*Aplicación Android para mejorar el rendimiento deportivo en piragüismo*

Trabajo de Fin de Grado

INGENIERÍA INFORMÁTICA

**Septiembre 2018**

**2017/2018**

**Autor:**

*Pablo Rubio Machacón*

**Tutores:**

*Guillermo González Talaván*

*Alberto Rodríguez Valle*

D. Guillermo González Talaván, profesor del Depto. De Informática y Automática de la Universidad de Salamanca y D. Alberto Rodríguez Valle, Director deportivo del Club Salamanca Canoe Kayak

DAN FE DE:

Que el trabajo titulado “Aplicación Android para mejorar el rendimiento deportivo en piragüismo” ha sido realizado por Pablo Rubio Machacón bajo su dirección y hacen constar que las aportaciones realizadas en este trabajo son suficientes, en su opinión, para presentarse a la superación de la asignatura Proyecto de Fin de Carrera de la titulación de Ingeniería Informática de la Universidad de Salamanca.

Y para que conste a todos los efectos oportunos lo firman en Salamanca a 6 de Septiembre de 2018.

|  |  |
| --- | --- |
|  |  |
| Fdo: Guillermo González Talaván | Fdo: Alberto Rodríguez Valle |

**Resumen y palabras clave**

* Resumen:

El Club Salamanca Canoe Kayak (CSCK) es un club de piragüismo situado a orillas del Tormes donde palean personas de todas las edades. Entre sus objetivos podemos mencionar la participación del club en campeonatos nacionales.

Este proyecto pretende desarrollar una aplicación que ayude a los palistas del CSCK por medio de un dispositivo Android aprovechando los diferentes sensores y equivalentes de que disponen los teléfonos móviles (GPS, acelerómetro, giroscopio, etc.) para conseguir datos físicos de los entrenamientos y así poder estudiarlos y tener un control sobre ellos. Estos datos pueden servir para poder tomar decisiones de entrenamiento encaminadas a mejorar el rendimiento deportivo de los deportistas.

* Palabras clave:

Java

Android Studio

Frecuencia de paladas

Interfaz

Servicio

OpenGL

Piragüismo

Rendimiento deportivo

Entrenamientos

Palista

**Summary and keywords**

* Summary:

The Club Canoe Kayak Salamanca (CSCK) is a canoeing club located on the banks of the river Tormes where people of all ages paddle, among its objectives we can mention the participation of the club in national championships.  
This project aims to develop an application that helps the CSCK paddlers with an Android device monitor their training sessions, taking advantage of the different sensors and equivalents that mobile phones have (GPS, accelerometer, gyroscope, etc.). The physical data retrieved can be used to study and have control over training, in order to make decisions aimed at improving athletes' sports performance.

* Keywords:

Java

Android Studio

Stroke frequency

Interface

Service

OpenGL

Kayaking

Sport performance

Trainings

Kayaker

**Tabla de contenidos**

1. Introducción……………………………………………………………………………………….……………… 8
2. Objetivos…………………………………………………………………………………………….……………. 10
3. Conceptos teóricos y tecnologías empleadas……………………………………….…..……... 11
   1. Android…………………………………………………………………………………..……………..….. 11
      1. Historia…………………………………………………………………………..……………… 11
      2. Arquitectura………………………………………………………………………………….. 12
      3. Aplicaciones………………………………………………………………………………….. 16
         1. Componentes…………………………………………………………………………. 16
         2. Recursos…………………………………………………………………………………. 20
         3. Manifiesto………………………………………………………………………………. 20
      4. Android Studio………………………………………………………………………………. 21
      5. SDK – Software Development kit…………………………………………………… 22
      6. NDK – Native Development Kit……………………………………………………… 22
   2. OpenGL……………………………………………………………………………………………………… 22
   3. Tecnologías empleadas para la obtención de medidas………………………………. 23
      1. Acelerómetro………………………………………………………………………………... 23
      2. GPS………………………………………………………………………………………………… 24
      3. Wifi-Direct…………………………………………………………………………………..... 25
      4. Lenguajes………………………………………………………………………………………. 25
         1. Java…………………………………………………………………………………………. 25
         2. XML…………………………………………………………………………………………. 26
   4. Herramientas CASE…………………………………………………………………………………….. 27
      1. Lenguaje Unificado de modelado (UML)………………………………………... 27
4. Obtención de medidas y objetivos …………………………………………………………………… 28
   1. Distancia y velocidad………………………………………………………………………………….. 28
   2. Frecuencia de paladas………………………………………………………………………………… 29
   3. Entrenamientos………………………………………………………………………………………….. 35
   4. Gestor de entrenamientos………………………………………………………………………….. 36
   5. Gráficos de los entrenamientos………………………………………………………………….. 37
   6. Crear entrenamientos personalizados………………………………………………………… 38
   7. Compartir entrenamientos con el entrenador de forma inalámbrica………….. 39
5. Aspectos relevantes de desarrollo…………………………………………………………………….. 40
   1. Descripción general…………………………………………………………………………………….. 40
   2. Ingeniería del software……………………………………………………………………………….. 47
      1. Metodología…………………………………………………………………………………… 47
      2. Planificación temporal………………………………………………………………….... 47
      3. Especificación de requisitos……………………………………………………………. 47
      4. Análisis del sistema…………………………………………………………………………. 48
      5. Diseño del sistema………………………………………………………………………….. 48
   3. Permisos……………………………………………………………………………………………………… 48
6. Resultados y conclusiones…………………………………………………………………………………. 50
7. Posibles líneas de trabajo futuro……………………………………………………………………….. 53
8. Referencias……………………………………………………………………………………………………….. 54

**Tabla de ilustraciones**

Ilustración 1. Arquitectura del sistema Android……………………………………………………………………… 12

Ilustración 2. Compilación código Java en Android………………………………………………………………… 14

Ilustración 3. Inicio del sistema Android………………………………………………………………………………… 15

Ilustración 4. Ciclo de vida de una actividad…………………………………………………………………………. 17

Ilustración 5. Ciclo de vida de un servicio……………………………………………………………………………… 18

Ilustración 6. Content Provider………………………………………………………………………………….…………. 19

Ilustración 7. Broadcast Receiver……………………………………………………………………………….………… 19

Ilustración 8. Recursos………………………………………………………………………………………….….………….. 20

Ilustración 9. IDE de desarrollo Android Studio…………………………………………………………...…….... 21

Ilustración 10. Acelerómetro……………………………………………………………………………………………….. 23

Ilustración 11. Acelerómetro en un móvil………………………………………………………………………….…. 24

Ilustración 12. Satélites GPS………………………………………………………………………………………………... 25

Ilustración 13. Acelerómetro de un móvil……………………………………………………………………………. 29

Ilustración 14. Tipos de datos PieceOfData…………………………………………………………………………. 30

Ilustración 15. Tipos de clases…………………………………………………………………………………………….. 31

Ilustración 16. Cadena de clases para calcular la frecuencia……………………………………………….. 32

Ilustración 17. Gráfica con los valores del acelerómetro……………………………………………………... 34

Ilustración 18. Interfaz principal de entrenamiento…………………………………………………………….. 35

Ilustración 19. Interfaz carpetas de entrenamientos…………………………………………………………… 36

Ilustración 20. Interfaz entrenamientos………………………………………………….…………………………… 37

Ilustración 21. Interfaz gráfica datos del entrenamiento………………………….…………………………. 38

Ilustración 22. XML con los datos de la sesión………………………………….…………………………………. 38

Ilustración 23. Interfaz con los datos de la sesión……………………………….………………………………. 39

Ilustración 24. Canoa……………………………………………………………………….………………………………… 40

Ilustración 25. Kayak…………………………………………………………………….…….……………………………… 40

Ilustración 26. K4 femenino de aguas tranquilas…………………….…………….…………………………… 41

Ilustración 27. Eslalon…………………………………………………………………………..…………………………… 41

Ilustración 28. Descenso de aguas bravas………………………………………….….…………………………. 42

Ilustración 29. Ascensos, descensos y travesías………………………………………..………………………. 42

Ilustración 30. Porteo de un k2 en un maratón…………………………………………………………………. 43

Ilustración 31. Kayak polo………………………………………………………………………………………………… 43

Ilustración 32. Kayak mar………………………………………………………………………………..…..…………… 44

Ilustración 33. Estilo libre…………………………………………………………………………………………… 44

Ilustración 34. Barcos dragón……………………………………………………………………………………. 45

Ilustración 35. Piragüismo adaptado………………………………………………………………………… 45

Ilustración 36. Permisos de la aplicación…………………………………………………………………… 49

Ilustración 37. Soporte del móvil 1……………………………………………………………………………. 50

Ilustración 38. Soporte del móvil 2……………………………………………………………………………. 51

Ilustración 39. Tutor y alumno……………………………………………………………………….…………. 52

Ilustración 40. Logo del Club Salamanca Canoe Kayak……………………………….…………….. 52

1. **Introducción**

El presente documento recoge la memoria del proyecto de Fin de Carrera titulado **“Aplicación Android para mejorar el rendimiento deportivo en piragüismo”** en la titulación de ingeniería informática de la Universidad de Salamanca. Ha sido desarrollado por el alumno Pablo Rubio Machacón durante los meses comprendidos entre octubre de 2017 y junio de 2018.

El proyecto ha sido desarrollado con la ayuda de mi tutor Guillermo Gonzalez Talaván, profesor de la Universidad de Salamanca y mi cotutor Alberto Rodríguez Valle, director deportivo del Club Salamanca Canoe Kayak (a partir de ahora, CSCK).

Además, cabe destacar, que este trabajo ha sido dotado con una beca conjunta entre la Universidad de Salamanca y el banco Santander llamada YUZZ (anteriormente conocida como beca EXPLORER). Durante los meses de desarrollo, se ha podido adquirir una formación específica sobre emprendimiento, creación de empresas, conocimiento de uno mismo y propiedad industrial.

La idea de este proyecto surgió a partir de mis años de experiencia y entrenamiento en el club de piragua de la ciudad de Salamanca, el más importante de la ciudad, el CSCK.

Tras muchos kilómetros encima de una piragua me di cuenta de que no existía una aplicación propia y específica para ayudar a llevar un control sobre los entrenamientos de este deporte. Gracias a los conocimientos adquiridos a lo largo de estos años en la Universidad de Salamanca, decidí programar una aplicación para ayudar a mejorar el rendimiento deportivo de los palistas que quieran llevar un control sobre sus entrenamientos.

Después de decidir el tema de mi TFG, pensé en qué plataforma o dispositivos podría llevar a cabo su implementación, y llegué a la conclusión de que la forma de extender mi aplicación al mayor número de personas posible sería implementarla en un dispositivo Android, ya que es el sistema operativo móvil más extendido en todo el mundo en la actualidad y los dispositivos sobre los que se instala y constan de los sensores necesarios para realizar las mediciones de los datos pertinentes y capacidad de cálculo suficiente para su procesado.

Los entrenamientos en piragüismo están formados por series y bloques, los cuales pueden ser de tiempo o de distancia. Estas series tiene un ritmo de paladas asociado, los ritmos se diferencian unos de otros según la frecuencia de paleo que se lleve (paladas por minuto). Pueden ser Ritmo 1 (R1 – 60/72), Ritmo 2 (R2 – 72/85), Ritmo 3 (R3 – 85/100), R4, R5 y R6 son al máximo, lo único que cambia es el tiempo de descanso entre series y por último R7 es el ritmo de competición, que dependerá de la distancia de la prueba, ya que no es lo mismo una competición de 500 metros que un maratón.

El objetivo principal de este proyecto es permitir a todo piragüista que lo desee llevar un control sobre sus entrenamientos encima de una piragua para, de esta forma, mejorar su rendimiento deportivo, la aplicación programada podrá poder medir los datos en directo del entrenamiento, como pueden ser la velocidad, la distancia o la frecuencia de paladas que lleva el piragüista en ese momento, así como el posterior estudio de estos datos una vez terminada la sesión.

La aplicación ofrecerá por medio de una interfaz fácil, sencilla e intuitiva la forma correcta para poder cumplir todos estos objetivos, gracias al uso de diferentes sensores, en especial, el GPS y el acelerómetro.

Los datos de estos sensores serán tratados de forma modular y flexible ante la posibilidad de usar nuevos sensores, tratarlos de una forma diferente con nuevas fórmulas y procesarlos de forma offline.

También, se tendrá la capacidad de que el entrenador planifique sesiones de entrenamiento, así como la recopilación final de los datos de sus palistas una vez finalizada la sesión.

* El documento se estructura de la siguiente forma:

1. Introducción.
2. Objetivos.
3. Conceptos teóricos y herramientas empleadas.
4. Obtención de medidas y objetivos.
5. Aspectos relevantes del desarrollo.
6. Resultados y conclusiones.
7. Posibles líneas de trabajo futuro.
8. Referencias.

* Aparte de estos apartados, la memoria del TFG se completará con los siguientes anexos:
* Anexo I: Planificación temporal.
* Anexo II: Especificación de requisitos.
* Anexo III: Análisis y diseño del sistema.
* Anexo IV. Manual del programador.
* Anexo V. Manual de usuario.

1. **Objetivos**

A continuación, se exponen los diferentes objetivos iniciales del proyecto:

El objetivo principal de este proyecto es crear una aplicación para dispositivos Android capaz de poder realizar mediciones de los datos en directo del entrenamiento, como son la velocidad, la distancia recorrida y la frecuencia de paladas, gracias a la recopilación de los datos y su posterior procesado de los diferentes sensores que tienen estos dispositivos móviles, como son el GPS, el acelerómetro o el giroscopio.

También, entre los objetivos está el posterior estudio de estos datos para calcular ritmos medios, aceleraciones y la realización de diferentes gráficos comparativos de los datos una vez finalizado el entrenamiento.

Durante el entrenamiento, la aplicación tendrá una realimentación sonora y de vibración para alertar a los palistas del inicio y final de series y bloques del entrenamiento.

También la aplicación tendrá la capacidad para guardar un historial de los entrenamientos realizados para su posterior estudio, o su comparación entre los diferentes palistas que tenga a su cargo un entrenador, gracias a que este podrá de forma inalámbrica recopilar los datos de los entrenamientos de estos. No solo se tendrá un historial de los entrenamientos realizados, sino que se podrán crear entrenamientos propios, para después llevarlos a la práctica y guardar sus resultados.

Opcionalmente, fijamos el objetivo de colocar un dispositivo que tuviese sensores como son el acelerómetro y el giroscopio en la pala (remo) para poder capturar los datos de movimiento y giro de esta y pasarlos a un ordenador para simular en 3D el paleo de la misma con la ayuda de la biblioteca OpenGL.

Gracias a la simulación del movimiento de la pala, se puede llegar a visualizar la forma de paleo o técnica del deportista en cuestión, y a su vez guardar los movimientos de esta para después visualizarlos, corregir los fallos e incluso comparar la forma de paleo del deportista con el de otro palista el cual tenga buena técnica de remar y se quiera aparentar su paleo.

1. **Conceptos teóricos y tecnologías empleadas**

En este apartado de la memoria trataré de explicar los conceptos teóricos más importantes necesarios para entender el desarrollo y funcionamiento del proyecto. Hay que destacar que las herramientas y tecnologías empleadas en este proyecto no son de elaboración propia, sino que se han utilizado o mejorado algunas de estas ya existentes para obtener las funcionalidades citadas en el apartado de objetivos.

Las tres principales tecnologías empleadas son Android, un sistema operativo de móviles que proporciona todo lo necesario para el desarrollo de aplicaciones para estos dispositivos, Android Studio, que es el entorno de desarrollo integrado para programar aplicaciones Android y OpenGL, una librería que permite trabajar con gráficos 3D, en nuestro caso, para simular por ordenador el movimiento de la pala.

* 1. **Android**

Android es un sistema operativo para dispositivos móviles cuya misión consiste en abstraer el hardware subyacente. Inicialmente fue desarrollado para dispositivos táctiles con recursos limitados a fin de facilitar el desarrollo de aplicaciones para dichos dispositivos.

La gran diferencia de Android respecto al resto de sistemas operativos para móviles es su núcleo basado en GNU/Linux. Esto hace que Android adquiera algunas de las principales características de Linux convirtiéndose en un software libre, gratuito y multiplataforma.

Entre sus peculiaridades podemos destacar:

* Android es una plataforma de desarrollo libre y código abierto.
* Gran cantidad de servicios disponibles.
* Aplicaciones hechas de componentes.
* Multitud de información.
* Multimedia.
* Seguridad.
* Gestión del ciclo de vida automático.
* Múltiple Hardware.
* Se programa en Java, C y C++.

**3.1.1 Historia**

Android es un sistema operativo diseñado para funcionar principalmente en dispositivos táctiles desarrollado inicialmente por Android IC, en el 2005 fue comprado por Google, cuando aún era un sistema operativo muy poco conocido.

En el 2007 se fundó Opend Handset Alliance, una agrupación de empresas, una agrupación de empresas de desarrollo software y hardware con el propósito de avanzar en los estándares para el desarrollo de software y hardware para dispositivos móviles.

Acto seguido se produjo la presentación de Android por parte de Google y se liberó gran parte de su código bajo licencia Apache.

Hubo que esperar varios años a que se extendiese el uso de estos dispositivos móviles que usaban este sistema operativo para que se convirtiese en el más usado de todos.

**3.1.2 Arquitectura**

La arquitectura del sistema Android puede verse como una arquitectura de capas o niveles, de forma que cada nivel puede utilizar servicios ofrecidos por los niveles superiores.

En la siguiente imagen puede verse esta arquitectura de capas:

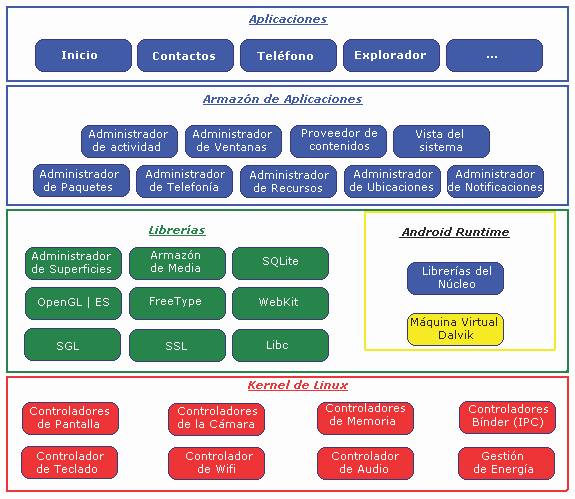


Ilustración 1. Arquitectura del sistema Android

* Aplicaciones: Constituyen el conjunto de aplicaciones presentes en un dispositivo móvil, las instaladas por el usuario, predeterminadas o administradas.
* Framework o armazón de aplicaciones: Plataforma de desarrollo que facilita la reutilización de componentes, permite el acceso a los diversos servicios ofrecidos y al hardware del dispositivo. Los más importantes son:
  + Administrador de actividad: Conjunto de APIs encargadas de gestionar el ciclo de vida de las aplicaciones.
  + Administrador de ventanas: Gestiona las ventanas de las aplicaciones por medio de la librería ‘Administrador de superficies’.
  + Administrador de telefonía: Conjunto de APIs que gestionan las funciones ‘básicas’ de los teléfonos móviles como son las llamadas, los mensajes, etc.
  + Proveedor de contenidos: Proporciona los mecanismos necesarios para la comunicación entre aplicaciones.
  + Vista del sistema: Ofrece los elementos básicos necesarios para la construcción de interfaces.
  + Administrador de ubicaciones: Posibilita a las aplicaciones el acceso a la ubicación del dispositivo.
  + Administrador de notificaciones: Permite a las aplicaciones notificar al usuario información asociada a ciertos eventos que ocurran durante la ejecución de una aplicación.
  + Servicio XMPP: Conjunto de APIs para el uso de este protocolo de intercambio de mensajes basado en XML.
  + Administrador de recursos: Encargado de gestionar todos los elementos que forman parte de una aplicación externos al código.
  + Administrador de paquetes: Gestor de todos los paquetes instalados en un dispositivo Android, permite la instalación de nuevos paquetes.
* Bibliotecas: Hacen referencia al conjunto de librerías presentes en Android, están escritas en C/C++ y proporcionan la mayoría de las características más representativas de esta plataforma. Las principales librerías presentes en cualquier dispositivo Android son:
  + Administrador de superficies: Gestión de la pantalla.
  + Media Framework o armazón de media: Reproducción de imágenes, video y audio.
  + SQLite: Motor de bases de datos relacionadas.
  + WebKit: Navegación web.
  + SGL: Gráficos 2D.
  + Open GL/ES: Gráficos 3D.
  + FreeType: renderizado de fuentes.
  + SSL: Comunicación segura mediante sockets.
  + Libc: Variante optimizada de C.
* Android Runtime: El entorno de ejecución, que se encuentra al mismo nivel que las librerías, está constituido por las librerías Java que forman el núcleo del lenguaje y la máquina virtual Dalvik (o ART en versiones más modernas). Es necesario remarcar que Java se usa únicamente como lenguaje de programación, el código obtenido como resultado de compilar un programa Android no es compatible con el bytecode de Java.

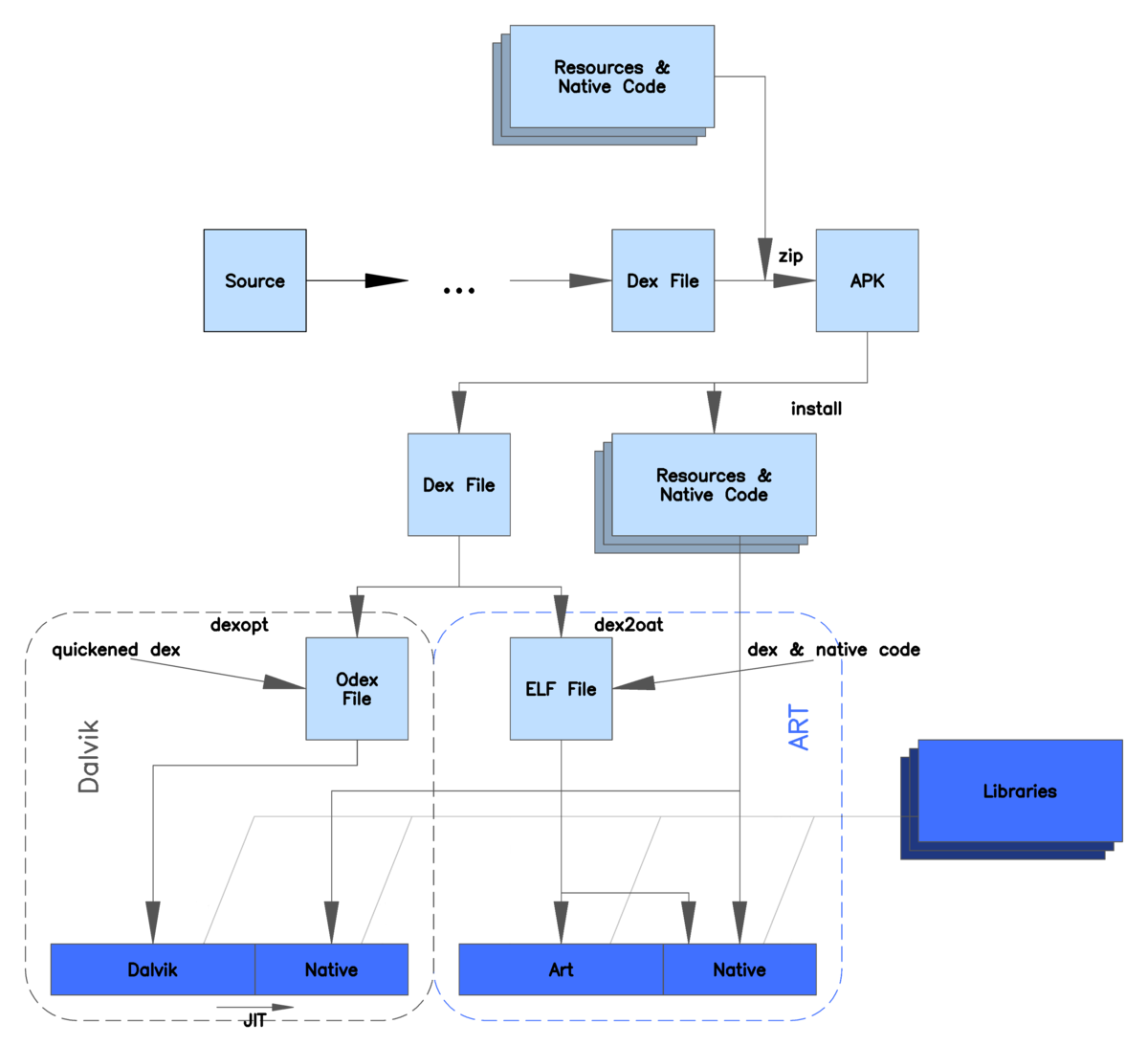


Ilustración 2. Compilación código Java en Android

* Núcleo Linux: Se utiliza como una capa de abstracción para el Hardware subyacente, contiene los drivers o controladores necesarios para el correcto funcionamiento del dispositivo.

Por último, para cerrar el apartado sobre la arquitectura Android, vamos a ver como se produce la inicialización de esta arquitectura cuando encendemos el dispositivo:

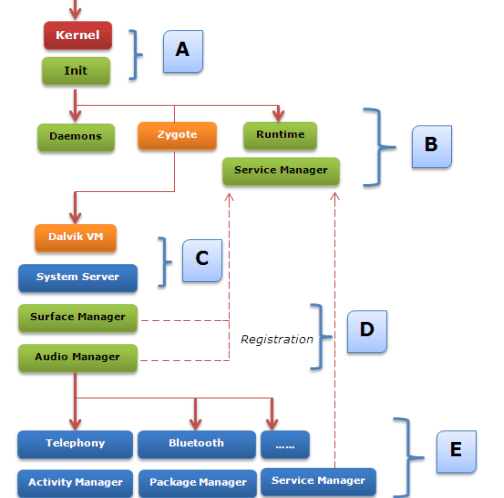


Ilustración 3. Inicio del sistema Android

* Los pasos que se realizan son los siguientes:

1. Inicialmente, el bootloader carga el kernel y lanza el proceso init.
2. Se crean los demonios o daemons encargados de gestionar el hardware subyacente, inmediatamente después se crea el Zygote, que será la primera instancia de la máquina virtual y el Runtime inicia el gestor de servicios.
3. El proceso Runtime pide a Zygote que lance una nueva instancia de la máquina virtual para ejecutar el servidor del sistema.
4. Los dos primeros procesos ejecutados son los necesarios para gestionar el audio y la pantalla.
5. Por último, se lanzan el resto de procesos.

**3.1.3 Aplicaciones**

Este subapartado lo dedicaremos al estudio de las aplicaciones Android. Las aplicaciones Android están escritas en el lenguaje de programación Java y se implementan mediante diferentes herramientas de desarrollo software (SDK a partir de ahora). Son las encargadas de recoger todo el código de la aplicación junto con los datos y recursos de esta en un mismo paquete con extensión APK.

Este paquete APK contiene toda la información necesaria para instalar una aplicación en un dispositivo. Una vez instalada esta, su vida transcurre dentro de su propio entorno de seguridad o sandbox.

El sistema operativo Android puede definirse como un sistema operativo multiusuario en el cual cada aplicación es un usuario diferente. Por defecto, el sistema operativo identifica a cada aplicación con un identificador único, desconocido por la propia aplicación. Según el identificador que tenga tendrá unos permisos u otros, de esta forma se garantiza que la aplicación no tenga acceso a todos los ficheros del sistema, sino solo a los que tenga permiso.

Cada aplicación Android tiene una máquina virtual propia, de forma que el código de cada aplicación se ejecuta de forma aislada del resto de estas.

De forma predeterminada, cada aplicación se ejecuta en su propio proceso Linux. Android inicializa una aplicación cuando se requiere cualquiera de sus componentes o servicios y termina el proceso cuando ya no es necesario o se requiere memoria para otras aplicaciones.

De lo dicho anteriormente, podemos deducir que Android crea un entorno de trabajo seguro, donde una aplicación solo puede acceder a la información que necesita para hacer su trabajo, este hecho se conoce como principio del mínimo privilegio.

Sin embargo, existen diversas formas mediante las cuales una aplicación puede compartir información con otra aplicación y acceder a los diferentes servicios que nos brinda el sistema. La existencia de dos aplicaciones con el mismo identificador Linux es posible, de esta forma una aplicación podrá acceder a los archivos de la otra y viceversa. Con objeto de ahorrar recursos, las aplicaciones también podrán compartir máquina virtual. Las aplicaciones también pueden acceder a recursos del dispositivo solicitando los pertinentes permisos al usuario.

**3.1.3.1 Componentes**

Los componentes de toda aplicación Android son los elementos esenciales que la componen. Cada componente supone un punto de entrada mediante el cual el sistema puede acceder a la aplicación, aunque no todos suponen un punto de entrada para el usuario.

Cada componente es un bloque básico que desempeña un papel específico en el funcionamiento de nuestra aplicación. Existen cuatro tipos de componentes diferentes, cada uno con un ciclo de vida y propósitos específicos. Son los siguientes:

* Actividades: Cada actividad representa una pantalla de nuestra aplicación. Las actividades trabajan en conjunto para dar una visión coherente de la aplicación, sin embargo, la vida de cada actividad es independiente al resto.

El ciclo de vida de una actividad es el siguiente:

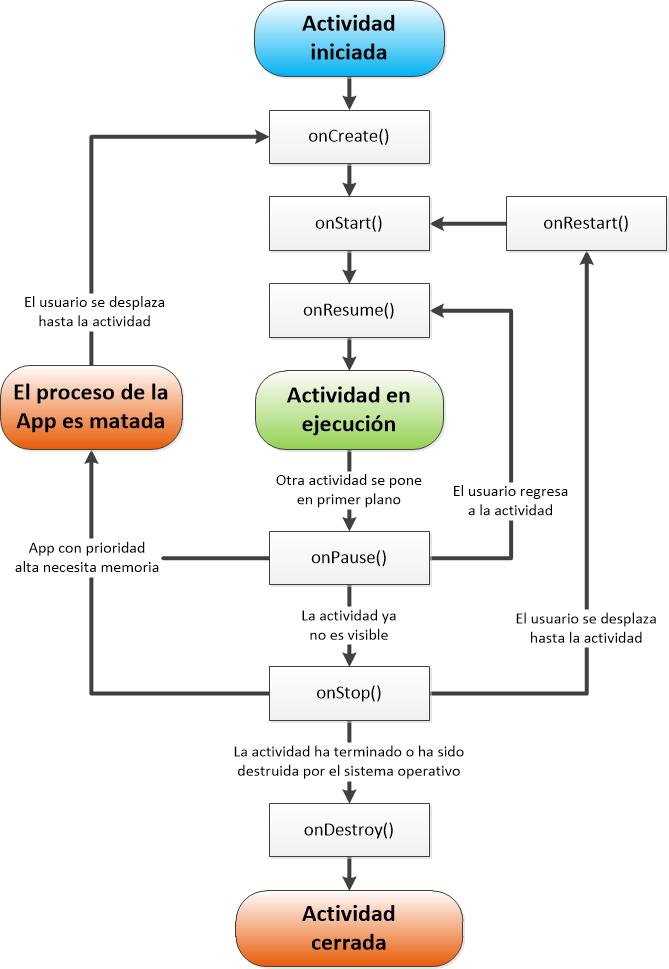


Ilustración 4. Ciclo de vida de una actividad

* Servicios: Son componentes que se ejecutan en segundo plano a fin de realizar operaciones de larga duración o procesos remotos. Los servicios carecen de interfaz y generalmente son llamados por las actividades para realizar diferentes tareas costosas sin bloquear la interfaz, se pueden dejar en ejecución en segundo plano o ligarlos a una actividad para interactuar con ella.

En la siguiente imagen vemos el ciclo de vida de un servicio Android:

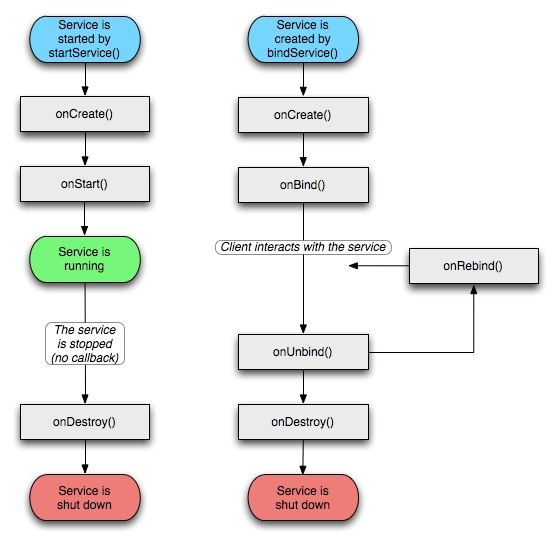


Ilustración 5. Ciclo de vida de un servicio

* Proveedor de contenido o ‘content provider’: El proveedor de contenido gestiona todos los datos que maneja una aplicación. Controla el acceso a archivos, bases de dato, etc. Mediante este proveedor las aplicaciones pueden consultar y modificar datos de otras aplicaciones siempre que posean los permisos adecuados.

En la siguiente imagen podemos ver como una actividad puede consultar datos de otra aplicación:

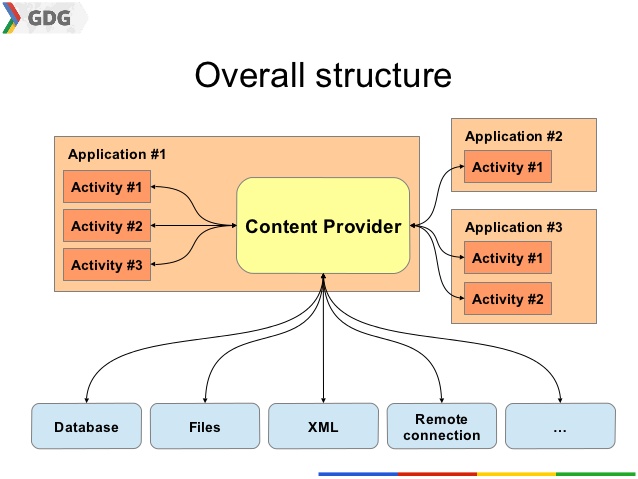


Ilustración 6. Content Provider

* Receptor de difusiones o ‘broadcast receiver’: Es el componente encargado de responder a las difusiones o anuncios del sistema. Carece de interfaz, pero debe crear una barra de estado cada vez que recibe una difusión para notificar al usuario de esta. Cada aplicación debe registrar un broadcast receiver para indicar que difusiones le interesan.

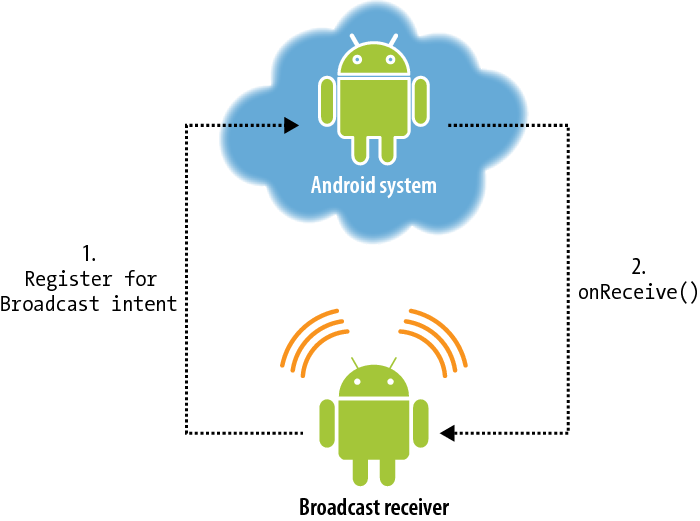


Ilustración 7. Broadcast Receiver

**3.1.3.2 Recursos**

Los recursos de una aplicación corresponden con todos los ficheros, imágenes, cadenas de texto, bases de datos… que nuestra aplicación utiliza para su correcto funcionamiento. Cuando trabajamos desarrollando aplicaciones para Android, las buenas prácticas nos dicen que este tipo de archivos debe mantenerse independiente al código de la aplicación.

Esto es conocido como externalización de recursos, permite adaptar un mismo código a dispositivos con diferentes configuraciones y características físicas como puede ser el tamaño de la pantalla de forma automática.

En la siguiente imagen podemos ver como quedaría una pantalla que no ha sido adaptada para diferentes dispositivos (arriba) y otra que si ha sido adaptada (abajo):

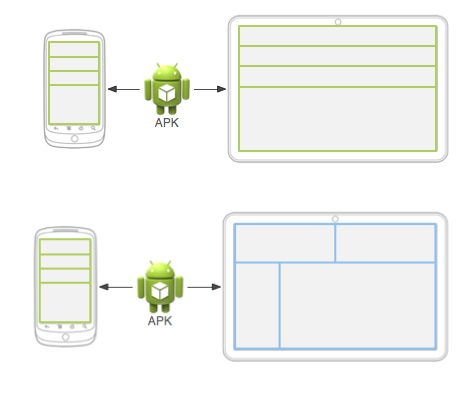


Ilustración 8. Recursos

**3.1.3.3 Manifiesto**

El manifiesto o ‘manifest’ en Android es un archivo XML que contiene información esencial acerca de la aplicación. Todas las aplicaciones deben tener un archivo AndroidManifest.xml en su directorio raíz.

Las funciones más significativas del manifiesto son las siguientes:

* Contiene la declaración de los permisos.
* Lista las librerías necesarias.
* Da nombre al paquete Java de la aplicación.
* Describe los componentes y qué proceso los hospeda.
* Declara el nivel mínimo de Android API requerido para ejecutar la aplicación.

**3.1.4 Android Studio**

Android Studio es el entorno de desarrollo integrado para el desarrollo de aplicaciones Android basado en ItelliJ-IDEA.

Entre las muchas ventajas que nos ofrece podemos contar con una vista ordenada y modular de los archivos que componen nuestro proyecto. La vista mostrada no se corresponde con la forma original de los archivos en disco, esta pesada para optimizar el trabajo del desarrollador.

Integrado en nuestro entorno de trabajo, podemos encontrar un sistema para el control de versiones que incluye soporte para los repositorios más conocidos (Git, Mercurial…).

La interfaz gráfica está dividida de forma inteligente para mostrar una gran cantidad de información de forma eficiente, es totalmente personalizable permitiendo al usuario mostrar y ocultar paneles de información según las preferencias de este. Android Studio sigue el contexto de trabajo del usuario mostrando las ventanas con las herramientas que considera oportunas en cada instante. A continuación, se muestra un ejemplo de interfaz gráfica de este entorno de desarrollo:

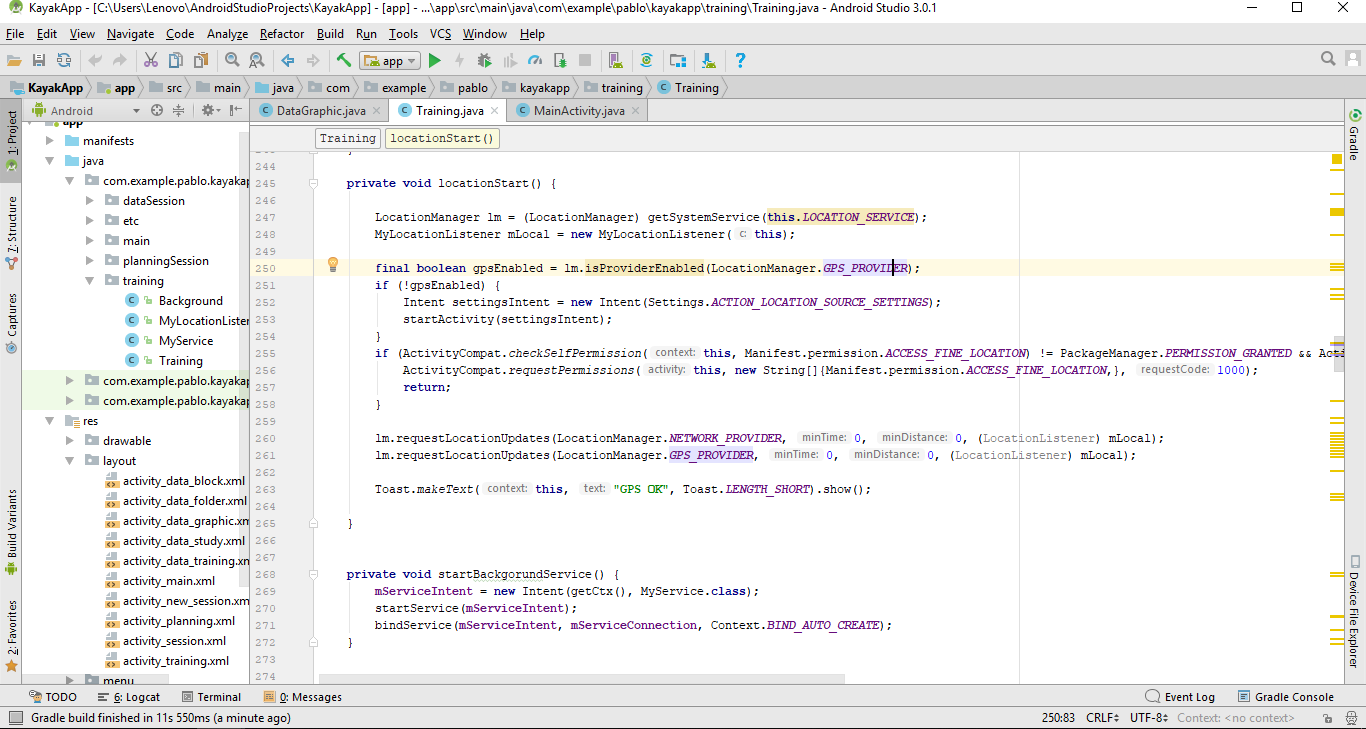


Ilustración 9. IDE de desarrollo Android Studio

Para terminar el apartado de Android Studio hablaremos del Gradle. Gradle es un sistema abierto que automatiza la compilación del código, utiliza un grafo dirigido acíclico para determinar en que orden se pueden ejecutar las tareas. Android Studio incorpora esta tecnología para facilitar la reutilización del código permitiendo generar APKs con distintas funcionalidades partiendo de un mismo proyecto.

**3.1.5 SDK – Software Development Kit**

Android SDK o Android Software Development Kit es un conjunto de herramientas para desarrollar, compilar y depurar aplicaciones para el sistema operativo Android, en definitiva, en una API que además de herramientas para el desarrollo, proporciona soporte técnico, ejemplos y una buena y desarrollada documentación.

El SDK incluye un emulador de dispositivos Android, el cual permite probar las aplicaciones de forma rápida y eficiente. Los parámetros de nuestro emulador pueden configurarse de forma que podemos elegir las características que tendrá nuestro dispositivo, como la memoria principal, el tamaño del dispositivo o la versión de Android, lo que permite al programador probar su código en una gran cantidad de dispositivos con diferentes características sin necesidad de adquirirlos.

**3.1.6 NDK – Native Development Kit**

Android NDK o Native Development Kit provee las herramientas necesarias para desarrollar código Android en C/C++, frente al SDK que lo desarrolla en lenguaje Java.

Inicialmente el NDK no está recomendado para programadores y desarrolladores sin experiencia en Android ya que incrementa notablemente la complejidad del código y no es muy útil en general para la mayoría de las aplicaciones.

NDK puede utilizarse para crear nuevo código o reutilizar el existente, y pese a lo anterior dicho, existen dos situaciones que justifican su uso: exprimir al máximo la capacidad de cómputo de los dispositivos o utilizar bibliotecas de otros desarrolladores escritas en C/C++, esto se usará en este proyecto mediante el uso de la librería OpenCV.

* 1. **OpenGL**

OpenGL es una especificación estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D. La interfaz consiste en más de 250 funciones diferentes que pueden usarse para dibujar escenas tridimensionales complejas a partir de primitivas geométricas simples, tales como puntos, líneas y triángulos

* 1. **Tecnologías empleadas para la obtención de medidas**

Entre los anteriores objetivos de la aplicación citados anteriormente, se encuentran la obtención de medidas directas del entrenamiento, como la velocidad, distancia recorrida o frecuencia de paleo del piragüista.

Para poder conseguir estos valores será necesaria la obtención de datos y medidas de los diferentes sensores que tiene la gran mayoría de dispositivos Android.

Estos dispositivos son los siguientes:

* + 1. **Acelerómetro**

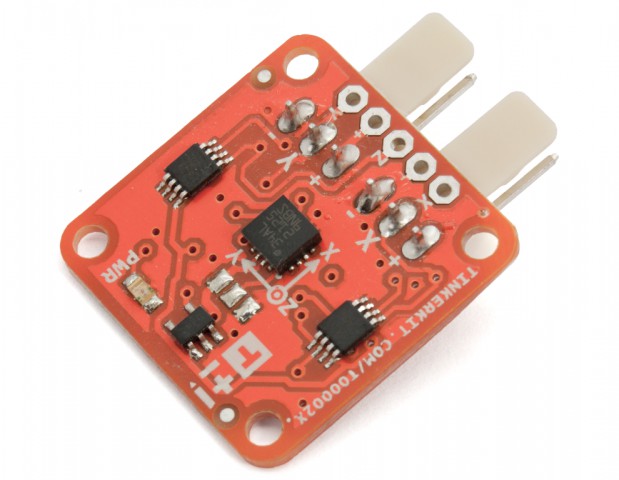
****

Ilustración 10. Acelerómetro

Un acelerómetro es, como su propio nombre indica, un sensor que mide la aceleración producida por el movimiento (variación de la velocidad en función del tiempo). Su uso es muy común en la electrónica actual para detectar el movimiento de dispositivos tales como teléfonos móviles, tablets o pads de videoconsolas, ya que, al conocer el sentido en el que se produce dicho movimiento, podemos saber la posición del sensor dentro de un espacio tridimensional.

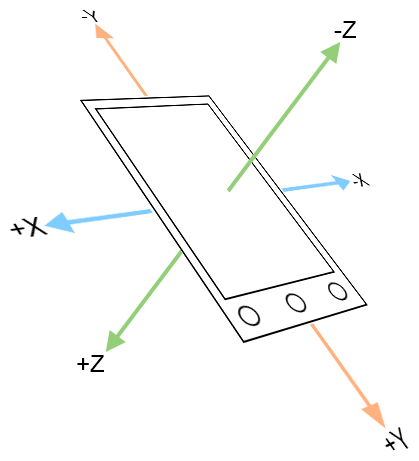


Ilustración 11. Acelerómetro en un móvil

Este sensor informa al sistema operativo del desplazamiento producido proporcionándole los datos generados por el movimiento, a continuación, el teléfono compara esos datos con los almacenados en una tabla. Dicha tabla tiene un registro en donde están relacionadas las acciones a ejecutar en función de unos valores concretos. Si los valores generados por el acelerómetro coinciden con alguno de los registros de la tabla, el teléfono ya sabe la orientación del sensor.

Según la información que proporcionan, existen varios tipos de acelerómetros, de un eje, de dos o de tres. Los sensores de tres ejes (el tipo que tienen los teléfonos móviles) proporcionan información sobre las coordenadas X, Y y Z dentro del espacio. En este proyecto trabajaremos con el acelerómetro de tres ejes, en especial, sobre el eje Z de este, que es mediante el cual conseguiremos calcular la frecuencia de paladas del piragüista.

* + 1. **GPS**

Los dispositivos Android han adquirido unas funciones impensables hace unos años, y entre ellas está el GPS: un sistema capaz de localizarnos en cualquier parte del mundo, y que resulta muy útil en combinación a aplicaciones de mapas o de indicaciones.

La red GPS emplea una red de 24 satélites (32 en total contando con los satélites adicionales que mejoran la precisión) en órbita que cubren toda la superficie de nuestro planeta, de tal forma que, desde cualquier punto en el que podamos estar, podamos tener línea de visión directa con un mínimo de 6 satélites.

A partir de esa base, cuando nuestro dispositivo móvil quiere localizarnos, se conectará a esta red con la mayor cantidad de satélites posibles, obteniendo una serie de datos, y utilizando la triangulación inversa, podrá situarnos en el mapa con un margen de error relativamente pequeño.

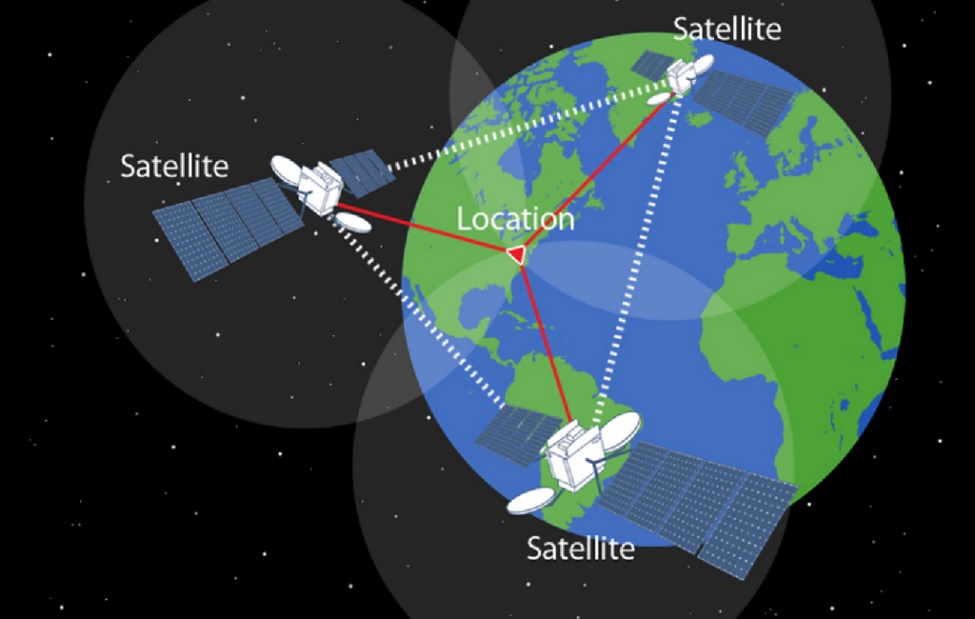


Ilustración 12. Satélites GPS

En nuestro proyecto usaremos los servicios del GPS para poder calcular la distancia que llevamos recorrida en piragua y la velocidad a la que nos movemos.

El sistema GPS tiene un gran inconveniente no funciona en interiores, o en lugares donde la línea de visión no es directa, ya que la señal es muy débil o nula y no se logran localizar satélites suficientes para conocer una ubicación precisa.

* + 1. **Wifi-Direct**

Wifi Direct es forma de conectividad Wifi, que permite conectar 2 dispositivos certificados como Wifi Direct (como tabletas, teléfonos, impresoras...) directamente.  
   
A diferencia de una conexión Wifi normal, Wifi Direct no necesita un punto de acceso para gestionar la conexión (como un router o hotspot) es decir, que sin ese punto de acceso, los dispositivos no se pueden "ver";  con lo que con Wifi Direct se podrá mandar archivos directamente y a una velocidad y alcance muy superior al Bluetooth (Unas 10 veces más rápido y casi 4 veces más alcance).

* + 1. **Lenguajes**

En este subapartado veremos los distintos lenguajes de programación que he usado en el proyecto, en particular son dos:

* + - 1. **Java**

Java es un lenguaje de programación con el que podemos realizar cualquier tipo de programa. En la actualidad es un lenguaje muy extendido y cada vez cobra más importancia tanto en el ámbito de Internet como en la informática en general. Está desarrollado por la compañía Sun Microsystems con gran dedicación y siempre enfocado a cubrir las necesidades tecnológicas más punteras.

Una de las principales características por las que Java se ha hecho muy famoso es que es un lenguaje independiente de la plataforma. Eso quiere decir que si hacemos un programa en Java podrá funcionar en cualquier ordenador del mercado. Es una ventaja significativa para los desarrolladores de software, pues antes tenían que hacer un programa para cada sistema operativo, por ejemplo, Windows, Linux, Apple, etc. Esto lo consigue porque se ha creado una Máquina de Java para cada sistema que hace de puente entre el sistema operativo y el programa de Java y posibilita que este último se entienda perfectamente.

La independencia de plataforma es una de las razones por las que Java es interesante para Internet, ya que muchas personas deben tener acceso con ordenadores distintos. Pero no se queda ahí, Java está desarrollándose incluso para distintos tipos de dispositivos además del ordenador como móviles, agendas y en general para cualquier cosa que se le ocurra a la industria.

Este proyecto está escrito en su totalidad en Java, salvo los archivos de las interfaces gráficas que están en XML.

**3.3.3.2 XML**

XML proviene de eXtensible Markup Language (“Lenguaje de Marcas Extensible”). Se trata de un metalenguaje (un [lenguaje](https://definicion.de/lenguaje/) que se utiliza para decir algo acerca de otro) extensible de etiquetas que fue desarrollado por el Word Wide Web Consortium (W3C), una sociedad mercantil internacional que elabora recomendaciones para la World Wide Web.

El XML es una adaptación del SGML (Standard Generalized Markup Language), un lenguaje que permite la organización y el etiquetado de [documentos](https://definicion.de/documento/). Esto quiere decir que el XML no es un lenguaje en sí mismo, sino un sistema que permite definir lenguajes de acuerdo a las necesidades

En este proyecto se usará XML para guardar los datos de los entrenamientos (descansos, tiempo de series, bloques, ritmos…) que creemos en nuestra aplicación, mediante el uso de Java.

* 1. **Herramientas CASE**

CASE (Computer Aided Software Engineering) es un conjunto de aplicaciones informáticas que tiene como objetivo semi automatizar las tareas de ingeniería del software de una aplicación, con el propósito de ayudar en el proceso de desarrollo e implementación y conseguir un sistema de alta calidad, sin fallos y con gran capacidad de mantenimiento y reutilización.

* + 1. **Lenguaje unificado de modelado (UML)**

UML es el lenguaje de modelado por excelencia para el diseño de proyectos informáticos orientados a objetos.

1. **Obtención de medidas y objetivos**

Ya se ha comentado anteriormente que el objetivo principal de esta aplicación es la de realizar las mediciones de los datos de un entrenamiento en piragüismo, es especial, la frecuencia o cadencia de paladas, la velocidad y la distancia recorrida.

En este capítulo trataremos de explicar las diferentes técnicas empleadas para conseguirlo. Para cada técnica implementada, explicaremos qué sensores intervienen, de qué forma el usuario las puede utilizar, y mostraremos los cálculos necesarios para convertir los datos de entrada en los valores que queremos medir.

* 1. **Distancia y Velocidad**

La señal del GPS nos proporciona la longitud (ф), distancia angular al meridiano de Greenwich o meridiano cero, y la latitud (ϕ), distancia angular al ecuador, ambas en grados, minutos y segundos (sexagesimales). En la zona de pruebas (Salamanca) ф es negativo y ϕ positivo.

La aplicación toma lecturas de la señal cada 10 segundos.

En cada sesión se establece un sistema de coordenadas cartesianas con el sentido positivo del eje X según el polo norte geográfico y el sentido positivo del eje Y según el este geográfico. En el momento de comenzar la sesión, instante inicial, se establece X0=0 e Y0=0.

Se trata de determinar las coordenadas X e Y (expresadas en metros) de la posición del móvil (piragua) con cada lectura de la señal GPS. De este modo se obtienen una sucesión de parejas de coordenadas (X0,Y0), (X1,Y1), (X2,Y2),…equidistantes temporalmente, cuya representación gráfica es una poligonal.

La distancia recorrida en el intervalo comprendido entre las lecturas j y j+1, es decir, un tramo recto de la poligonal recorrida durante 10 segundos, es:

dj,j+1 (m) = [(Xj – Xj+1)2 + (Yj – Yj+1)2]1/2

La velocidad media en dicho intervalo es:

vj,j+1 (m/s) = dj+1,j/ 10

El cálculo de las coordenadas X1 e Y1 en las que se encuentra el móvil transcurrido el primer intervalo de 10 s, así como la distancia recorrida y la velocidad media, se realiza como sigue:

* Como ya se ha indicado se fijan los valores X0=0 e Y0=0.
* Para el radio de la tierra en la zona de pruebas se fija el valor R = 6371000 m.
* Se transforman a radianes los valores de ф0 y ϕ0 en el instante inicial.
* Se transforman a radianes los valores de ф1 y ϕ1 transcurridos 10 s.
* Calculamos el valor del radio del paralelo en la zona de pruebas R01, tomando para la latitud la media de ϕ0 y ϕ1:

R01 = R x cos ((ϕ0 + ϕ1)/2)

* Se determinan las distancias recorridas según los ejes X e Y:

ΔX01 = (ф1 - ф0) x R01

ΔY01 = (ϕ1 - ϕ0) x R

* Ahora se calculan las coordenadas X1 e Y1

X1 = X0 + ΔX01

Y1 = Y0 + ΔY01

* Finalmente, la distancia recorrida y velocidad media en el primer intervalo son:

d01 = [(X0 – X1)2 + (Y0 – Y1)2]1/2 ; v0,1 = (d0,1/ 10000)\*3600\*1000

El procedimiento se repite para determinar las distancias y velocidades en el segundo y sucesivos intervalos.

* 1. **Frecuencia de paladas**

Para calcular la frecuencia o cadencia de paladas que lleva un piragüista en un momento determinado del entrenamiento, se hará uso del acelerómetro y de los datos que obtenemos de él.

Es importante medirla ya que los entrenamientos se crean en función de este dato, Ritmo1 (R1 – 60/72 paladas/minuto), Ritmo2 (R2 – Hasta 75/85 paladas/minuto), Ritmo3 (R3 – Hasta 85/100 paladas/minuto), etc. Así, un piragüista podrá llevar un control de su cadencia de palada sin necesidad de irla contando mentalmente.

Ya explicamos anteriormente que el acelerómetro cuando actualiza sus valores nos da tres valores, el eje X (De izquierda a derecha), el eje Y (De arriba a abajo) y el eje Z (De atrás hacia delante).

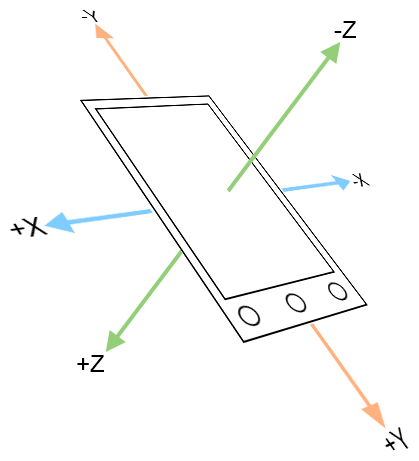


Ilustración 13. Acelerómetro de un móvil

El móvil, lo he colocado en la parte de delante de la bañera de la piragua (“hueco donde se sienta el piragüista”) ya que es el lugar mas accesible y cómodo para visualizar e interactuar con este. Por lo tanto, el eje del acelerómetro en el que nos centraremos será el Z, ya que al aplicar una palada mientras se está remando crea una fuerza o aceleración que nos impulsa hacia adelante, dejando los ejes X e Y para un posible uso futuro y así calcular el cabeceo y tambaleo de la piragua, ya que supone una pérdida de fuerza y velocidad, que todo piragüista desea evitar.

PONER FOTO DE COMO QUEDA EL MOVIL

Antes de explicar cómo conseguimos la frecuencia de paladas a partir de los datos que nos brinda el acelerómetro, voy a explicar los tipos de datos e interfaces que usan y tienen las clases que nos calcularán la frecuencia.

* Tipos de datos:

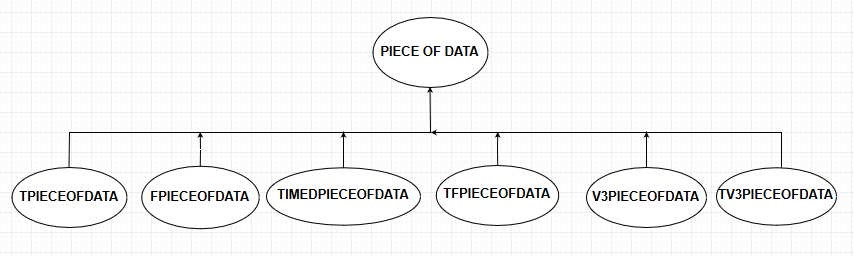


Ilustración 14. Tipos de datos PieceOfData

* PIECEOFDATA: Tipo de dato general, tiene las funciones básicas y variables comunes con las que trabajarán todos los tipos de datos, además de una etiqueta que permitirá saber de qué sensor proviene el dato.
* TPIECEOFDATA: Añade un tiempo.
* FPIECEOFDATA: Añade un float que se usará para la calcular la media de valores del eje Z del acelerómetro, así como cualquier otro float.
* TIMEDPIECEOFDATA: Añade un tiempo.
* TFPIECEOFDATA: Añade un tiempo y un float para el valor de eje Z del acelerómetro cada vez que se actualice.
* V3PIECEOFDATA: Añade un vector de tres elementos, uno para cada eje del acelerómetro [x,y,z].
* TV3PIECEOFDATA: : Añade un vector de tres elementos, uno para cada eje del acelerómetro [x,y,z] y un tiempo.
* Tipos de clases:

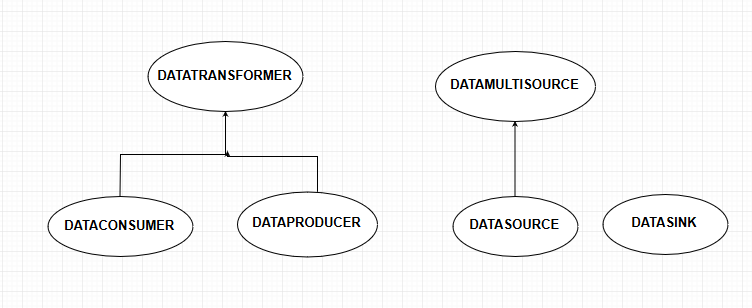


Ilustración 15. Tipos de clases

Hay cuatro tipos base, de los cuales dos de ellas son activas, es decir, actúan solas (producen o piden datos) sin que les llame ninguna clase (DataSink y DataSource) y los otras dos son pasivas, es decir, solo actúan si se las llama, es decir, se les pide o se les da datos (DataConsumer y DataProducer).

Entrando en más detalles:

* DATACONSUMER: Como su propio nombre indica, consume y trata los datos que le llegan mediante la llamada a su función onNewData(PieceOfData);
* DATAPRODUCER: Como su propio nombre indica, toma, trata y produce nuevos datos mediante la llamada a su función getData(float);
* DATASINK: Consume y trata datos que consigue de forma activa en su función go() en un hilo propio.
* DATASOURCE: Toma, trata y produce datos de una fuente (el acelerómetro en este caso) de forma activa en su función go(), y al igual que los DataSink, en un hilo propio.

A partir de estos cuatro tipos de clases, aparecen dos más que derivan de ellas, como son DATAMULTISOURCE cuando obtenemos datos de más de una fuente, ya que el móvil puede tener varios sensores de los que puede ser útil obtener información (está pensado en posibles líneas de trabajo futuro), y DATATRANSFORMER, que serán clase que consuman y produzcan a su vez datos ya que deriva de DataConsumer y DataProducer.

Una vez explicados los tipos de datos y clases que se usarán para obtener la frecuencia del palista, vamos a ver la cadena de clases por la que pasarán los datos desde que salen en bruto del acelerómetro hasta la frecuencia que nos aparecerá en la interfaz de la aplicación, así como todas las transformaciones que sufran en ellas.

* La secuencia es la siguiente:

Servicio (registrado al acelerómetro) 🡪 ExtrapolatorTransformer 🡪 SamplerSinkAndSource 🡪 ZReverseFilter 🡪 AverageReverseFilter 🡪

StrokeDetectorReverseFilter 🡪 RateMeasurerReverseFilter 🡪

StandardOutputConsumer (Que devolverá la frecuencia final calculada).

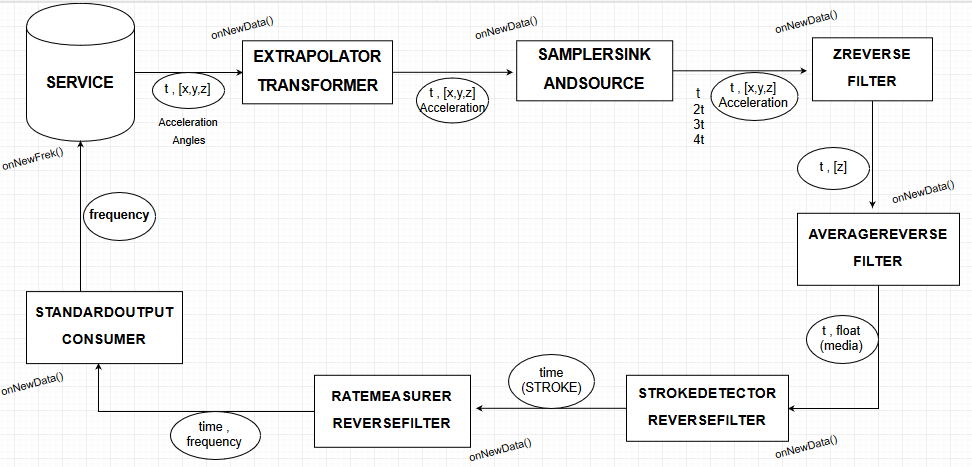


Ilustración 16. Cadena de clases para calcular la frecuencia

* SERVICE: (DataMultiSource) Esta clase es un servicio que se ejecuta en segundo plano y se encarga de registrarse a los sensores correspondientes, instanciar todas las clases necesarias para la obtención de la frecuencia, recibir la frecuencia calculada y presentarla en la actividad que gestiona la interfaz correspondiente de la aplicación.

El servicio, una vez registrado al acelerómetro recibirá una notificación cada vez que este se actualice y sus valores correspondientes, y enviará a la siguiente clase un tipo de dato TV3PieceOfData (tiempo , [x,y,z]).

* EXTRAPOLATORTRANSFORMER: (DataTransformer) Esta clase se encarga de, al recibir los datos en bruto de los diferentes sensores, etiquetarlos con su tag correspondiente, en nuestra aplicación como solo usamos el acelerómetro, solo usará esa etiqueta.

Esta clase está pensada para líneas de trabajo futuro en las cuales se usen más sensores.

Pasa a la siguiente clase el TV3PieceOfData que recibe añadiendo su etiqueta correspondiente.

* SAMPLERSINKANDSOURCE: (DataSink & DataSource) Esta clase se encarga de sacar en un intervalo de tiempo conocido (10ms) el mismo TV3PieceOfData que recibe, actualizando el tiempo de este y mandándoselo a la siguiente clase.

La razón de sacar los datos en un intervalo de tiempo conocido es que para hacer derivada y cualquier cálculo en general, se simplifica bastante. Si los valores no estuvieran tomados a intervalos regulares, habría que considerar cuanto tiempo le asignamos a cada valor y un peso.

* ZREVERSEFILTER: (DataTransformer) Como comentamos anteriormente, para conseguir la frecuencia de paladas solo íbamos a necesitar el eje Z del acelerómetro, esta clase se encarga de filtrar estos tres ejes y pasar la siguiente clase un TFPieceOfData (t , [z]).
* AVERAGEREVERSEFILTER: (DataTransformer) Esta clase se encarga de calcular una media de un valor calculado (para que el dato final sea más efectivo, en nuestro caso 33) de los valores del eje Z del acelerómetro y pasar a la siguiente clase un TFPieceOfData (t , media[ (33) Z]).
* STROKEDETECTORREVERSEFILTER: (DataTransformer) Esta clase se encargará de calcular cuando se detecta una palada y pasará a la siguiente clase el tiempo en el cuál esta ocurre.

Para ello se ayuda de dos variables, ‘derivativeThreshold’, que mide la pendiente de la curva y ‘accelerationThreshold’ que mide un umbral de la aceleración.

Para que detectemos una palada, se pone la condición mixta de que la aceleración debe superar un umbral, pero también la derivada de un umbral (una gran pendiente seguida de un valor bajo).

Los valores del eje Z del acelerómetro representados gráficamente forman una onda senoidal, como veremos a continuación.

Cada vez que el palista da una palada, el eje Z del acelerómetro sufrirá un cambio y un aumento de sus valores debido a esa aceleración que sufre la piragua, hasta que la resistencia que ofrece el agua a la piragua los reduce a sus valores normales. Hasta que se vuelve a sufrir una aceleración debido a otra palada.

Gracias a estos cambios en los valores del eje Z puede detectarse cuando se detecta una palada y en qué momento ocurre.

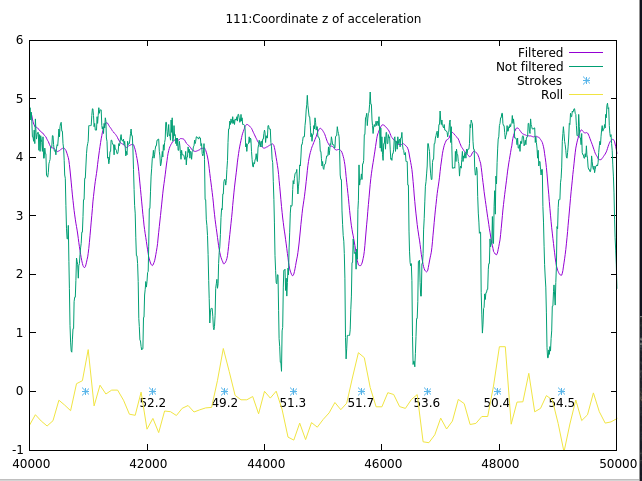


Ilustración 17. Gráfica con los valores del acelerómetro

Comentando esta imagen, se pueden observar diferentes gráficas:

* Not filtered (verde): Valores del eje Z del acelerómetro sin filtrar, como puede verse tiene bastante ruido.
* Filtered (morado): Valores anteriores filtrados (en la clase AverageReverseFilter), para reducir el ruido, eliminar las altas frecuencias y suavizar la gráfica.
* Strokes ( \* azules): Es el momento en que se detecta cada palada y su media, como comentamos anteriormente corresponde a una gran pendiente seguida de un valor bajo.

Este momento del entrenamiento correspondería con un R1 (Ritmo 1), no más de 60 paladas por minuto.

* Roll (amarillo): Valores del eje X del acelerómetro, que indican el bamboleo de la piragua. Se puede observar que existe una asimetría de fuerzas en paladas de un lado al otro, esto es así ya que la piragua se inclina hacia el lado en el cual estamos dando una palada.
* RATEMEASURERREVERSEFILTER: (DataConsumer) Clase que se encarga a partir de los tiempos de cada palada calcular una frecuencia final.
* STANDARDOUTPUTCONSUMER: (DataConsumer) Recibe la frecuencia final en cada momento y se encarga de devolver estos datos al servicio inicial subscrito al acelerómetro llamando a la función de esta onNewFrek();
  1. **Entrenamientos**

Una vez calculadas estas tres medidas ya se podrían mostrar en directo los datos de un entrenamiento.

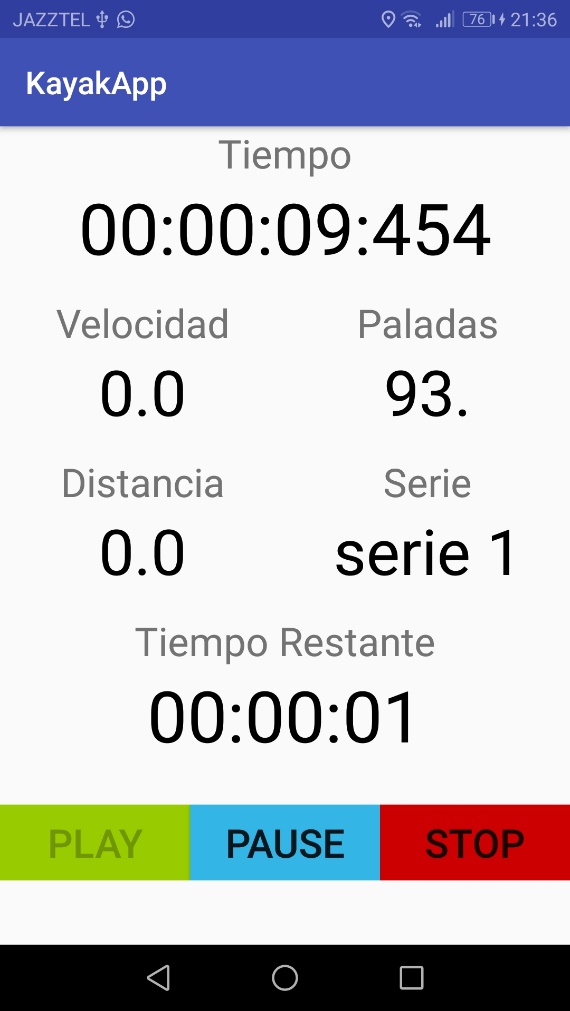


Ilustración 18. Interfaz principal de entrenamiento

En la anterior imagen podemos ver como sería la interfaz donde se nos va a presentar estos datos calculados anteriormente.

En ella podemos ver el tiempo total que llevamos entrenando, la velocidad, la frecuencia de paladas y la distancia que llevamos en ese momento. A parte, podemos ver el número de series que llevamos o si es un descanso y el tiempo restante para terminar esa serie o descanso.

Al terminar esa serie o descanso, al inicio del entrenamiento y al final de este, habrá una realimentación mediante sonidos y vibración para indicar el inicio, cambio o fin.

* 1. **Gestor de entrenamientos**

Los datos de estos entrenamientos se podrán guardar en un fichero de texto plano para su posterior estudio. Se presentarán como un directorio de carpetas y archivos. Los ficheros de datos se guardarán con el siguiente formato:

**“ Tiempo Frecuencia Velocidad Distancia Nº Serie Nº Bloque “**

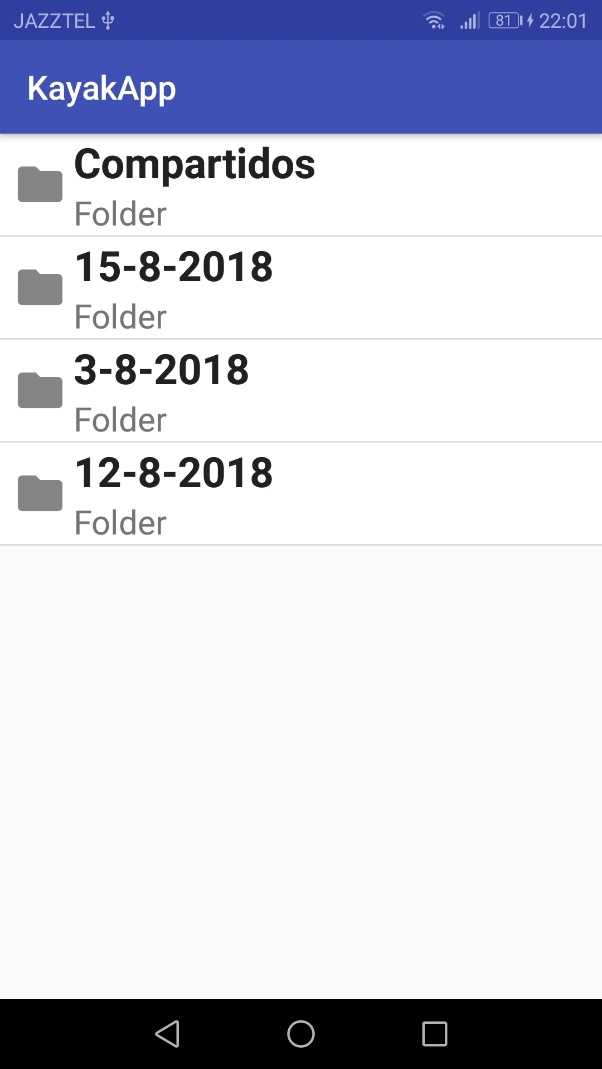


Ilustración 19. Interfaz carpetas de entrenamientos

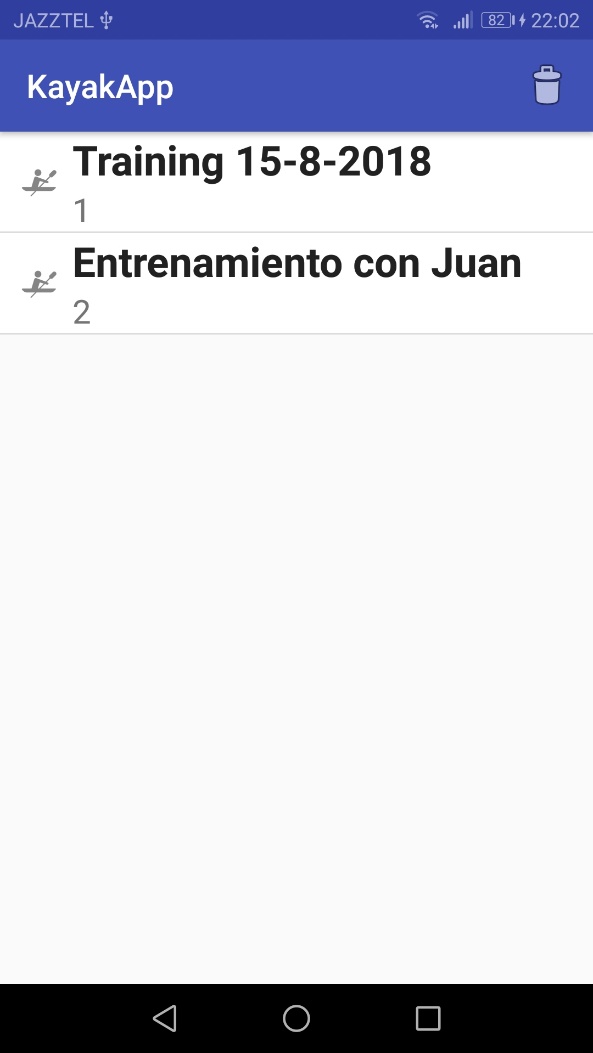


Ilustración 20. Interfaz entrenamientos

* 1. **Gráficos de los entrenamientos**

No solo se podrá guardar un historial de los entrenamientos, sino que se podrán visualizar gráficas de diferentes datos con sus valores correspondientes.

Para ello utilizaré la biblioteca de gráficos “jjoe64:graphview:4.2.1“. Los datos representados serán los que se han guardado en el fichero de texto plano que comentamos en el apartado anterior.

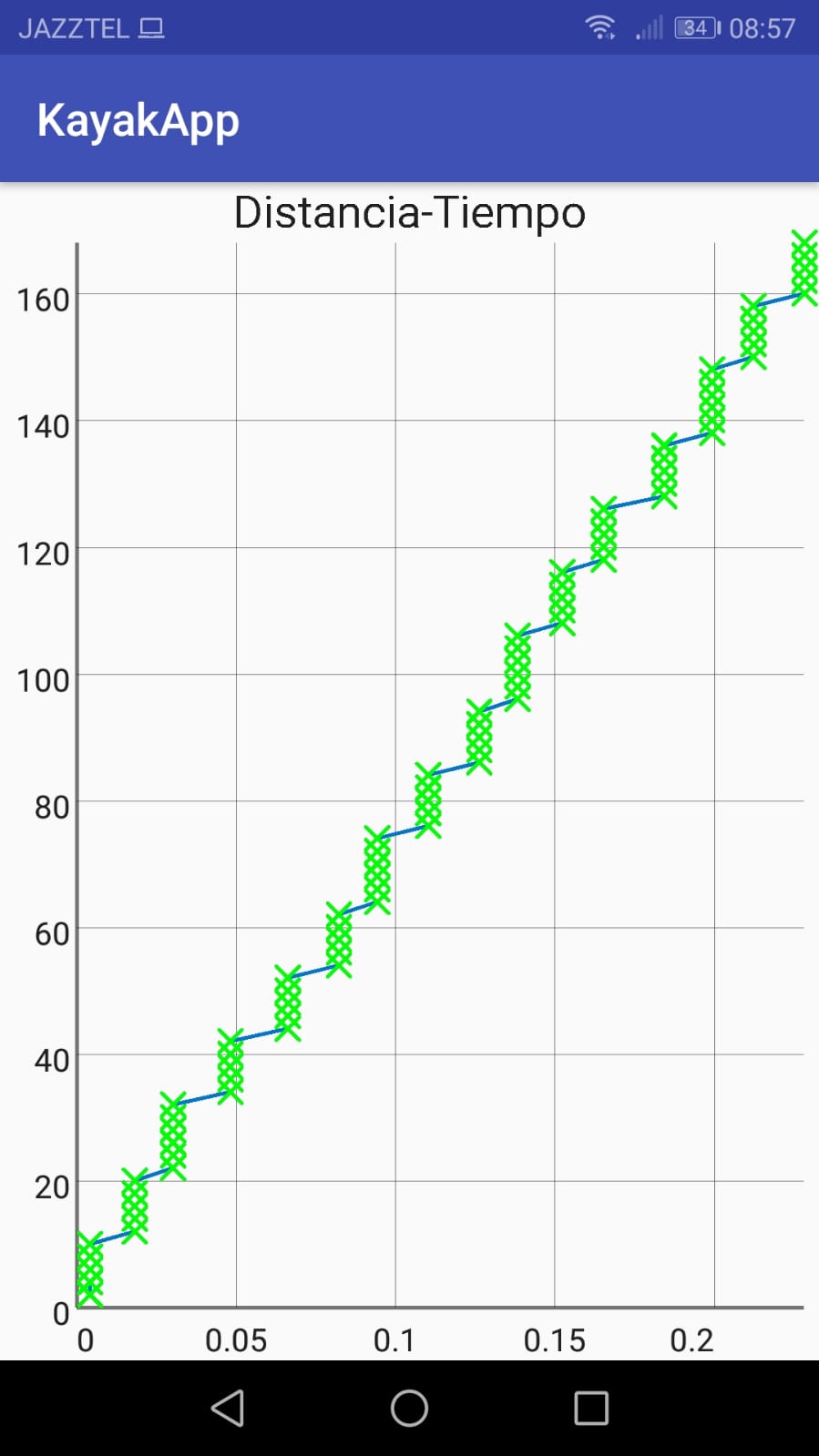


Ilustración 21. Interfaz gráfica datos del entrenamiento

* 1. **Crear entrenamientos personalizados**

Para poder diferenciar entre series, bloques y descansos mientras se está entrenando, se tendrán que crear entrenamientos personalizados.

Estos se guardarán en un fichero XML con el siguiente formato:

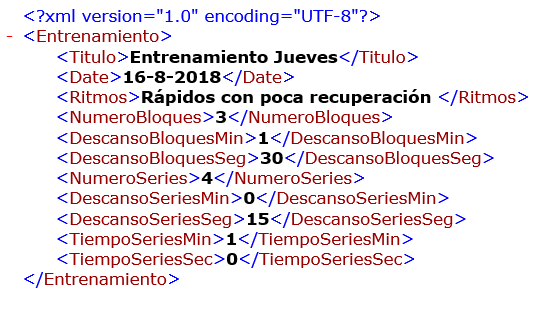


Ilustración 22. XML con los datos de la sesión

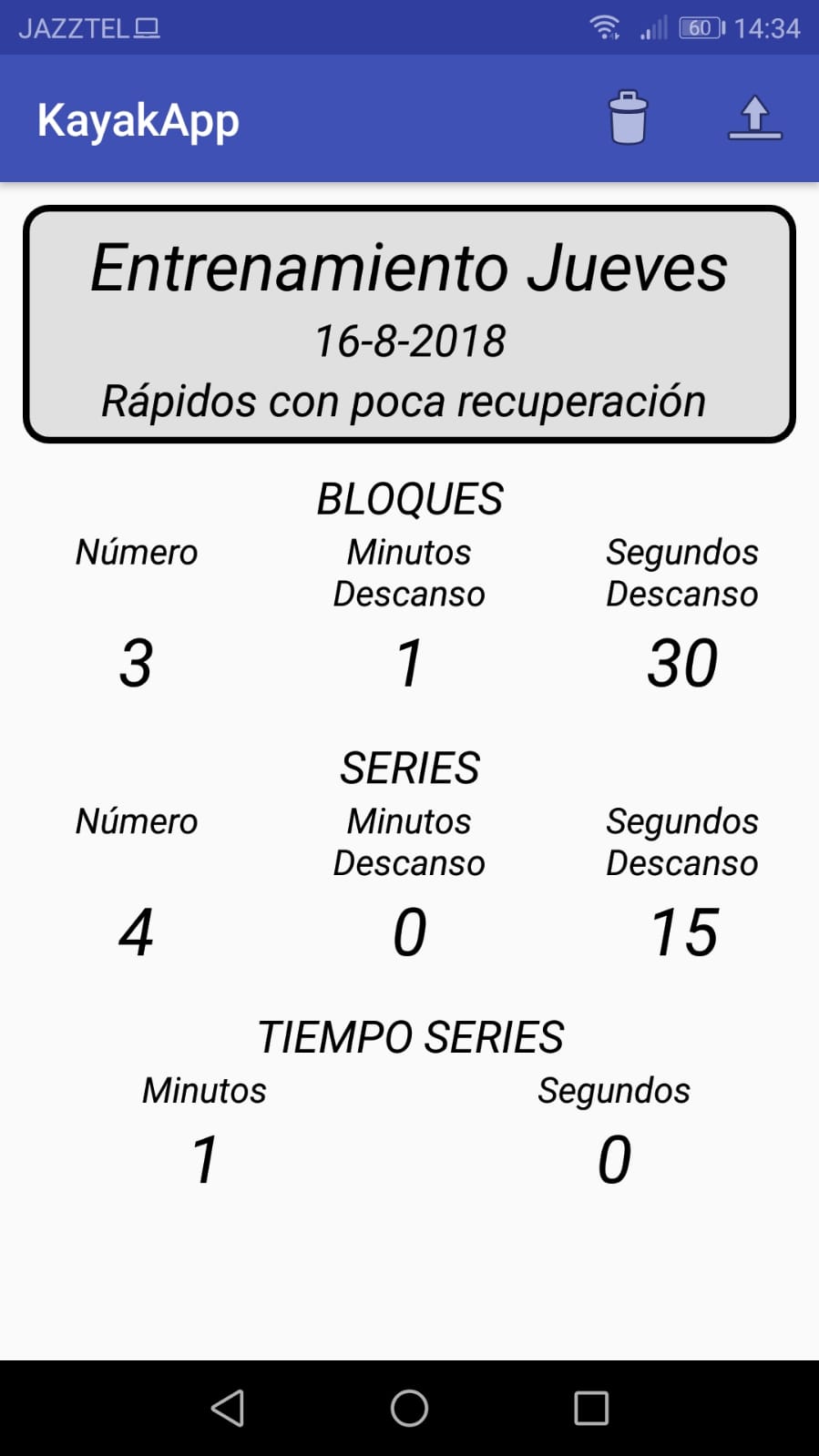


Ilustración 23. Interfaz con los datos de la sesión

* 1. **Compartir entrenamientos con el entrenador de forma inalámbrica**

Una vez realizado un entrenamiento y guardados sus datos, el entrenador podrá recopilarlos de sus palistas, y así, tenerlos guardados para tener un historial y su posterior estudio.

Esto se llevará a cabo gracias a la tecnología Wifi-Direct, que creará una conexión cliente (Palista) servidor(Entrenador) para compartir los datos del entrenamiento.

1. **Aspectos relevantes del desarrollo**

En este apartado se expondrán los detalles y consideraciones más importante del proceso de desarrollo del proyecto final de carrera, desde que problemas se han tenido en su realización, como las posibles soluciones que resolverían el problema, cómo y por qué.

* 1. **Descripción general**

El piragüismo es un deporte acuático que se practica sobre una embarcación ligera, normalmente de fibra o de plástico en embarcaciones de recreo y de kevlar o carbono en embarcaciones de competición.

Esta embarcación de divide en dos grandes conjuntos, la Canoa y el Kayak, siendo la diferencia entre estas dos claras.



Ilustración 24. Canoa



Ilustración 25. Kayak

La embarcación es propulsada por una, dos o cuatro personas, dando lugar al k1, k2 y k4 en caso del kayak, y el c1,c2 y c4 en caso de la canoa.

En cuanto a las competiciones, hay diferentes modalidades:

* Aguas tranquilas



Ilustración 26. K4 femenino de aguas tranquilas

* Eslalon



Ilustración 27. Eslalon

* Descenso de aguas bravas



Ilustración 28. Descenso de aguas bravas

* Ascensos, descensos y travesías



Ilustración 29. Ascensos, descensos y travesías

* Maratón



Ilustración 30. Porteo de un k2 en un maratón

* Kayak polo



Ilustración 31. Kayak polo

* Kayak mar



Ilustración 32. Kayak mar

* Estilo libre



Ilustración 33 Estilo libre

* Barco dragón



Ilustración 34. Barcos dragón

* Piragüismo adaptado



Ilustración 35. Piragüismo adaptado

El desarrollo del proyecto final de carrera comenzó, al darme cuenta de la necesidad de una aplicación deportiva específica para medir los entrenamientos y así ayudar a mejorar el rendimiento deportivo de cada palista.

Seguido del estudio del problema, es decir, como se podría medir los datos en directo de un entrenamiento, en especial, la frecuencia de paladas, ya que no existía ningún sensor que me la diese, al contrario que la ubicación y velocidad con el GPS del teléfono.

La solución, hacer uso del acelerómetro que viene integrado en la totalidad de los móviles actuales. Ya que gracias a los datos que capta y su posterior estudio, se puede calcular en qué momento se realiza una palada y como consecuencia, la frecuencia que se tiene en un momento determinado.

Para ello, desarrollé una aplicación sencilla que recogiese los valores del acelerómetro en un fichero para poder representarlos gráficamente después. No sabía si esta solución iba a funcionar, y para mi sorpresa, los datos del sensor dejaban ver perfectamente cuando se producía una palada. Lo siguiente fue crear un algoritmo para poder diferenciarlas y calcular su media, todo esto se realizó de forma offline sacando los valores del fichero anterior.

Una vez que se consiguió sacar por pantalla la frecuencia de forma offline, se integró en la aplicación, con la diferencia de qué en vez de leer los datos de un fichero, sería el acelerómetro quien se los proporcionase.

El último problema al que me enfrenté fue al decidir como iba a mandar de forma de forma inalámbrica los entrenamientos de los palistas al entrenador. Decidí hacerlo mediante Wifi-Direct, ya que es una tecnología que permite conectar dispositivos entre sí sin necesidad de un router, y así compartir archivos de forma más rápida y con mayor alcance que el Bluetooth. Por no hablar del cifrado mucho más seguro.

* 1. **Ingeniería del Software**

**5.2.1 Metodología**

Para la realización de este proyecto se ha seguido el Proceso Unificado, que se caracteriza, por la utilización del Lenguaje Unificado de Modelado (UML) en las fases del proyecto (especificación de requisitos, análisis, diseño, implementación y pruebas).

El Proceso Unificado es un modelo de proceso basado en componentes y compatible con la filosofía de Programación Orientada a Objetos (POO), que ha sido utilizada en la implementación de la aplicación.

Las características principales de este proceso son:

* Modelo iterativo e incremental.
* Centrado en la arquitectura.
* Dirigido por casos de uso.

Este proceso se repite a lo largo de una serie de ciclos, en los cuales se repiten cuatro fases (Inicio, Elaboración, Construcción y Transición). Cada fase cuenta con una serie de iteraciones que se caracterizan con la consecución de hitos que ayudan a controlar el progreso del proyecto.

**5.2.2 Planificación temporal**

La planificación temporal del proyecto queda realizada en el Anexo I de la documentación.

**5.2.3 Especificación de requisitos**

Para la captura de requisitos funcionales, no funcionales, de información, así como otros requisitos del sistema se ha seguido el método y las plantillas propuestas por Durán y Bernárdez y se ha hecho uso de los diagramas de casos de uso correspondientes.

La especificación de requisitos del proyecto queda realizada en el Anexo II de la documentación.

**5.2.4 Análisis del sistema**

**­­­­** Una vez realizada la elicitación y especificación de los requisitos y objetivos necesarios para el sistema analizando a través de la metodología de Durán y Bernárdez, como parte de la ingeniería de requisitos, se procede al correspondiente análisis de los mismos, completando así las siguientes disciplinas o flujos de trabajo del Proceso Unificado:

* Modelado de negocio
* Requisitos
* Análisis

Dando lugar a los diferentes artefactos (plantillas, diagramas de casos de uso, diagrama de clases, etc.) necesarios para las etapas posteriores del ciclo de vida del Proceso Unificado seguido.

En concreto, en este apartado nos centraremos en el desarrollo del diagrama de clases, diagramas de secuencia y una propuesta de arquitectura (clases de análisis).

El análisis del sistema queda realizado en el Anexo III de la documentación.

**5.2.5. Diseño del sistema**

En esta fase se terminará de definir la arquitectura que permita construir la aplicación siguiendo los principios de alta cohesión y bajo acoplamiento entre módulos y componentes, de tal forma que se pueda construir una aplicación final con un alto grado de calidad, reutilización y fácil mantenimiento.

El diseño del sistema queda realizado en el Anexo III de la documentación.

**5.3. Permisos**

Android es un sistema operativo con privilegios independientes, en el que cada aplicación se ejecuta con una identidad de sistema distinta (ID de usuario de Linux y ID de grupo). También se separan partes del sistema en identidades distintas. Asó, Linux aísla las aplicaciones entre sí y del sistema operativo.

Se ofrecen funciones de seguridad adicionales más precisas mediante un mecanismo de “permisos” que aplica restricciones en las operaciones específicas para que un proceso en particular puede realizar y permisos URI para dar acceso ad-hoc a elementos específicos de datos.

Los permisos que se usan en este proyecto son los siguientes.



Ilustración 36. Permisos de la aplicación

1. **Resultados y conclusiones**

Una vez finalizado el proyecto, se puede concluir, que se han conseguido todos los objetivos funcionales, implementando un servicio que consigue registrar todas las medidas imprescindibles para el control de los entrenamientos de piragüismo, además de permitir su posterior estudio. También se han logrado todas las funciones deseadas en la presentación de este trabajo.

Se ha construido una aplicación Android que funciona en todos los dispositivos móviles (a partir de la versión 4.0 de Android) debido a la gran portabilidad de estas aplicaciones, he de comentar que durante el desarrollo de la aplicación cambié de móvil por motivos externos e hice pruebas en los de mis amigos y familiares dando en todos ellos un resultado satisfactorio.

Todo el diseño y la implementación de este proyecto se ha hecho de manera que cada parte sea independiente de las demás, dotando así a la aplicación de una gran capacidad de reutilización y adaptabilidad ante distintas líneas de trabajo futuro que vayan surgiendo.

Además, cabe destacar, que este trabajo ha sido dotado con una beca conjunta entre la Universidad de Salamanca y el banco Santander llamada YUZZ (anteriormente conocida como beca EXPLORER). Durante los meses de desarrollo, se ha podido adquirir una formación específica sobre emprendimiento, creación de empresas, conocimiento de uno mismo y propiedad industrial.

Para hacer las pruebas de la aplicación sobre la piragua, tuve que meterme en el rio con el móvil. Por lo tanto, tuve que desarrollar una estructura para mantener el móvil seco y fijo en la parte delantera de la piragua, que es donde mejor visualización tiene. Para ello, desarrollé este aparato de forma artesanal:



Ilustración 37. Soporte del móvil 1



Ilustración 38. Soporte del móvil 2

En cuanto al aspecto personal, no solo he disfrutado adentrándome en el mundo de Android y conociendo a fondo los sensores de los dispositivos móviles, sino que he disfrutado realmente desarrollándolo, ya que llevo toda mi vida entrenando sobre una piragua y he podido compaginar mis conocimientos sobre informática y piragüismo aportando mi granito de arena desarrollando esta aplicación para este bonito deporte.

Para terminar, darle las gracias a mi tutor, Guillermo González Talaván por la ayuda y los ánimos que me ha dado durante este curso para realizar este proyecto, y a mi club, el CSCK, por todos estos años encima de una piragua.



Ilustración 39. Tutor y alumno



Ilustración 40. Logo del Club Salamanca Canoe Kayak

1. **Líneas de trabajo futuro**

Una vez finalizado el proyecto, se han pensado y propuesto algunas ampliaciones que podría ser interesante realizar en un futuro:

* Usar otros sensores como el giroscopio y los ejes X y Y del acelerómetro para calcular

el cabeceo y la desviación de remar en línea recta de la piragua mientras se entrena.

Estos movimientos no deseados hacen que parte de la energía desarrollada por el deportista se pierda y no se aproveche para ganar velocidad.

* Crear un servidor con aplicación web para poder alojar de forma online los entrenamientos y su posterior estudio en un ordenador.
* Mejorar y añadir nuevas gráficas y mejores.
* Añadir la posibilidad de una banda de frecuencia cardiaca para añadir sus medidas a los datos de estos entrenamientos, incluyendo las calorías quemadas.
* Extender la aplicación para dispositivos IOS.
* Extender la aplicación a otros idiomas.
* Crear una segunda aplicación que permita recoger al entrenador en directo los datos de los entrenamientos de sus palistas mientras navega al lado de ellos en lancha motora y así tener un control total sobre ellos en directo.
* Por último, crear una nueva funcionalidad de la aplicación que calibre la fuerza de palada del piragüista y vea en que ángulo está colocado el móvil al principio del entrenamiento, ya que la variación de esto puede provocar que la medición de la frecuencia no sea exacta o incluso no funcione.

1. **Referencias**

* Documentación de desarrolladores de Android, donde he sacado la gran mayoría de ayuda para este proyecto <https://developer.android.com/>
* Tratamiento de XML en Android

<http://www.sgoliver.net/blog/tratamiento-de-xml-en-android-i-sax/>

<http://www.sgoliver.net/blog/tratamiento-de-xml-en-android-ii-sax-simplificado/>

<http://www.sgoliver.net/blog/tratamiento-de-xml-en-android-iii-dom/>

<https://gist.github.com/ripper2hl/0b4f1e3ccfc339a40604>

* Libro ‘Desarrollo de aplicaciones para Android. Edición 2018

<https://www.amazon.es/Desarrollo-Aplicaciones-Para-Android-Imprescindibles/dp/8441538921?anstfa-21>

* Sensores de movimiento en Android

<http://www.aprendeandroid.com/l9/sensores_android_movimiento.htm>

* Conexiones P2P para compartir archivos

<https://developer.android.com/training/connect-devices-wirelessly/wifi-direct>

* Cuadros de diálogo

<https://developer.android.com/guide/topics/ui/dialogs>

* Listas personalizadas

<https://androidstudiofaqs.com/tutoriales/adapter-personalizado-en-android>

* Permisos del sistema

<https://developer.android.com/guide/topics/security/permissions?hl=es-419>

* Gráficos Android

<https://javapapers.com/android/android-chart-using-androidplot/>

* Servicios Android

<http://www.androidcurso.com/index.php/tutoriales-android/38-unidad-8-servicios-notificaciones-y-receptores-de-anuncios/288-un-servicio-para-ejecucion-en-segundo-plano>

* Uso del GPS en Android

<https://medium.com/@victor.garibayy/obteniendo-mi-ubicaci%C3%B3n-en-android-studio-377226910823>

* Interfaces de usuario en Android

<https://developer.android.com/guide/topics/ui/?hl=es-419>