o si la contraseña es correcta, será válido el acceso y se accederá al menú de profesor. En el caso de que la contraseña fuese incorrecta, se mostrará un mensaje de error. Si se pulsara el botón cancelar se retornaría a la pantalla principal.

Por la parte de la vista, únicamente se utiliza un Relative Layout, con un TextView en el que se escribe la etiqueta, un EditText que va a posibilitar la entrada de texto del usuario, que en este caso va a ser la contraseña y dos botones en los que se carga una imagen y se les asigna un método al atributo onClick, que va a ejecutarse cuando se presione sobre cada uno de ellos.

6.2.2.3 Clase Menú del profesor

Figura 6.10: Pantalla correspondiente al Activity MenuProfesor

La clase corresponde a la "Activity MenuProfesor", es a la cual se accede después de que el usuario haya introducido la contraseña válida o no exista contraseña guardada y haya pulsado aceptar, en la actividad anterior (PasswordProfe).

En esta actividad, se tiene la posibilidad de acceder a un menú de configuración, para cambiar determinados parámetros de Sphero, Flurry, del alumno que vaya a hacer uso de los mandos y de la contraseña de acceso al menú de profesor. O bien, acceder a un menú de

selección de mandos en los que podrán añadirse a una lista, los que el profesor desee, para que el alumno pueda utilizarlos de forma posterior. Es otra clase muy sencilla, pues únicamente dependiendo del icono que el usuario toque, saltará a la actividad correspondiente a configuración o selección de mandos.

Por la parte de la vista, únicamente se utiliza un Relative Layout con dos botones en los que se carga una imagen y se les asigna un método al atributo onClick, que va a ejecutarse cuando se presione sobre cada uno de ellos.

6.2.2.4 Clase Configuración avanzada

Figura 6.11: Pantalla correspondiente al Activity ConfigAvanzada

La clase corresponde a la "Activity ConfigAvanzada", es a la cual se accede después de que el usuario haya pulsado el icono de configuración en la actividad anterior (MenuProfesor).

- Se podrá acceder a cambiar los datos del alumno que vaya a utilizar la aplicación, mediante el botón "Alumno".
- Cambiar la contraseña de acceso al menú profesor y añadir el token deFlurry, mediante el botón "Contraseña".
- Activar o desactivar la iluminación del LEDTail (Cola de Sphero),que ayudará a marcar la dirección del Sphero.
- Activar o desactivar la opción "Límite de distancia", que impedirá que un Sphero sincronizado con el dispositivo móvil se salga del rango de distancia que cubre Bluetooth.
- Asignar un valor al "Retardo de Inicio". Este parámetro sirve para que después del vencimiento de los segundos indicados en ese valor, se ejecute un mandato en el Sphero sincronizado.
- Asignar un valor al "Tiempo off". Este parámetro hace referencia al tiempo para el que después de vencerse la Sphero se apagará para no seguir consumiendo batería.
- Regular el valor de "Velocidad máxima". Mediante un Seekbar, se podrá establecer un máximo en el parámetro velocidad, del que dispondrá el alumno, una vez esté controlando un mando.

Se ha utilizado un Relative Layout con un fondo diferente en esta actividad y en las dos, a las que se enlazan mediante los botones "Alumno" y "Contraseña", para diferenciar el tipo de menú, ya que es un menú en el que se configuran parámetros, no un simple menú de acceso a opciones. Se han utilizado dos ToogleButtons, para activar y desactivar las opciones "Iluminación de la Cola" y "Límite de distancia". Otra posible solución hubiera sido utilizar un Button, ya que en ambas opciones una vez que se pulsa el botón, si está a ON, carga la imagen de un botón con el texto OFF y viceversa.

Todos los datos se guardan en la memoria del teléfono, en una carpeta determinada mediante un método. Y si se sale de la actividad y se vuelve a entrar, se recuperan, para que el usuario, pueda ver los parámetros existentes desde la última vez que se guardaron.

```
public void setParameters(String headerLight, String limDistance,
        String startDelay, String maxVel, String offTime) {

    File dir = new File(
            android.os.Environment
                    .getExternalStoragePublicDirectory(Environment.DIRECTORY_MUSIC),
            "Spheropassword");
    if (!dir.exists()) {
        dir.mkdirs();
    }
    String filename = "Parameters.txt";
    try {
        File f = new File(dir + File.separator + filename);
        FileWriter fileWriter = new FileWriter(f);
        BufferedWriter bufferWriter = new BufferedWriter(fileWriter);
        bufferWriter.write(headerLight);
        bufferWriter.write("-");
        bufferWriter.write(limDistance);
        bufferWriter.write("-");
        bufferWriter.write(startDelay);
        bufferWriter.write("-");
        bufferWriter.write(maxVel);
        bufferWriter.write("-");
        bufferWriter.write(offTime);
        bufferWriter.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

6.2.2.5 Clase Datos del alumno

Figura 6.13: Pantalla correspondiente al Activity FlurryUserData

La clase corresponde a la "Activity FlurryUserData", es a la cual se accede después de que el usuario haya pulsado el botón "Alumno" en la actividad anterior (ConfigAvanzada).

Todos los datos que se rellenen en esta pantalla, servirán para poder registrar los eventos, generados por el alumno, al pulsar cualquier botón y en general al interactuar con la aplicación, en la web de Flurry. Estos eventos aparecerán, una vez registrados en la web, ligados a esos datos:

- Nombre del alumno.
- Fecha de nacimiento. No se ha utilizado edad, para ser más precisos a la hora de registrar los datos y poder sacar de forma futura estadísticas.
- Género del alumno.

Se ha utilizado un Relative Layout con el mismo fondo que en la actividad anterior. Se han utilizado tres TextView donde se colocan las etiquetas, dos EditText, donde se permite la entrada de texto por parte del usuario y dos checkbox, donde se selecciona el género. Se ha creado un método en el impide que no se activen los dos checkbox a la vez, sólo es posible seleccionar uno. Al hacerlo, se bloquea el otro, para no generar errores a la hora de guardar los parámetros.

Todos los datos se guardan en la memoria del teléfono, en una carpeta determinada mediante un método. Y si se sale de la actividad y se vuelve a entrar, se recuperan, para que el usuario, pueda ver los parámetros existentes desde la última vez que se guardaron.

```
public void setParamsUserFlurry(String paramsFlurry) {

    File dir = new File(
            android.os.Environment
                    .getExternalStoragePublicDirectory(Environment.DIRECTORY_MUSIC),
            "Spheropassword");
    if (!dir.exists()) {
        dir.mkdirs();
    }
    String filename = "ParamsFlurry.txt";
    try {
        File f = new File(dir + File.separator + filename);

        FileWriter fileWriter = new FileWriter(f);
        BufferedWriter bufferWriter = new BufferedWriter(fileWriter);
        bufferWriter.write(paramsFlurry);
        bufferWriter.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Figura 6.14: Fragmento de código correspondiente al Activity FlurryUserData

```
public String getParamsUserFlurry() {
    File dir = new File(
            android.os.Environment
                    .getExternalStoragePublicDirectory(Environment.DIRECTORY_MUSIC),
            "Spheropassword");
    String paramsFlurry = "";
    if (!dir.exists()) {
        dir.mkdirs();
    }
    String filename = "ParamsFlurry.txt";

    try {
        File f = new File(dir + File.separator + filename);
        FileReader fileReader = new FileReader(f);
        BufferedReader bufferReader = new BufferedReader(fileReader);
        paramsFlurry = bufferReader.readLine();
        String[] parameters = paramsFlurry.split("-");
        nomUsuario.setText(parameters[0]);
        fechaUsuario.setText(parameters[1]);
        if(parameters[2].equals("masculino")){
            generoMasculino.setChecked(true);
            generoFemenino.setEnabled(false);
        }
        else{
            generoFemenino.setChecked(true);
            generoMasculino.setEnabled(false);
        }

        bufferReader.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
    return paramsFlurry;
```

6.2.2.6 Clase Cambio de contraseña y token Flurry

Figura 6.16: Pantalla correspondiente al Activity CambioPassword

La clase corresponde a la "Activity CambioPassword", es a la cual se accede después de que el usuario haya pulsado el botón "Contraseña" en la actividad (ConfigAvanzada).

El primer parámetro a rellenar, corresponde al token Flurry asignado, cuando un usuario dado de alta en la web de Flurry, quiere descargar el SDK de Flurry para integrarlo en su proyecto Android. Dicho token, es único para cada usuario de Flurry y para cada aplicación de ese usuario dado de alta. De esta forma es posible registrar eventos desde

diferentes aplicaciones que un mismo desarrollador (usuario dado de alta en Flurry) quiera registrar. Si el profesor que vaya a utilizar la aplicación no tiene cuenta en Flurry, puede dejar este campo en blanco, pero los eventos que se produzcan durante el uso de los mandos, no se registrarán en la web de Flurry.

En el caso de que el usuario, sólo quiera rellenar el token, deberá dejar vacíos los campos de texto correspondientes a la contraseña y pulsar aceptar.

Los otros tres campos de texto restantes, se utilizan, para cambiar la contraseña de acceso al perfil del profesor, en caso de que exista alguna guardada:

- Contraseña actual que exista en el sistema.
- Nueva contraseña, a elegir.
- Repetición de la nueva contraseña, para no validar una contraseña introducida por error.

Se ha utilizado un Relative Layout con el mismo fondo que en la actividad anterior. Se han utilizado cuatro TextView donde se colocan las etiquetas de los campos de entrada de texto, otros dos TextView a modo de tíituloy cuatro EditText, donde se introducen los datos indicados por las etiquetas.

Todos los datos se guardan en la memoria del teléfono, en una carpeta determinada mediante un método. Y si se sale de la actividad y se vuelve a entrar, se recupera el valor del token guardado, para que el usuario, pueda ver el parámetro existente desde la última vez que se guardó.

6.2.2.7 Clase Menú selección de mandos

Figura 6.17: Pantalla correspondiente al Activity MenuSelecMandos

La clase corresponde a la "Activity MenuSelecMandos", es a la cual se accede después de que el usuario haya pulsado el icono de "Selección de Mando" en la actividad (MenuProfesor).

Esta actividad hace uso de la clase "SlideHolder", que implementa un menú swipe, en el que se establecen dos layouts, uno a la izquierda desplegable y otro a la derecha fijo.

Se modeló de tal forma que pudiesen verse ambos menús a la vez, uno con una vista en miniatura, correspondiente a los cuatro mandos, y otro que contiene una imagen dinámica (ImageView). Según la imagen tocada (mandos) en el layout izquierdo, hace que se cargue en el layout derecho, esa misma imagen con una serie de detalles.

Si el usuario (profesor), decide añadir una imagen a la lista de mandos, que va a disponer el alumno posteriormente, basta con presionar el botón (ToogleButton) "No Añadido" en el menú derecho. Dicho botón cambiará al ser tocado con el texto "Añadido" y con una luz encendida.

Cuando se desee guardar la lista de mandos, se deberá pulsar el botón "Guardar", y quedará guardada en la tarjeta de memoria. Si el profesor decide hacer algún cambio, podrá volver a entrar al menú de selección de mandos y según vaya tocando los iconos de los mandos, sabrá según el botón de "Añadido"/"No Añadido" si están añadidos ya o no.

En el siguiente fragmento de código se indica, la forma en la que se va añadiendo mandos a la lista o se eliminan de la lista los mandos seleccionados. El método se basa en obtener del ToogleButton su nombre identificativo, si es igual a "No Añadido" se añade dicho mando a la lista, si por el contrario estaba añadido y se ha presionado de nuevo el ToogleButton, que en este momento contendrá como nombre identificativo "Añadido", se quitará de la lista y el ToogleButton, retornará al estado "No Añadido".

```
case R.id.toggleButtonCheck:

    if (toogleButtonCheck.getText().equals(
            getResources().getText(R.string.ToogleButtonAON))) {
        selectedControllers = selectedControllers + addController;
        checkActualButtonToogle(addController);
    }
    if (toogleButtonCheck.getText().equals(
            getResources().getText(R.string.ToogleButtonAOFF))) {
        if (selectedControllers.contains(addController)) {

            selectedControllers = selectedControllers.replace(
                    addController, "");
        }
    }
    break;
```

6.2.2.8 Clase Lanzador de mando

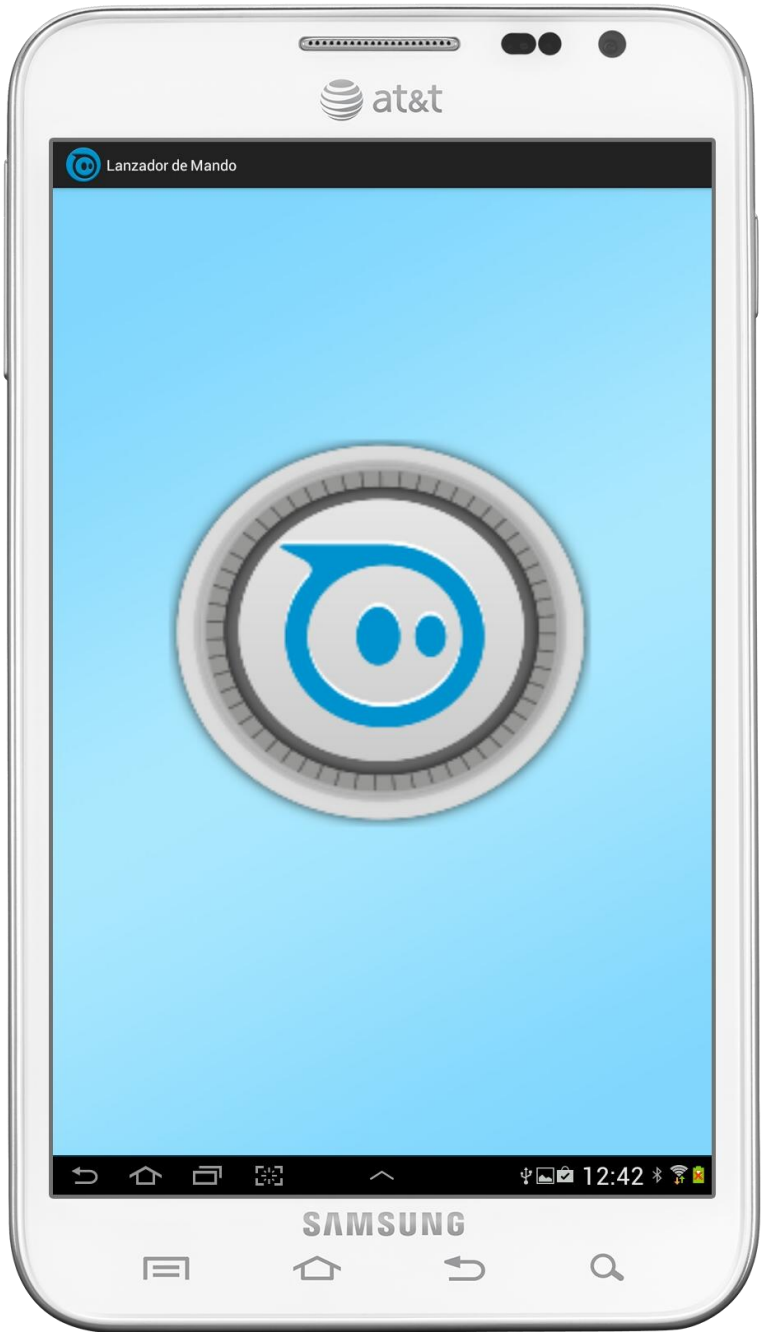


Figura 6.19: Pantalla correspondiente al Activity LaunchController

La clase corresponde a la "Activity LaunchController", es a la cual se accede después de que el usuario haya pulsado el icono de "Alumno" en la actividad (Main).

Mediante esta actividad, se producirá la sincronización con un Sphero, una vez que se haya sincronizado, va a obtener del Sphero un ID único, que posteriormente se utilizará para nuevas sincronizaciones cuando el usuario tenga que pasar de un mando a otro. Se almacenará en la memoria del dispositivo móvil, para recuperarla cuando interese.

```
public void onRobotConnected(Robot robot) {
    mRobot = robot;
    mSpheroConnectionView.setVisibility(View.GONE);
    setUniqueID(mRobot.getUniqueId());
    launchController();

}
```

Dentro de ese método, una vez que se detecta, que hay un Sphero sincronizado, se obtiene el ID, como se comentó anteriormente y se lanza un mando.

En esta actividad, se comprueba además la validez del token Flurry guardado en la clase "CambioPassword" (6.2.2.6). Si el token es válido, se iniciará una sesión Flurry, mediante el método onStartSession, contenido en el método onStart. Una vez iniciada la sesión se enviarán los datos del alumno registrados en la actividad "FlurryUserData".

Además la sesión concluirá cuando el usuario presione el botón "Atrás/Back" del propio terminal.

Según los mandos guardados por el profesor en la actividad "MenuSelecMandos", se cargará uno u otro. Y una vez cargado un mando, si el profesor añadió más de uno a la lista, el alumno podrá cambiar al siguiente mando, como se verá a continuación.

6.2.2.9 Clase Mando joystick

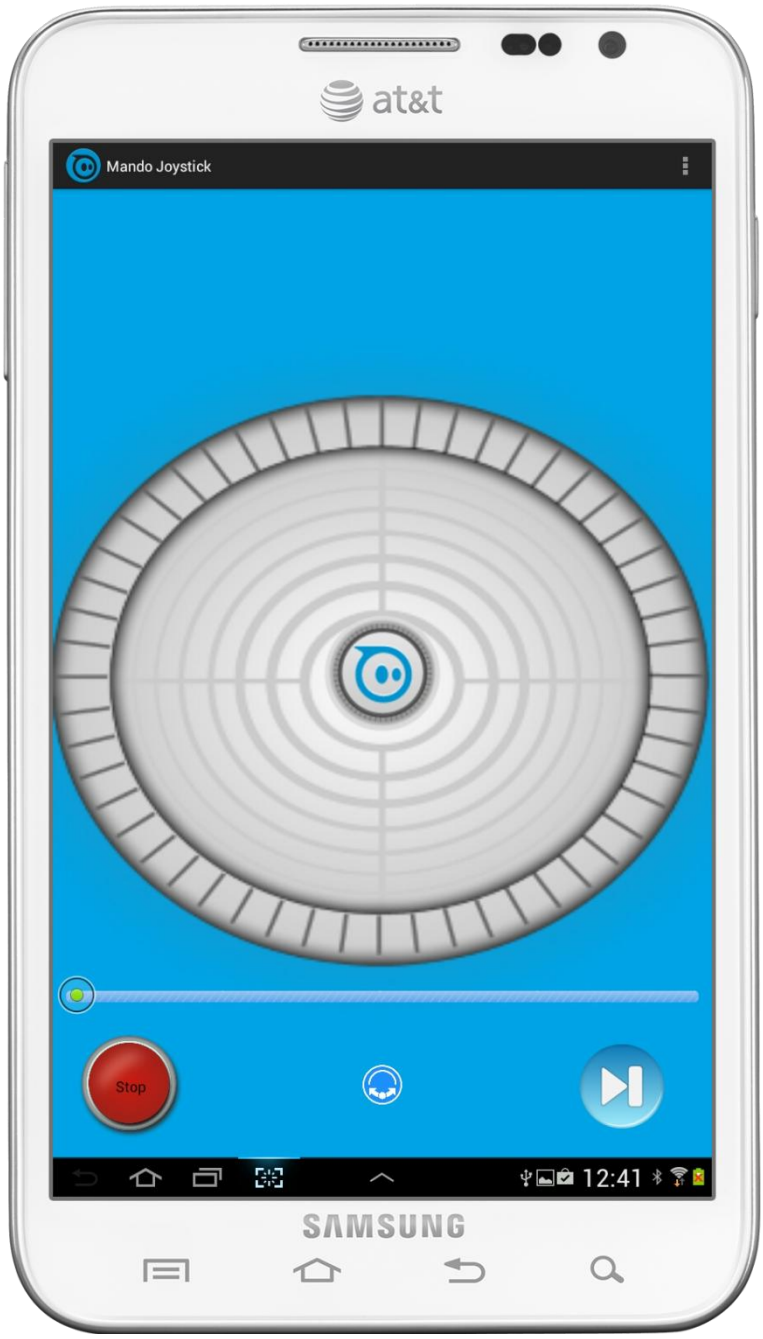


Figura 6.20: Pantalla correspondiente al Activity MJoystick

La clase corresponde a la "Activity MJoystick", corresponde al primer mando de la lista, que ha sido seleccionado por el profesor en la actividad "MenuSelecMandos" y que ha sido lanzado desde la actividad "LaunchController".

Dicho mando, contiene:

- Una zona circular, en la parte superior, que representa, la región por la que el Joystick será desplazado.
- El joystick, representado por el icono e insignia de Sphero.

- Un control de velocidad, en el que es posible seleccionar, desde el 0% al 100%, dentro de un margen de velocidad ya establecido por el por el profesor en la actividad "ConfigAvanzada".
- Un botón de parada, el cual, tras ser pulsado, envía a Sphero un mensaje indicando que debe detenerse inmediatamente. Al pulsarse, dicho botón cambia de estado y aparece con el aspecto de que ha sido pulsado.
- Un botón que permite calibrar la dirección de un Sphero. La función está contenida en el API de Sphero y simplemente invocando al método y añadiendo ese elemento el layout es posible utilizarlo. Permite rotar un Sphero sobre sí mismo para situar la cola en la posición deseada. Eso permite, tener un punto de referencia, a partir del cual la Sphero se desplazará siguiendo un ángulo de dirección.
- Un botón con una flecha hacia la derecha, que posibilita, el cambio al siguiente mando de la lista, seleccionado por el profesor. Si no existe ninguno más guardado, no hay posibilidad de cambiar de mando.
- Un menú de opciones, que contiene las opciones de mostrar el menú de ayuda, o lanzar el siguiente mando.

  Es importante destacar, que se ha implementado de tres maneras diferentes el cambio al siguiente mando:

  1. Mediante el botón descrito, que aparece en la pantalla con una flecha hacia la derecha.
  2. Mediante el botón "Subir volumen" del lateral del dispositivo móvil.
  3. Mediante el icono, de la parte superior derecha de la pantalla, que desplegará el Menú Bar y que, contendrá entre otras opciones, la opción de "Siguiente Mando".

Esto es importante, ya que, puede haber usuarios con problemas a la hora de pulsar de una forma, como consecuencia de la discapacidad, por ello, con tres alternativas, se favorece la posibilidad de cambio de mando.

Uno de los requisitos funcionales de la aplicación, era que existiera la posibilidad de utilizar un mando en el que se simulara la interacción mediante un joystick. Por ello, se pensó en una de las posibilidades que ofrece Android, y es, utilizar el Layout de fondo de pantalla (en este caso RelativeLayout), como un listener, que cada vez que recibiera una notificación de que se ha tocado una zona del layout, posicionase el icono distintivo del joystick en ese punto y además, dependiendo de la zona en la que se hubiese producido, se enviara al Sphero sincronizado hacia una dirección determinada, mediante un algoritmo desarrollado, para el cual se calcula los grados de desviación con los que hay que dirigir a la cabeza del Sphero.

Mediante ese listener, se obtiene la posición tocada, mediante dos coordenadas (X,Y); se calcula posteriormente el grado de desviación, que se almacenará en el atributo "heading",

se pinta el distintivo del joystick en ese punto, para dar sensación al usuario de que está moviendo el joystick y finalmente se envía un mandato al Sphero sincronizado con el valor de la dirección que debe tomar y la velocidad.

```
relativeLayout.setOnTouchListener(new View.OnTouchListener() {

    @Override
    public boolean onTouch(View v, MotionEvent event) {

        valueOfX = event.getX();
        valueOfY = event.getY();

        heading = getHeadingValue(valueOfX, valueOfY);

        joystick.setAdjustViewBounds(true);
        joystick.setX(valueOfX);
        joystick.setY(valueOfY);

        // Roll robot
        RollCommand.sendCommand(mRobot, heading, allowedVelocity);
        FlurryAgent.logEvent("Ha usado el Joystick en dirección: "
                + heading, userParams);
        return true;

    }
});
```

Se produjo una gran dificultad a la hora de pasar de una actividad a otra, con un Sphero sincronizado. Ya que el fabricante Orbotix, creó un API que no contemplaba que fuese a estar sincronizado un dispositivo Sphero y el usuario quisiese mantener la sincronización mientras cambiaba de una actividad a otra. Por ello, en la actividad "LaunchController", cuando se produce la primera sincronización, se realizó el guardado del ID del dispositivo, que corresponde con su MAC, en la memoria del terminal, para luego recuperar dicho parámetro cuando se desee y restablecer la sincronización.

A continuación, se muestra el método, con el que se consiguió restablecer la sincronización, cuando se pasaba de un mando a otro:

```
if (robot_id != null && !robot_id.equals("")) {
    RobotProvider provider = RobotProvider.getDefaultProvider();
    mRobot = provider.findRobot(robot_id);
    provider.initiateConnection(robot_id);
    provider.control(mRobot);
    provider.connectControlledRobots();
}
```

El atributo “robot_id” corresponde al valor del ID único del dispositivo Sphero. Mediante una serie de métodos del API proporcionado por Orbotix, se consiguió restablecer la conexión, es algo que ningún usuario había conseguido.

Se ha añadido una funcionalidad importante y es la de avisar al usuario que esté utilizando el mando, de colisión del Sphero con una pared o cualquier otro obstáculo. Esto, se le notifica mediante un aviso en forma de vibración en el dispositivo móvil. También se notifica en forma de vibración, cuando el usuario está realizando una calibración de dirección.

En la siguiente figura, se muestra un ejemplo del uso del calibrador de dirección. Es posible utilizarlo con un dedo, tocando el icono de calibración, o con dos dedos tocando sobre cualquier zona de la pantalla:

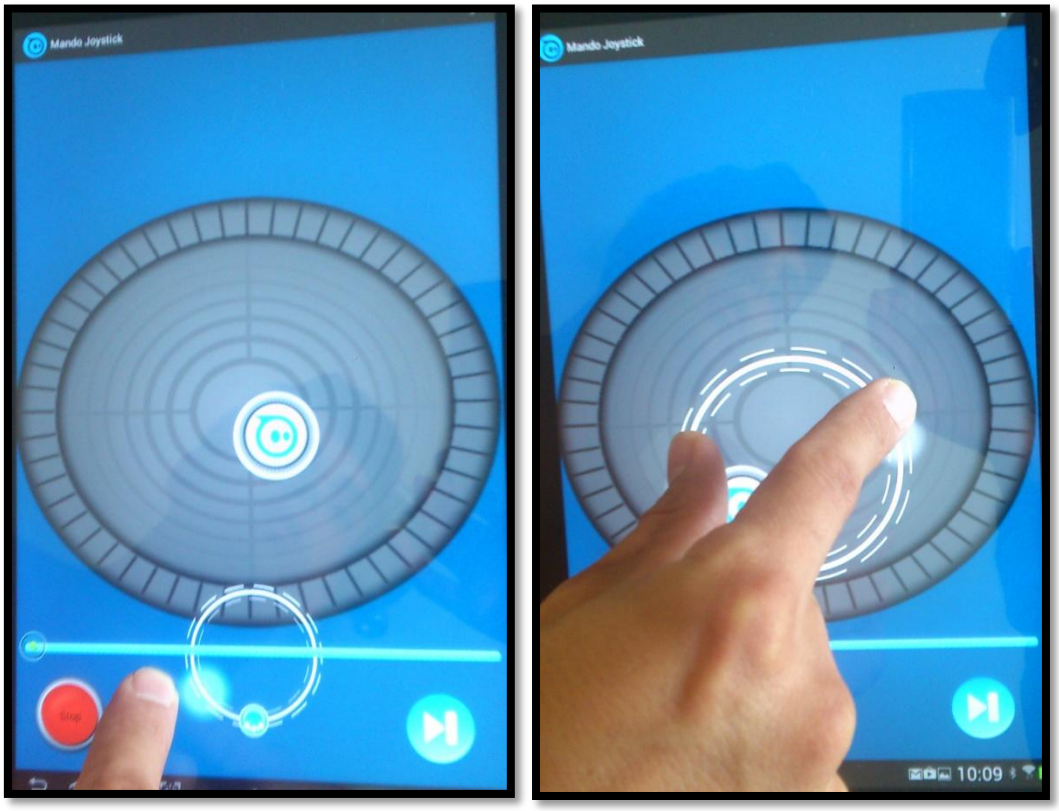


Figura 6.21: Capturas de pantalla del uso del Calibrador de dirección

Cabe de destacar, que, todo el comportamiento del usuario que esté utilizando el mando se registrará, haciendo uso de los eventos Flurry.

En este caso, se registrarán para un perfil de usuario registrado (nombre, fecha de nacimiento y género):

- Modificación de la velocidad.

- Uso del joystick y en qué dirección (dada en grados).
- Uso del botón de parada “Stop”.
- Uso del botón “Siguiente Mando”.
- Consulta del “Menú de Ayuda”.

6.2.2.10 Clase Mando figuras predeterminadas

Figura 6.22: Pantalla correspondiente al Activity Mfiguras

La clase corresponde a la “Activity Mfiguras”, corresponde al segundo mando de la lista, que ha sido seleccionado por el profesor en la actividad “MenuSelecMandos” y que

puede haber sido lanzado desde la actividad “LaunchController” o desde un mando anterior, mediante el botón “Siguiente Mando”.

Dicho mando, contiene:

- Un botón con una figura pre configurada, que al activarse, hace que el Sphero dibuje una trayectoria en forma de cuadrado, en la superficie en la que se encuentre.
- Un botón que al ser activado, produce una vibración en el Sphero y proyecta en él una luz roja.
- Un botón que al ser activado, hace que el Sphero realice un movimiento rotatorio sin desplazarse en la superficie.
- Un botón con una figura pre configurada, que al activarse, hace que el Sphero dibuje una trayectoria en forma de ocho, en la superficie en la que se encuentre.
- Un botón que al ser activado, hace que el Sphero realice un movimiento al que se ha identificado como ” tornado”, porque realiza movimientos muy bruscos, con poco desplazamiento pero a gran velocidad y rotando sobre sí mismo.
- Un botón que al ser activado, hace que el Sphero cambie de color de forma progresiva.
- Un botón con una figura pre configurada, que al activarse, hace que el Sphero dibuje una trayectoria en forma de elipses entrelazadas.
- Un botón de parada, que una vez pulsado, hace que el Sphero se pare si está realizando cualquier función de los otros botones.
- Un botón con una flecha hacia la derecha, que posibilita, el cambio al siguiente mando de la lista, seleccionado por el profesor. Si no existe ninguno más guardado, no hay posibilidad de cambiar de mando.
- Un menú de opciones, que contiene las opciones de mostrar el menú de ayuda, o lanzar el siguiente mando.

Como en el mando anterior, todo el comportamiento del usuario que esté utilizando el mando se registrará, haciendo uso de los eventos Flurry.

En este caso, se registrarán para un perfil de usuario registrado (nombre, fecha de nacimiento y género):

- Cualquiera de las figuras definidas
- Uso del botón de parada “Stop”.
- Uso del botón “Siguiente Mando”.
- Consulta del “Menú de Ayuda”.

6.2.2.11 Clase Mando porciones

Figura 6.23: Pantalla correspondiente al Activity MPorciones

La clase corresponde a la "Activity MPorciones", corresponde al tercer mando de la lista, que ha sido seleccionado por el profesor en la actividad "MenuSelecMandos" y que puede haber sido lanzado desde la actividad "LaunchController" o desde un mando anterior, mediante el botón "Siguiente Mando".

Dicho mando, contiene:

- Dieciséis botones que marcan la trayectoria del Sphero. Cada uno representado con color diferente y con un grado de desviación memorizado.
- Un botón de parada, que una vez pulsado, hace que el Sphero se pare si está realizando cualquier función de los otros botones.
- Un control de velocidad, en el que es posible seleccionar, desde el 0% al 100%, dentro de un margen de velocidad, ya establecido por el por el profesor en la actividad "ConfigAvanzada".
- Un botón con una flecha hacia la derecha, que posibilita, el cambio al siguiente mando de la lista, seleccionado por el profesor. Si no existe ninguno más guardado, no hay posibilidad de cambiar de mando.
- Un menú de opciones, que contiene las opciones de mostrar el menú de ayuda, o lanzar el siguiente mando.

Uno de los requisitos funcionales de la aplicación, era que existiera la posibilidad de desarrollar un mando en que se dividiera toda la pantalla en botones, de esta forma, el área de pulsación para el usuario, es completa y en cualquier zona que pulse, la pelota actuará de una forma diferente.

A día de hoy, no existe ninguna aplicación en Android, en la que se haya divido una pantalla en 16 botones en la disposición que se ha utilizado en esta aplicación, por lo que, se pensó en una de las posibilidades que ofrece Android, y es, utilizar el layout de fondo de pantalla (en este caso RelativeLayout) de forma dinámica, como un listener, que cada vez que recibiera una notificación de que se ha tocado una zona del layout, se obtuviese información acerca del pixel que se ha tocado. Dentro de esa información, interesaba obtener el color de ese pixel e iba a ser usado para compararlo con el color de cualquiera de los dieciséis botones triangulares. Una vez identificado el botón al que corresponde ese color, se cargaría una imagen nueva en ese layout, en la que se mostraría toda la pantalla exactamente igual, menos, ese botón que aparecería con aspecto de pulsado. Además al identificar el botón al que pertenece ese color del pixel, se enviaría un mandato al Sphero para desplazarse en una dirección determinada, según un algoritmo desarrollado, que obtiene el grado de desviación.

Igual que en la actividad (MJoystick), se ha añadido:

- Una funcionalidad extra, la cual consiste en avisar al usuario que esté utilizando el mando, de colisión del Sphero con una pared o cualquier otro obstáculo. Esto, se le notifica mediante un aviso en forma de vibración en el dispositivo móvil. También se notifica en forma de vibración, cuando el usuario está realizando una calibración de dirección.

- El calibrador de dirección. Es posible utilizarlo con un dedo, tocando el icono de calibración, o con dos dedos tocando sobre cualquier zona de la pantalla.

Y como en los demás mandos, todo el comportamiento del usuario que esté utilizando el mando se registrará, haciendo uso de los eventos Flurry.

En este caso, se registrarán para un perfil de usuario registrado (nombre, fecha de nacimiento y género):

- Uso de cualquiera de los dieciséis botones de dirección.
- Uso del botón de parada "Stop".
- Uso del botón "Siguiente Mando".
- Consulta del "Menú de Ayuda".
- Modificación de la velocidad.

Para la adecuada obtención del pixel, lo primero había que habilitar el caché del layout que utilizaba como fondo:

```
relativeLayout = (RelativeLayout) findViewById(R.id.relativeLayout);
relativeLayout.setDrawingCacheEnabled(true);
bitmap = getBitmapFromView(relativeLayout);
relativeLayout.setDrawingCacheEnabled(false);
```

Mediante el método getBitmapFromView, se obtiene el bitmap, del cual sacaré el color, gracias a que se puede conseguir en forma de RGBColor:

```
public Bitmap getBitmapFromView(RelativeLayout relativeLayout) {
    DisplayMetrics dm = this.getApplicationContext().getResources()
            .getDisplayMetrics();
    relativeLayout.measure(MeasureSpec.makeMeasureSpec(dm.widthPixels,
            MeasureSpec.EXACTLY), MeasureSpec.makeMeasureSpec(
            dm.heightPixels, MeasureSpec.EXACTLY));
    relativeLayout.layout(0, 0, relativeLayout.getMeasuredWidth(),
            relativeLayout.getMeasuredHeight());
    relativeLayout.setGravity(Gravity.CENTER);
    Bitmap returnedBitmap = Bitmap.createBitmap(
            relativeLayout.getMeasuredWidth(),
            relativeLayout.getMeasuredHeight(), Bitmap.Config.ARGB_8888);
    Canvas c = new Canvas(returnedBitmap);
    relativeLayout.draw(c);
    return returnedBitmap;
}
```

```
relativeLayout.setOnTouchListener(new View.OnTouchListener() {
    @Override
    public boolean onTouch(View v, MotionEvent event) {

        int x = (int) event.getX();
        int y = (int) event.getY();
        int pixel = bitmap.getPixel(x, y);

        int redValue = Color.red(pixel);
        int blueValue = Color.blue(pixel);
        int greenValue = Color.green(pixel);
        heading = getHeadingValue(redValue, blueValue, greenValue);

        // Roll robot
        RollCommand.sendCommand(mRobot, heading, allowedVelocity);
        return false;
    }
});
```

Una vez se hayan obtenido los tres valores del RGBColor (rojo, verde y azul), mediante un método (getHeadingValue(…)), se obtendrá el valor de desviación de dirección, que posteriormente se mandará al Sphero mediante el método (sendCommand).

6.2.2.12 Clase Mando paleta de colores

Figura 6.24: Pantalla correspondiente al Activity MPaleta

La clase corresponde a la "Activity MPaleta", corresponde al cuarto mando de la lista, que ha sido seleccionado por el profesor en la actividad "MenuSelecMandos" y que puede haber sido lanzado desde la actividad "LaunchController" o desde un mando anterior, mediante el botón "Siguiente Mando".

Dicho mando, contiene:

- Un layout con una imagen que presenta un amplio abanico de tonalidades.

- Un botón con una flecha hacia la derecha, que posibilita, el cambio al siguiente mando de la lista, seleccionado por el profesor. Si no existe ninguno más guardado, no hay posibilidad de cambiar de mando.
- Un menú de opciones, que contiene las opciones de mostrar el menú de ayuda, o lanzar el siguiente mando.

La peculiaridad de este mando es que, se ha vuelto a utilizar la propiedad del layout que permite usarlo como un listener y de esa forma hacer que cada vez que se detecte que se ha tocado, pueda obtenerse qué color exacto corresponde a ese pixel tocado y enviarlo al Sphero para que lo refleje con sus LEDs en tiempo real, sin demoras. Gracias a que Sphero, es capaz de proyectar hasta 16 millones de colores debido a la cantidad de LEDs que contiene, se presenta un mando sencillo y de un uso fácil, que no requiere ninguna habilidad especial para pulsar, ya que carece de importancia la zona que se toque, pues siempre se va a reflejar algún color en el Sphero.

Para llevar a cabo esta función de proyección de un color en tiempo real, se ha reutilizado el método del mando anterior (6.2.2.11 MPorciones), para obtener el Bitmap asociado a la zona del layout tocada y posteriormente se descompone ese bitmap en los tres colores que componen el RGB, para así ser introducidos como pará metro en el método propio de la API de Sphero para cambiar el color de dicho dispositivo:

```
relativeLayout.setOnTouchListener(new View.OnTouchListener() {

    @Override
    public boolean onTouch(View v, MotionEvent event) {

        int x = (int) event.getX();
        int y = (int) event.getY();
        int pixel = bitmap.getPixel(x, y);

        int redValue = Color.red(pixel);
        int blueValue = Color.blue(pixel);
        int greenValue = Color.green(pixel);
        // Set the color
        RGBLEDOutputCommand.sendCommand(mRobot, redValue, greenValue,
                blueValue);

        return true;
    }

});
```

Y como en los demás mandos, todo el comportamiento del usuario que esté utilizando el mando se registrará, haciendo uso de los eventos Flurry.

En este caso, se registrarán para un perfil de usuario registrado (nombre, fecha de nacimiento y género):

- Uso del botón “Siguiente Mando”.
- Consulta del “Menú de Ayuda”.

No se ha incluido un tipo de evento para los colores tocados, porque, existe la posibilidad de que un usuario esté deslizando el dedo sobre la pantalla, por lo que, se enviarían centenares de eventos, informando del color del pixel utilizado en cada momento.

6.2.2.13 Clase Menú ayuda

Figura 6.25: Pantalla correspondiente al Activity MenuAyuda

La clase corresponde a la "Activity MenuAyuda", corresponde a una actividad utilizada cuando se pulsa la opción "Ayuda", del menú de opciones:

Figura 6.26: Menú de opciones

Contiene un layout dinámico, que dependiendo de la actividad desde la que se lance, mostrará un layout u otro. Para identificar desde qué mando se lanzó este menú de ayuda, se pasan mediante la forma que existe en Android de paso de flujo de datos entre actividades. A continuación se muestra el paso de parámetros necesarios, desde el mando Joystick, para cargar el layout específico en el menú ayuda:

```
intent = new Intent();
intent.setClass(MJoystick.this, MenuAyuda.class);
intent.putExtra("nameLayout",
        getResources().getResourceName(R.layout.ayudajoystick));
intent.putExtra("idLayout",
        getResources().getResourceName(R.id.linearAyudaJoystick));
intent.putExtra("nameController", "Joystick");
```

Y se recuperan en el menú ayuda:

```
Intent intent = getIntent();
String namelayout = intent.getStringExtra("nameLayout");
String idlayout= intent.getStringExtra("idLayout");
nameController= intent.getStringExtra("nameController");
```

Según esos parámetros, se cargará un layout determinado. Lo importante, es que se plasmará exactamente encima del mando la descripción de cada botón.

Se ha hecho de esta forma, para no tener que crear una actividad de menú de ayuda por cada mando que hay en la aplicación.

Además este layout contiene un listener, que permitirá volver al mando desde el cual se consultó esta opción de ayuda, tocando en cualquier punto de la pantalla.

Cuando se consulte este menú de ayuda, se enviará un evento Flurry, para registrar que ha sido consultado.

CONCLUSIONES

Este proyecto, surgió con la idea de fomentar la interacción con usuarios que presentasen una discapacidad y poco a poco se han ido ampliando los objetivos. De los cuales, se han podido cumplir todos.

- Realización de cuatro mandos, con diferentes niveles de dificultad de manejo.
- Realización de un menú con opciones avanzadas, que permitan configurar la aplicación particularizando al usuario con discapacidad que vaya a utilizarla.
- Implementar mecanismos que permitan al usuario tener el control del juguete en todo momento, sin que se vea comprometida su discapacidad.
- Realizar un sistema, por el cual, se consiga registrar todo el comportamiento del usuario, durante el tiempo de uso de la aplicación. Con el fin de llevar un seguimiento del mismo; evaluar si existen cambios en él; determinar si son necesarios algunos cambios en la aplicación que favorezcan una mejoría en cuanto al uso y consecución de resultados en el paciente, etc.

Una vez descritos, es posible enunciar una serie de mejoras que constituirían una nueva versión de la aplicación, que podrían implementarse en futuras proyectos y complementarían al proyecto presente:

- Realización de una interfaz de usuario sonora utilizando cualquier paquete compatible con Android, como por ejemplo “android.speech”. Ya que cualquier mandato realizado por voz, bastaría para interactuar con un Sphero, sin necesidad de tocar y ver la pantalla del dispositivo Tablet PC.
- Realización de un mando para controlar un Sphero, que únicamente utilice el acelerómetro del dispositivo Tablet PC para dirigir en las diferentes direcciones.
- Realización de una interfaz de rápido y fácil acceso en la que puedan observarse los eventos guardados en la web de Flurry. Bien utilizando un WebView de Android o creando una vista específica. De esta forma, el usuario con rol de profesor, podría consultar los resultados, sin tener que salir de la aplicación y ver los eventos en el formato deseado.
- Implementación de un Service, que corra en segundo plano en todo momento, capaz de crear, guardar y liberar una conexión WIFI con un Sphero, al arrancar la aplicación, sin necesidad de volver a sincronizarlo.
- Crear una interfaz de usuario, que permita en varios pasos, entender el funcionamiento de Sphero y verificar su correcta sincronización, siguiendo el modelo de la aplicación del fabricante Orbotix.
- Crear un mando, capaz de, a partir de la figura dibujada en el layout por el usuario que esté utilizando, enviarla como trayectoria al Sphero, para describirla en la superficie en la que se encuentre.
- Crear un layout, que aprovechando las propiedades de Sphero, se convierta en una pizarra, en la que el usuario pueda dibujar moviendo la Sphero con la mano a modo de ratón.

BIBLIOGRAFÍA

❖ **Páginas Web Consultadas:**

- Historia de Android:

  http://es.wikipedia.org/wiki/Android

  http://www.slideshare.net/JoseDavidLeon/historia-de-android

- Evolución de Android:

  http://www.elandroidelibre.com/2013/06/especial-la-evolucion-de-android-desde-sus-inicios-hasta-ahora-1a-parte.html

  http://www.elandroidelibre.com/2013/06/especial-la-evolucion-de-android-de-froyo-a-jelly-bean-un-dulce-presente-y-futuro.html

- Dalvik:

  http://androideity.com/2011/07/07/la-maquina-virtual-dalvik/

- Ciclo de vida de actividades

  http://elbauldelprogramador.com/opensource/fundamentos-programacion-android-ciclo/

- Arquitectura

  http://androideity.com/2011/07/04/arquitectura-de-android/

- Flurry Analytics

  http://support.flurry.com/index.php?title=Analytics/Overview

- Comunicación, definición:

  http://www.definicionabc.com/comunicacion/comunicacion.php

- Elementos de la comunicación:

  http://recursos.cnice.mec.es/lengua/profesores/eso1/t1/teoria_1.htm

- Factores que limitan la comunicación:

  http://cangurorico.com/2009/02/barreras-en-la-comunicacion.html

- Trastornos en la comunicación

  http://diversidad.murciaeduca.es/orientamur/gestion/documentos/unidad23.pdf

- Discapacidad psíquica

  http://multiblog.educacion.navarra.es/iibarrog/diversidad/discapacidad-psiquica/

- Discapacidad motora

  http://soleyma.wordpress.com/discapacidades/discapacidad-motora-2/

- Discapacidad visual

  http://www.news-medical.net/health/Types-of-visual-impairment-%28Spanish%29.aspx

- Juguetes adaptados

  http://www.bj-adaptaciones.com

- Sistemas aumentativos SAAC

  http://www.catedu.es/arasaac/aac.php

- Teléfonos adaptados para mayores

  http://www.emporia.eu/es/productos/vista-general/emporiacareplus

- Flurry Analytics

  http://www.flurry.com/flurry-analytics.html

❖ **Documentación Android consultada para implementación de la aplicación:**

http://developer.android.com

http://stackoverflow.com/

https://github.com