Trabajo Fin de Grado

# Detección de barreras urbanas mediante acelerometría

Laura Gómez Pérez

Grado en Ingeniería de Sistemas Audiovisuales

---

Tutor:

Mario Muñoz Organero

# ÍNDICE DE CONTENIDOS

# ÍNDICE DE FIGURAS

# ÍNDICE DE TABLAS

# RESUMEN

La evolución de la telefonía móvil durante los últimos años ha generado un mercado que evoluciona cada día y en el que cada vez se invierte más dinero y tiempo para desarrollar tecnologías que tengan mejores prestaciones. Los teléfonos móviles convencionales se han convertido en smartphones que disponen de todo tipo de herramientas: conexiones Wi-Fi o bluetooth, sensores de movimiento o posición y un sinfín de posibilidades como cámara de fotos y vídeo.

Alrededor del mercado de la telefonía móvil ha surgido un nuevo mercado capaz de desarrollar todo tipo de aplicaciones para estos terminales. Se puede encontrar desde juegos y navegadores hasta todo tipo de sistemas para el reconocimiento de actividades. Dentro de esta última categoría se centra este Trabajo Fin de Grado.

El objetivo de este Trabajo Fin de Grado es la realización de una aplicación móvil para el sistema operativo Android en la que se detecten distintos tipos de barreras urbanas. Lo que se pretende es que, a partir de una serie de datos recogidos en el smartphone del usuario, haciendo uso de los sensores que tiene disponibles, en este caso el acelerómetro y el sensor de gravedad, y usando un clasificador integrado dentro de la aplicación se puedan reconocer patrones de movimiento que permitan identificar los distintos tipos de barreras urbanas.

Para que la aplicación pueda ser utilizada por un gran número de usuarios cuenta con una interfaz diseñada de forma sencilla y que resulta intuitiva por lo que es fácil aprender rápidamente su manejo.

La principal característica de este proyecto es la inclusión de un sistema de clasificación en tiempo real dentro de la misma aplicación. Esto es posible gracias a la librería Weka que entrenada previamente permite generar un modelo de clasificación que puede ser utilizado dentro de una aplicación Android.

# 1 INTRODUCCIÓN

## 1.1 ESCENARIO DEL PROYECTO

Barreras urbanas o arquitectónicas son todos aquellos obstáculos físicos que puedan limitar o impedir la movilidad de ciertos grupos de población por el espacio urbano. [1]

Actualmente, gracias a los smartphones, disponemos de una gran cantidad de aplicaciones móviles que utilizan los sensores presentes en los teléfonos para detectar todo tipo de acciones, como por ejemplo aplicaciones para monitorizar el ciclo de sueño, contar los pasos o registrar la actividad deportiva.

Dada la variedad de sensores disponibles en los dispositivos móviles, existen grandes cantidades de situaciones que pueden ser detectadas.

La aplicación que se expondrá en los siguientes capítulos se centrará en la detección de barreras urbanas. A continuación, se expondrán los objetivos del proyecto, junto con una breve descripción de la estructura de este documento. Después de esto se estudiará el contexto en el que se desarrolla este trabajo junto con el entorno socioeconómico y el marco regulador. Finalmente se planteará la solución propuesta y las líneas de trabajo futuras.

## 1.2 OBJETIVOS

El objetivo de este proyecto es la detección de barreras urbanas utilizando los sensores presentes en los dispositivos móviles. En especial se utilizará el acelerómetro para la detección que estará centrada en escaleras, rampas y suelo llano. Para ello también se utilizará también el aprendizaje automático mediante algoritmos de clasificación.

Realizando un análisis más profundo del proyecto, se deberán cumplir los siguientes objetivos:

- Estudio de los sensores que se van a utilizar en este caso, acelerómetro y gravedad.
- Determinación de todos los parámetros posibles que se pueden obtener para posteriormente utilizarlos en la clasificación.
- Obtención de datos fiables para poder realizar un buen entrenamiento que permita la correcta clasificación de nuevos datos.
- Estudio de los posibles métodos de aprendizaje automático programados en Java para escoger el método que sea más rápido y fiable utilizado en un sistema operativo Android.
- Desarrollo de una aplicación que obtenga los datos de los sensores del teléfono, junto con otros parámetros característicos y que sea capaz de decidir en tiempo real qué actividad está realizando el usuario.
- Realización de pruebas de la aplicación en diferentes dispositivos para comprobar su correcto funcionamiento y realización de los ajustes necesarios.

## 1.3 METODOLOGÍA

El proyecto se ha desarrollado siguiendo una metodología en cascada con un ciclo de vida iterativo creciente, de esta forma el proyecto se ha desarrollado en tres etapas que comparten un mismo esquema. En la Figura 1.1 se puede observar un diagrama de esta metodología.

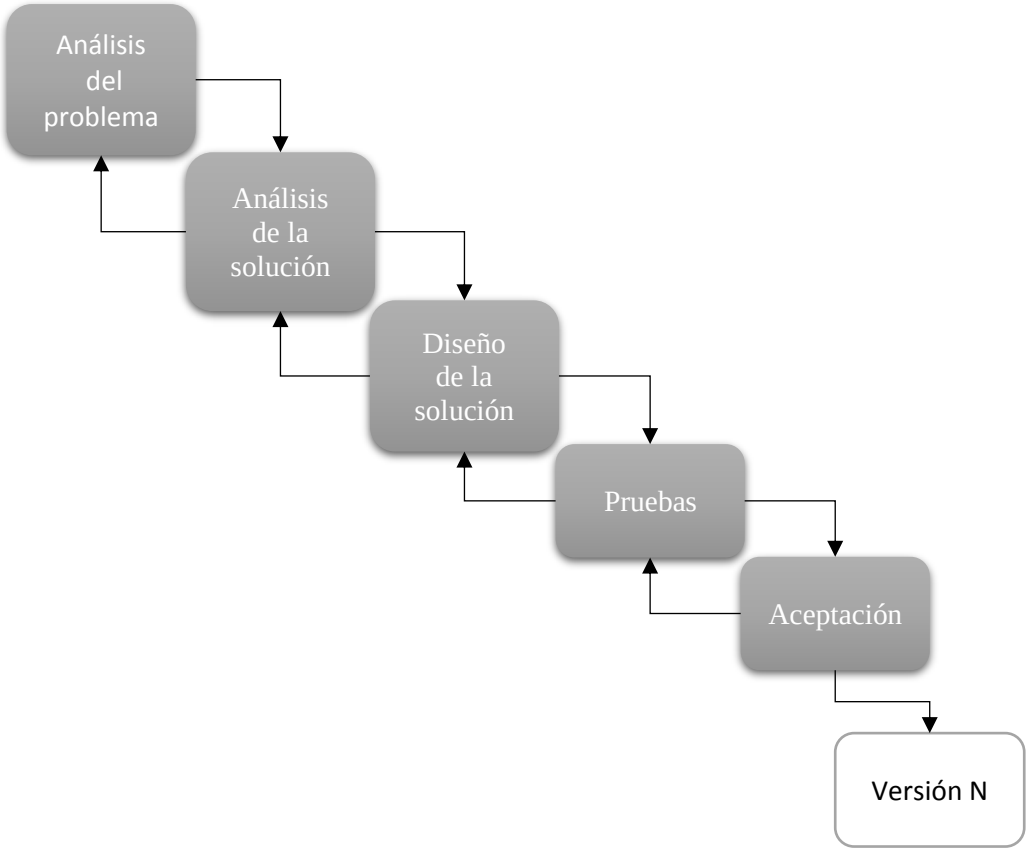


*Figura 1.1. Diagrama general de la metodología de trabajo utilizada.*

Lo primero que se realiza es un análisis del problema que se plantea. Después, se analizan los beneficios de las posibles soluciones. Una vez elegida una solución se pasa a desarrollarla y a realizar una serie de pruebas para evaluar que es adecuada, en caso de no satisfacer las expectativas se modificará su diseño y éste volverá a ser evaluado. Una vez se obtienen los resultados deseados se pasa a diseñar la siguiente etapa del proyecto hasta concluir la última fase.

## 1.4 ESTRUCTURA DE LA MEMORIA

El presente documento está estructurado en ocho capítulos y dos anexos organizados de la siguiente forma:

- **Capítulo 1**: pequeña introducción sobre el escenario y los objetivos que se pretenden cubrir con la realización de este trabajo. También se incluye la metodología de trabajo utilizada y la estructura de la memoria.
- **Capítulo 2**: se analiza la situación actual de los aspectos principales relacionados con el desarrollo de una aplicación móvil, centrándose en las tecnologías que van a ser

utilizadas. Además, se realiza una pequeña comparativa de la aplicación desarrollada con otras similares presentes en el mercado.

- **Capítulo 3**: se analiza el marco regulatorio que afecta al desarrollo de aplicaciones móviles.
- **Capítulo 4**: incluye el presupuesto y planificación de la elaboración del proyecto junto con un detallado análisis del impacto socioeconómico derivado del uso de la aplicación.
- **Capítulo 5**: describe cómo se ha diseñado la aplicación describiendo la arquitectura utilizada y los requisitos que debe cumplir.
- **Capítulo 6**: expone los pasos seguidos para la implementación de la aplicación. Contiene una detallada explicación de las decisiones de diseño tomadas por los desarrolladores, una presentación de la imagen de la aplicación y una explicación de las clases que componen la aplicación.
- **Capítulo 7**: presenta todos los pasos seguidos para verificar el correcto funcionamiento del sistema desarrollado.
- **Capítulo 8**: extrae una serie de conclusiones sobre el trabajo realizado e introduce unas líneas que deberían seguirse en un futuro para continuar mejorando la aplicación.
- **Anexo I**: resumen extendido del presente trabajo.
- **Anexo II**: versión en inglés del resumen extendido.

# 2 ESTADO DEL ARTE

En este capítulo nos centraremos en dos cuestiones. En primer lugar, estudiaremos las tecnologías utilizadas para la realización de la aplicación, junto con las diferentes tecnologías que se encuentran disponibles en el mercado. En segundo lugar, se explicarán las aplicaciones que existen actualmente en el mercado que utilizan tecnologías similares.

## 2.1 ESTUDIO DE LAS TECNOLOGÍAS UTILIZADAS

### 2.1.1 Sistemas operativos móviles

Actualmente existen tres grandes desarrolladores de sistemas operativos móviles:

1. Android
2. iOS
3. Windows Phone

En la Figura 2.1 se puede observar la evolución de la cuota de mercado de estos sistemas operativos:

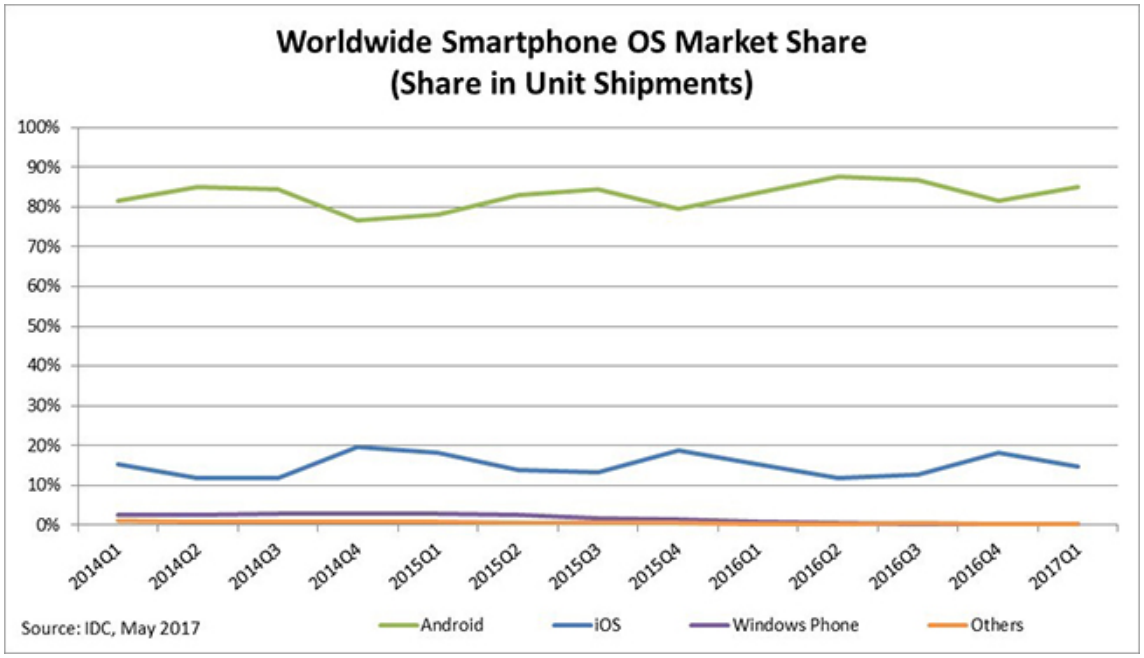


*Figura 2.1. Datos de ventas de smartphones desde 2014. Figura extraída de IDC [2]*

A continuación, vamos a ver cuáles son las principales características de cada uno de los sistemas operativos.

#### *2.1.1.1 Android*

Android es una plataforma de código abierto desarrollada por Google junto con la Open Handset Alliance. El sistema operativo está basado en Linux y el entorno de ejecución está basado en Java.

Este sistema operativo tiene una licencia libre y de código abierto, lo que facilita que los desarrolladores puedan modificarla e instalarla en cualquier dispositivo móvil.

La creación de nuevas aplicaciones puede hacerse desde cualquier ordenador (Linux, Windows y Mac OS X) utilizando Eclipse con un paquete de extensión [3] o usando Android Studio [4] que actualmente está considerado como entorno oficial de desarrollo.

Desde el lanzamiento de la primera versión en 2008 ha ido evolucionando en sucesivas versiones:

- 1.0 – Apple pie (septiembre 2008)
- 1.1 – Petit four (febrero 2009)
- 1.5 – Cupcake (abril 2009)
- 1.6 – Donut (septiembre 2009)
- 2.0/2.1 – Eclair (octubre 2009)
- 2.2 – Froyo (mayo 2010)
- 2.3 – Gingerbread (diciembre 2010)
- 3.0/3.1/3.2 – Honeycomb (febrero 2011)
- 4.0 – Ice cream sándwich (octubre 2011)
- 4.1/4.2/4.3 – Jelly bean (julio 2012 – julio 2013)
- 4.4 – KitKat (octubre 2013)
- 5.0 – Lollipop (noviembre 2014)
- 6.0 – Marshmallow (octubre 2015)
- 7.0 – Nougat (agosto 2016)

Dentro de las diferentes versiones de Android actualmente el mercado se encuentra distribuido de la siguiente forma:

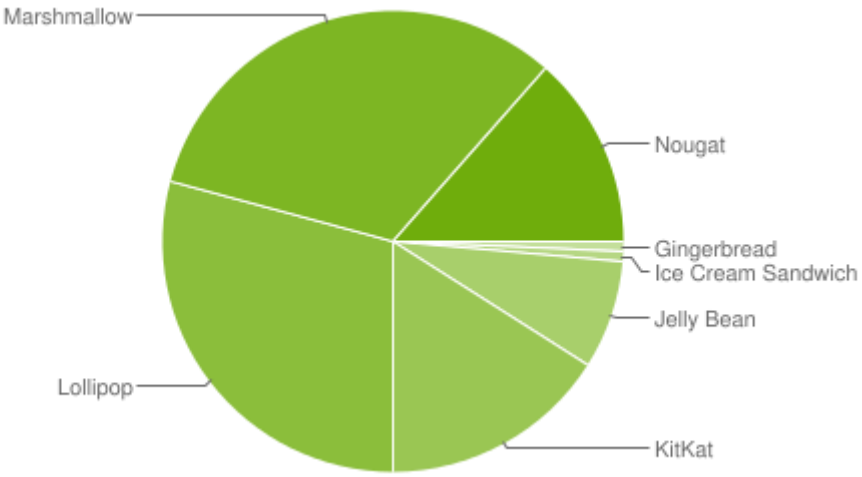


*Figura 2.2. Cuota de mercado de las diferentes versiones Android. Figura extraída de Developers Android [5]*

Las aplicaciones se introducen en el mercado a través de Google Play (anteriormente conocida como Android Market). Para ello los usuarios se deben registrar como desarrolladores en Google Play Console y pagar una cuota de 25$. Después de esto se podrán subir las aplicaciones incluyendo la ficha de descripción de la aplicación que aparecerá en Google Play y la clasificación del contenido.

#### *2.1.1.2 iOS*

Es el sistema operativo que utilizan los móviles Apple Inc. Originalmente fue desarrollado para iPhone, aunque después de ha utilizado en otros dispositivos como iPod o iPad.

Este sistema salió a la venta en 2007 y sólo puede ser instalado en dispositivos de la propia marca. Al igual que en Android ha ido evolucionando en una serie de versiones:

- iPhone OS 1.0
- iPhone OS 2.0
- iPhone OS 3.0 (junio 2009)
- iOS 4 (junio 2010)
- iOS 5 (junio 2011)
- iOS 6 (junio 2012)
- iOS 7 (junio 2013)
- iOS 8 (junio 2014)
- iOS 9 (septiembre 2015)
- iOS 10 (septiembre 2016)
- iOS 11 (septiembre/octubre 2017)

En la siguiente imagen se puede ver cómo está distribuido el uso de las diferentes versiones de los sistemas operativos iOS:

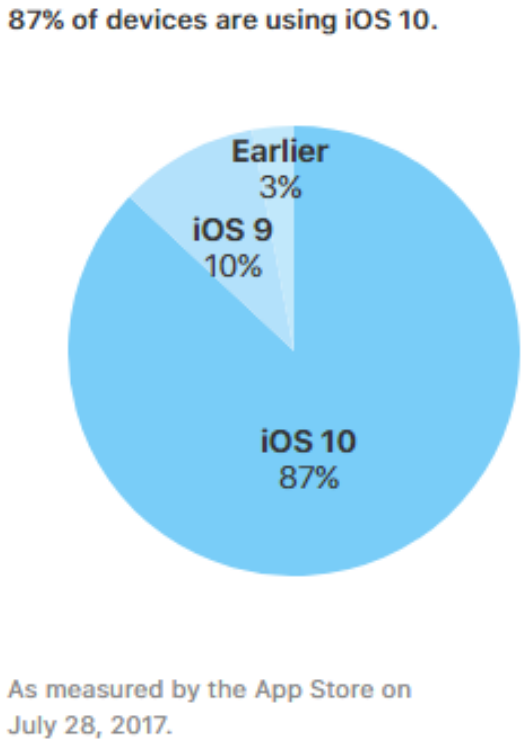


*Figura 2.3. Distribución de mercado de las versiones iOS. Figura extraída de Support App Store [6]*

En cuanto a las opciones para desarrollar aplicaciones cuenta con un kit de desarrollo de software desde el 2008, aunque al contrario que en Android sólo está disponible para Mac OX 10.6.6 o posterior. El entorno de desarrollo que está disponible en los dispositivos es XCode. En este caso el lenguaje está orientado a objetos y basado en C.

La publicación de aplicaciones se realiza a través de App Store y para ello es necesario adquirir una licencia que permite su distribución por un precio de 99$ al año.

#### *2.1.1.3 Windows Phone*

Es el sistema operativo móvil desarrollado por Windows. Está diseñado como una versión reducida de Windows por lo que mantiene la interfaz de usuario y las aplicaciones Office.

Salió a la venta en octubre de 2010 como sustituto de la anterior versión Windows Mobile. Desde entonces se han desarrollado las siguientes versiones:

- Windows Phone 7 (octubre 2010)
- Windows Phone 8 (octubre 2012)
- Windows Phone 8.1 (abril 2014)
- Windows 10 Mobile (marzo 2016)

En la siguiente imagen se puede ver cómo está distribuido el mercado en función de las diferentes versiones a finales de 2016:

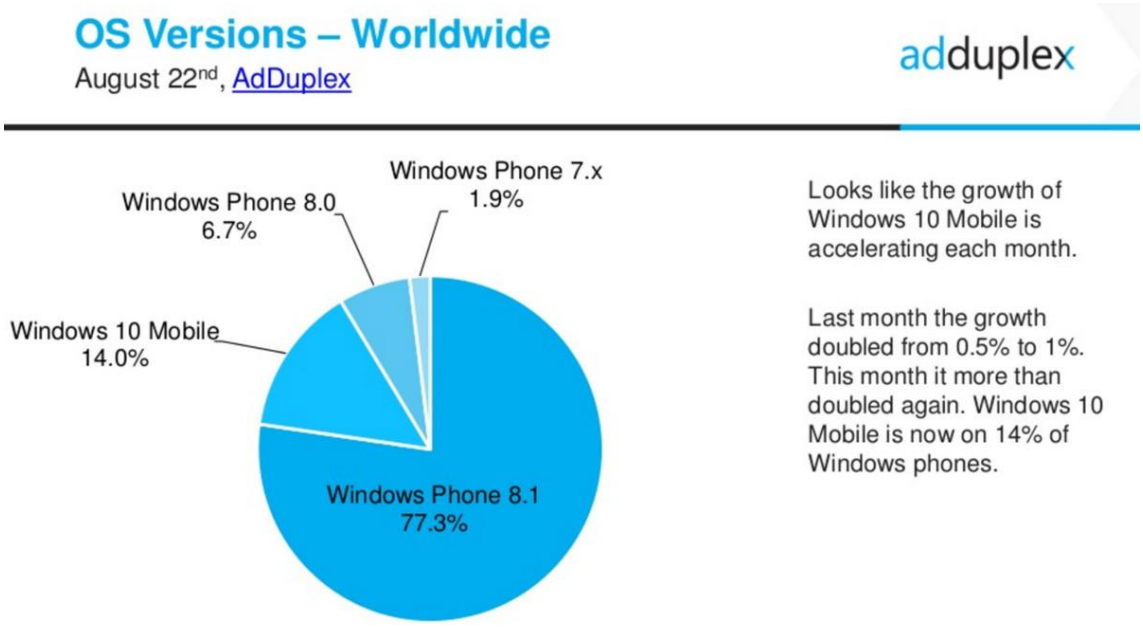


*Figura 2.4. Distribución del mercado de las distintas versiones de Windows Phone. Figura extraída de Windows Central [7]*

En cuanto al entorno para el desarrollo de aplicaciones utiliza Visual Studio junto con el Windows Phone SDK. En la actual versión (Windows 10 Mobile) está disponible la herramienta Windows App Studio (en línea) con la que se puede elaborar la aplicación que después se podrá subir a Windows Phone Store estando registrado como desarrollador y habiendo sido certificada previamente.

#### *2.1.1.4 Elección del sistema operativo*

Según lo descrito en las secciones anteriores podemos evaluar los distintos sistemas operativos teniendo en cuenta tres factores: número de usuarios, facilidad para la creación de nuevas aplicaciones y coste.

En cuanto al número de usuarios como se ha visto en la Figura 2.1 Android es el sistema operativo que cuenta con el mayor número de usuarios, seguido de iOS y Windows Phone que posee un mercado minoritario junto con otros desarrolladores como BlackBerry OS o Symbian.

Respecto a la facilidad para la creación de nuevas aplicaciones en primer lugar tendríamos al sistema Android ya que Android Studio es compatible con la mayoría de sistemas operativos de ordenadores, además de que es relativamente sencillo probar las aplicaciones mientras se desarrollan dado que cuenta con un emulador y también puede probarse en un terminal móvil. Después estaría Windows Phone ya que el desarrollo de aplicaciones parece relativamente sencillo al igual que en Android, especialmente para la versión Windows 10 Mobile. En último lugar estaría el desarrollo de aplicaciones para iPhone ya que es necesario contar tanto con un ordenador como con un móvil Apple.

Por último, teniendo en cuenta el coste de lanzar las nuevas aplicaciones al mercado el sistema operativo más restrictivo es iOS dado que requiere pagar una cuota anual elevada. En este aspecto Android y Windows Phone están igualados teniendo los dos una cuota única que se paga para darse de alta como desarrollador y poder subir aplicaciones a las respectivas tiendas. Aunque la cuenta de desarrollador de Windows es un poco más barata que la de Android, 19$ frente a 25$.

Por todo esto la aplicación se va a desarrollar sobre Android, principalmente porque va a ser accesible para un mayor número de usuarios y por la facilidad para el desarrollo de aplicaciones. Se va a trabajar para una versión 4.4.2 (KitKat) que pese a no ser la que tiene más terminales actualmente en el mercado (16%), sí que es compatible con las siguientes versiones. Esta versión utiliza un API nivel 19.

### 2.1.2 Sensores

Los teléfonos móviles que hay actualmente en el mercado presentan una serie de características que permiten obtener una gran cantidad de información. Dentro de estas características estarían los sensores. Android soporta tres categorías de sensores:

- **Sensores de movimiento**: estos sensores miden las fuerzas de aceleración y de rotación en tres ejes. Dentro de esta categoría tenemos acelerómetros, sensores de gravedad o giroscopios.
- **Sensores ambientales**: miden condiciones ambientales como la temperatura, la presión atmosférica, la humedad o la luz. En esta categoría tenemos termómetros, barómetros o fotómetros.
- **Sensores de posición**: estos sensores miden la posición física de un dispositivo. Esta categoría incluye magnetómetros y sensores de orientación. [8]

Dentro de estos sensores tenemos algunos basados en hardware y otros basados en software.

A continuación, describiremos los sensores que se utilizan dentro de la aplicación que se está describiendo en este trabajo.

#### *2.1.2.1 Acelerómetro*

El acelerómetro mide la aceleración (cambio de velocidad) que está siendo aplicada al dispositivo en cada uno de los tres ejes. Esta medida está influida por el valor de la gravedad por lo que cuando el dispositivo está en reposo el valor obtenido es 9.81 $m/s^2$. Por lo tanto, para obtener un valor real de la aceleración va a ser necesario eliminar la componente de la gravedad de la medida que se obtiene de la aceleración.

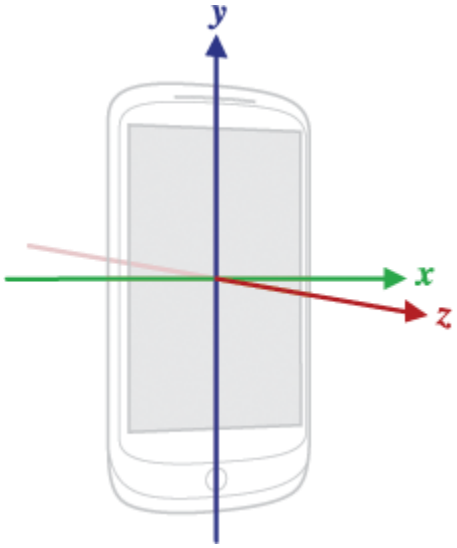


*Figura 2.5. Ejes de coordenadas utilizados por los sensores. Figura extraída de Developers Android [8]*

#### *2.1.2.2 Sensores de gravedad*

El sensor de gravedad proporciona información sobre la magnitud de la gravedad en los tres ejes. Habitualmente se utiliza para determinar la orientación del dispositivo en el espacio.

Para la aplicación que estamos desarrollando el valor que nos proporciona el sensor de gravedad se va a utilizar para eliminar la componente de la gravedad de los valores que obtiene el acelerómetro, como se ha comentado antes.

### 2.1.3 Weka para Android

Weka es una colección de algoritmos de aprendizaje automático para tareas de minería de datos. Contiene herramientas para el preprocesamiento de datos, clasificación, regresión, clustering, reglas de asociación y visualización. [9]

Weka dispone de una serie de librerías programadas en Java con unos algoritmos que podemos llamar dentro de nuestro código Java. El problema es que esta librería no se puede utilizar en Android dado que utiliza código referente a la interfaz del programa. A pesar de ello, está disponible una versión compatible con Android disponible gratuitamente mediante el repositorio GitHub [10]. La adaptación para Android está basada en la versión 3 de Weka, aunque no se garantiza la completa funcionalidad dado que en algunos casos puede no ser estable. Para la aplicación que estamos desarrollando las pruebas realizadas indican que sí es estable dentro de nuestro código.

## 2.2 APLICACIONES SIMILARES EN EL MERCADO Y CASOS DE USO

A continuación, se detallarán los posibles usos que puede tener nuestra aplicación con el fin de poder compararla con otras aplicaciones similares que podemos encontrar en el mercado actualmente.

### 2.2.1 Usos para la aplicación desarrollada

La detección de barreras urbanas podría utilizarse con distintas finalidades. En primer lugar y como principal finalidad se usaría para mejorar la vida de personas con algún tipo de minusvalía que limite el movimiento.

La aplicación podría posicionar dentro de un mapa las barreras urbanas que hay en una ciudad, así estas personas sabrían dónde van a poder encontrar algún obstáculo que les va a impedir moverse con facilidad. Para ello se necesitaría que muchos usuarios registraran los datos de las barreras urbanas de la ciudad para así verificar que los mapas generados son fiables y no tienen falsos positivos. Este tipo de aplicación también podría ser de interés para los gobiernos locales que podrían utilizarlos para mejorar la ciudad y hacer que sea más accesible.

Por otra parte, también podría utilizarse para promover un estilo de vida saludable dado que la aplicación podría registrar el tiempo que una persona está activo y el tiempo que dedica a cada una de actividades que registra la aplicación en forma de barreras urbanas.

### 2.2.2 Aplicaciones del mercado

A continuación, vamos a ver una serie de aplicaciones disponibles en el mercado que son semejantes a la principal finalidad que tiene la aplicación que estamos desarrollando.

#### *2.2.2.1 Accieo*

Es una aplicación que se centra en indicar qué lugares son accesibles dentro de la ruta de un usuario y permite añadir una descripción del tipo de accesibilidad que tienen nuevos lugares.

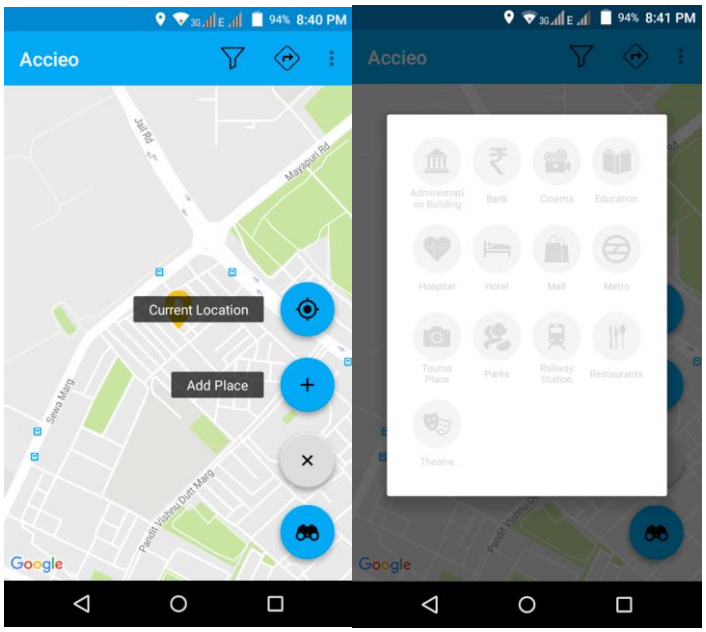


*Figura 2.6. Imágenes de la aplicación Accieo. Figura extraída de Play Store [11]*

Si la comparamos con el uso que pretendemos darle a la aplicación que estamos desarrollando se diferencia en un aspecto fundamental, Accieo sólo registra tipos de lugar y su localización junto con la descripción de accesibilidad. Lo que nuestra aplicación haría en su uso final sería detectar el tipo de barrera urbana que hay en cada lugar independientemente del tipo de lugar,

es decir, ya sea en la calle o dentro de un restaurante, por ejemplo. De esta forma ya quedaría indicado el tipo de accesibilidad que hay en cada localización.

#### *2.2.2.2 MEP APP*

Esta aplicación surge para que las personas con problemas de movilidad puedan desplazarse con libertad por las ciudades. La aplicación permite que los usuarios registren la accesibilidad de los lugares y genera rutas con niveles asequibles de accesibilidad.

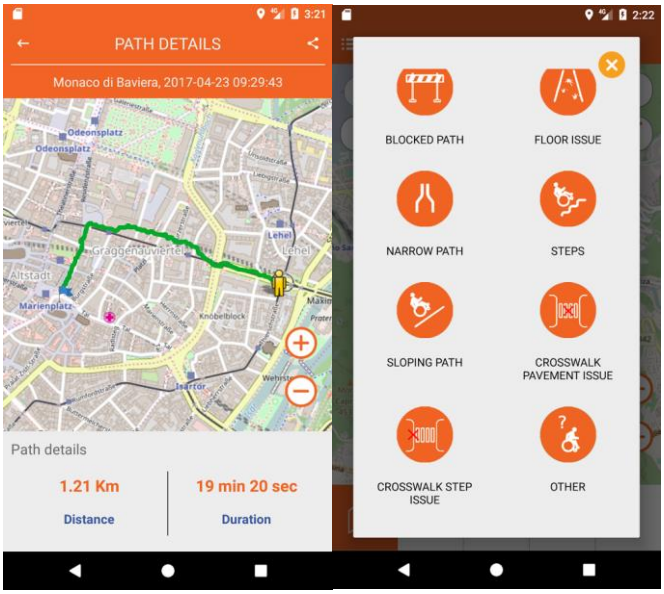


*Figura 2.7. Imágenes de la aplicación MEP APP. Figura extraída de Play Store [12]*

MEP APP es aparentemente una aplicación similar a la que se puede desarrollar finalmente con la aplicación para la detección de barreras urbanas desarrollada en este trabajo. Aunque en nuestra aplicación se diferenciaría en que incluiría la opción para que los usuarios soliciten accesibilidad en los lugares que tengan barreras urbanas que la limiten.

#### *2.2.2.3 Zaragoza Accesible*

La aplicación tiene registrada una serie de lugares accesibles dentro de la ciudad y permite solicitar accesibilidad a lugares que no sean fácilmente accesibles.

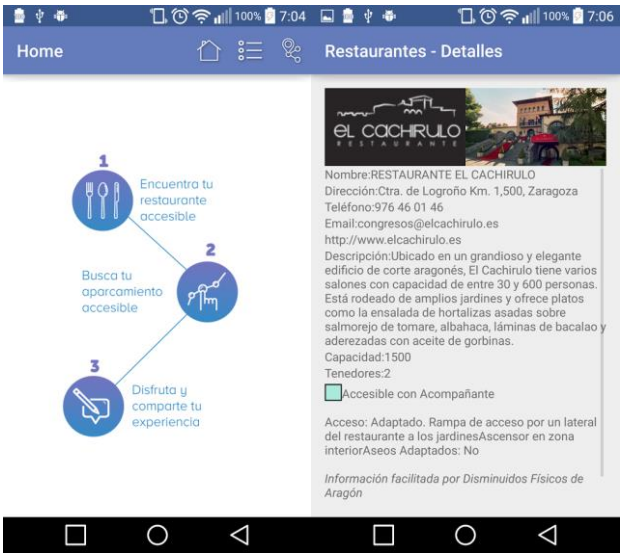


*Figura 2.8. Imágenes de la aplicación Zaragoza Accesible. Figura extraída de Play Store [13]*

En este caso la aplicación Zaragoza Accesible es la más limitada, dado que se centra en una sola ciudad y únicamente actúa como una guía con la información que posee. Si la comparamos con nuestra futura aplicación la característica más distintiva es que nuestra aplicación estaría disponible para cualquier localización de la que los usuarios hayan dado información a los desarrolladores.

# 3 MARCO REGULADOR

## 3.1 ANÁLISIS DE LA LEGISLACIÓN APLICABLE

En la creación de aplicaciones móviles hay que tener en cuenta una serie de leyes y normativas de cara a su distribución.

En primer lugar, tenemos que tener en cuenta la protección de datos de carácter personal. Para ello son aplicables la Directiva 95/46/CE que crea un marco regulador sobre la circulación de datos personales dentro de la Unión Europea. A nivel nacional esta directiva se transpone en la Ley Orgánica de Protección de Datos (en adelante LOPD).

Las aplicaciones también deben cumplir la Ley de Servicios de la Sociedad de la Información y Comercio Electrónico (en adelante LSSI).

Por último, hay que tener en cuenta las políticas de privacidad. En este caso al tratarse de una aplicación desarrollada en el entorno Android se deberá cumplir la política que exige Play Store a los desarrolladores para la publicación de aplicaciones.

A continuación, se verá con detalle cada una de las normas anteriormente mencionadas.

### 3.1.1 Directiva 95/46/CE

Esta directiva es el texto de referencia a nivel europeo en cuanto a la protección de datos personales. Por lo tanto, constituye el marco legal aplicable a cualquier aplicación disponible a nivel europeo.

Se basa en que el tratamiento de los datos sea lícito, de forma que los datos obtenidos se hayan dado de forma consentida por el interesado, sean necesarios para el cumplimiento de un contrato o sean necesarios para el interés del responsable del tratamiento (entre otros). Establece unos principios de calidad para el tratamiento de los datos:

- Los datos recogidos tendrán un fin concreto y explícito. Serán actualizados cuando sea necesario y serán almacenados durante el tiempo necesario para los fines que fueron recogidos.
- Está prohibido recoger datos de opiniones políticas, origen racial, convicciones religiosas o relativos a la salud.

Las personas cuyos datos son tratados tienen los siguientes derechos:

- Derecho a obtener información. Pueden solicitar al responsable del tratamiento información sobre los fines para los que se recogen los datos, destinatarios de los datos, etc.
- Derecho de acceso a los datos del interesado.
- Derecho a oponerse al tratamiento de los datos.

También se podrá limitar el acceso a los datos para salvaguardar la seguridad del Estado, la seguridad pública o el interés económico o financiero de un país de la Unión Europea. El responsable del tratamiento de los datos deberá garantizar la seguridad de éstos, la destrucción, pérdida o el acceso no autorizado. Notificación del tratamiento a la autoridad de control, ésta

realizará unas comprobaciones para garantizar que no hay riesgo en los derechos y libertades de los interesados. [14]

Por último, las autoridades de cada país deben tener un recurso judicial para el caso en el que el responsable del tratamiento de los datos no respete los derechos anteriormente mencionados.

### 3.1.2 LOPD

La Ley Orgánica de Protección de Datos de Carácter Personal o LOPD se encarga de garantizar el trato y uso correcto de datos de carácter personal.

Para cumplir con esta ley las aplicaciones deben cumplir los siguientes aspectos:

- Solicitar a los usuarios el consentimiento con carácter previo y específico.
- Cumplir con las obligaciones responsables cuando se tratan datos de los usuarios.
- Permitir a los usuarios rescindir los consentimientos y desinstalar la aplicación.
- Facilitar una política de privacidad inteligible.
- Indicar la finalidad con la que serán utilizados los datos obtenidos de la aplicación.
- Establecer unos períodos de almacenamiento de datos.
- Facilitar a los usuarios el ejercicio de los derechos ARCO (acceso, rectificación, cancelación y oposición). [15]

En general, lo que se debe procurar es tener una comunicación clara con los usuarios y sólo solicitar el acceso a los datos que sean estrictamente necesarios.

Además, se deben dar de alta los ficheros en la Agencia Española de Protección de Datos, se debe tener un documento de seguridad en el que se indiquen todas las medidas que se realizan para la protección de los datos de carácter personal. Si los datos personales van a ser tratados por un tercero se debe tener un contrato de tratamiento de datos siguiendo lo establecido en el artículo 12 de la LOPD. [16]

### 3.1.3 LSSI

La Ley de Servicios de la Sociedad de la Información y Comercio Electrónico incorpora en la legislación española la Directiva 2000/31/CE que regula a nivel europeo aspectos jurídicos en los referente a comercio electrónico.

La LSSI, en este sentido, establece tanto a los proveedores de servicios de intermediación, como a las empresas que ofrecen sus productos y a los ciudadanos que posean una página web, las reglas necesarias para que el uso y disfrute de esta red, así como la posible actividad económica generada en torno a la compra y venta de todo tipo de productos y servicios, sea una experiencia positiva, segura y confiable. [17]

En este caso, aunque las aplicaciones sean gratuitas sí que tendrían que cumplir esta ley si reciben dinero por parte de publicidad o patrocinadores.

La aplicación debe contener la información que identifique a su responsable (datos de contacto, NIF, domicilio, etc.). También, en el caso de que vaya a enviar algún tipo de publicidad a los usuarios, debe contener un lugar donde los usuarios den su consentimiento y este envío de publicidad debe estar indicado en todo momento como tal.

### 3.1.4 Política de privacidad

La política de privacidad es el lugar en el que se describe la forma en la que nuestra aplicación recoge los datos de carácter personal y el tratamiento que hace con ellos. En ella se debe detallar cómo se adecua nuestra política a la LOPD y a la LSSI. Por lo tanto, es un texto que tiene que tener cada aplicación de forma independiente.

Además, las aplicaciones con base Android también deben cumplir los términos de la política de privacidad de Google Play dado que si no lo hacen serán eliminadas de la tienda y por lo tanto no tendrán forma de distribuirse.

#### *3.1.4.1 Política de privacidad de Google Play*

Respecto a los datos del usuario, si la aplicación va a obtener cualquier tipo de dato personal del usuario tiene que contener una política propia de privacidad donde explique cómo se van a obtener, almacenar y tratar los datos. Tiene que tener alguna tecnología para proteger los datos y almacenarlos de forma segura. Además, si la aplicación obtiene algún dato personal que no sea relevante para lo indicado en la descripción de la aplicación tendrá que obtener permiso expreso del usuario dentro de la propia aplicación, no valdrá sólo con incluirlo dentro de la política de privacidad.

De esta forma la política de privacidad de Google Play obliga a los usuarios a cumplir con la normativa europea y por tanto también con la legislación española que se ha comentado en las secciones anteriores.

# 4 ENTORNO SOCIOECONÓMICO

## 4.1 PRESUPUESTO Y PLANIFICACIÓN

En este apartado se describirá el presupuesto detallado de la realización del proyecto, incluyendo los gastos del material utilizado, así como los gastos de personal. También se incluye una panificación temporal del desarrollo del proyecto.

### 4.1.1 Presupuesto

En primer lugar, vamos a detallar los costes materiales. Para la realización de la aplicación se ha necesitado un ordenador y dos teléfonos móviles para realizar las pruebas. La Tabla 1 incluye el precio de cada equipo junto con el factor de amortización estimado.

| Equipo | Coste (€) | Factor de amortización | Coste total (€) |
|---|---|---|---|
| Ordenador | 600 | 1/4 | 150 |
| Móvil: Bq Aquaris U Lite | 150 | 1/5 | 30 |
| Móvil: Wiko Pulp | 130 | 1/5 | 26 |
| Software: Matlab R2015b | 500 | 1/3 | 166,6 |
| Material de oficina | 15 | 1 | 15 |
| Coste total | | | 387,6 |

*Tabla 1. Costes materiales*

A continuación, veremos el desglose de los costes de personal.

| Personal | Descripción | Coste/hora (€) | Horas | Coste (€) |
|---|---|---|---|---|
| Laura Gómez Pérez | Ingeniero | 10 | 620 | 6200 |
| Mario Muñoz Organero | Ingeniero (Dr.) | 35 | 40 | 1400 |
| Coste total | | | | 7600 |

*Tabla 2. Costes de personal*

| Descripción | Coste (€) |
|---|---|
| Costes materiales | 387,6 |
| Costes personales | 7600 |
| Subtotal | 7987,6 |
| IVA (21%) | 1677,4 |
| Total | 9665 |

*Tabla 3. Coste total del proyecto*

Por lo tanto, el presupuesto total del proyecto asciende a un total de nueve mil seiscientos sesenta y cinco euros.

### 4.1.2 Planificación temporal

En la Tabla 4 se muestra un resumen de las actividades desarrolladas durante la elaboración del proyecto, así como la duración de cada una de ellas. A partir de esta planificación se ha desarrollado un diagrama de Gantt (Figura 4.1) que muestra la evolución temporal (en semanas) de dichas actividades.

| ID | Actividad | Duración (semanas) |
|---|---|---|
| | Fase 1 | |
| A | Estudio de la propuesta y documentación sobre los sensores y ficheros. | 2 |
| B | Primeras pruebas de la obtención de datos de los sensores y su almacenaje en ficheros. | 3 |
| C | Obtención de datos y ajuste del almacenamiento | 2 |
| | Fase 2 | |
| D | Documentación sobre las posibles características a utilizar | 1 |
| E | Pruebas de la extracción de características | 8 |
| F | Primera versión del algoritmo extractor de características | 3 |
| | Fase 3 | |
| G | Traslado del algoritmo a la aplicación | 5 |
| H | Documentación sobre posibles algoritmos de clasificación | 2 |
| I | Pruebas de posibles algoritmos de clasificación | 1 |
| J | Escritura del primer borrador de la memoria del proyecto | 1 |
| K | Implantación del algoritmo de clasificación en la aplicación | 2 |
| L | Pruebas y ajustes de la aplicación | 2 |
| M | Actualización y finalización de la memoria | 1 |

*Tabla 4. Planificación temporal del proyecto*

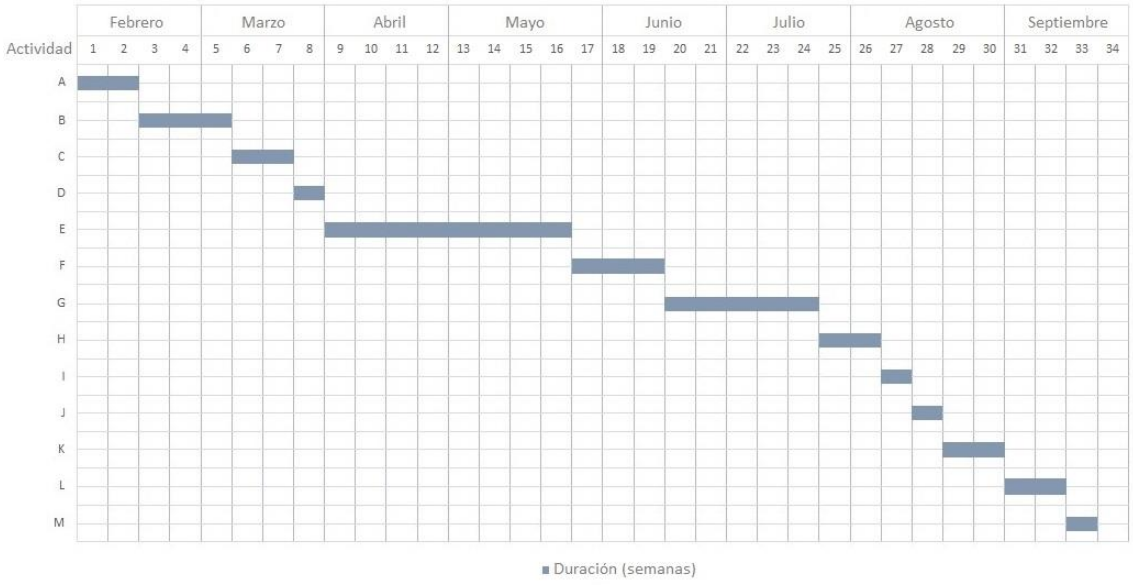


*Figura 4.1. Diagrama de Gantt asociado al proyecto*

Para una mejor comprensión de la planificación temporal del proyecto se va a hacer una breve explicación de las fases en las que se ha desarrollado.

- **Fase 1**: durante la primera fase se estudiaron las tecnologías necesarias para realizar una aplicación Android que hace uso de los sensores presentes en el teléfono. También se estudiaron las formas de almacenamiento de datos en Android seleccionando la más adecuada. Se realizó una primera versión de prueba de los sensores y almacenamiento de datos en ficheros. Se recogieron datos de diferentes barreras urbanas para comenzar la siguiente fase.

- **Fase 2**: en esta fase se trabajó con Matlab los datos obtenidos en la fase anterior viendo cuales eran las posibles características que se podrían extraer y haciendo una primera versión de un código que fuera capaz de extraerlas, teniendo en cuenta todas las variables y problemas que se iban presentando.
- **Fase 3**: la primera parte de esta fase se centró en el traslado del algoritmo que extraía todas las características de los datos recogidos por los sensores programado en Matlab al lenguaje Java. Además de ajustarlo para que funcionara en tiempo real. Entre la primera y segunda parte de esta fase se dedicó un breve espacio de tiempo para escribir un primer borrador de la presente memoria. La segunda parte de la fase se centró en la obtención de un algoritmo que fuera capaz de clasificar en tiempo real las nuevas muestras que la aplicación iba recogiendo, además del ajuste de este algoritmo para conseguir los mejores resultados posibles. Una vez ese trabajo estaba finalizado se terminó la redacción de esta memoria.

## 4.2 IMPACTO SOCIOECONÓMICO

A la hora de evaluar el impacto que el producto que estamos desarrollando puede tener en nuestra sociedad debemos tener en cuenta tanto la repercusión como los beneficios que va a generar.

En primer lugar, se va a evaluar cómo afecta a las Administraciones Públicas centrándonos en tres aspectos: el retorno de las inversiones, el beneficio para la sociedad y la generación de puestos de trabajo.

Después se evaluará cómo afecta nuestra aplicación desglosándola en los distintos tipos de impacto. También se tendrá en cuenta el impacto medioambiental.

Por último, se planteará un plan de explotación del trabajo junto con un análisis global del impacto socioeconómico.

### 4.2.1 Impacto económico y social

Para valorar cómo va a afectar nuestra aplicación a la sociedad vamos a valorar cómo afecta a las Administraciones Públicas.

Debemos tener en cuenta que parece que la crisis económica finamente está terminando por lo que el Estado debería tener más dinero para invertir en diversas tecnologías. La inversión realizada es de esperar que generen un beneficio para la sociedad.

Según se ha comentado en capítulos anteriores la aplicación desarrollada podría utilizarse dentro de las Administraciones para utilizar las partidas destinadas a la mejora de las infraestructuras de la vía pública.

De esta forma quedaría justificado el uso de estos presupuestos y se generaría una mejora para la sociedad. Además, también se estarían generando puestos de trabajo dado que para realizar estas mejoras se necesita contratar personal cualificado.

Respecto al beneficio que genera la inversión realizada, en este caso no se busca tanto un beneficio económico sino social. Dado que los colectivos a los que afectan las barreras urbanas que se encuentran dentro de una ciudad verán mejorada su calidad de vida si pueden moverse con mayor facilidad por todo el territorio.

### 4.2.2 Tipos de impacto

Para poder evaluar el impacto socioeconómico global generado por nuestra aplicación debemos valorar los diferentes tipos de impacto.

#### *4.2.2.1 Impacto directo*

El impacto directo está relacionado con la generación de producción y empleos dentro de los sectores que puedan ser receptores de las inversiones realizadas. Igualmente está relacionado con los gastos que atrae el despliegue de una nueva infraestructura.

En este aspecto el uso de nuestra aplicación por parte de las Administraciones como se ha mencionado anteriormente estaría generando puestos de trabajo para realizar las posibles mejoras de la vía pública de cara a hacer más accesibles los lugares que cuenten con barreras urbanas que limitan la accesibilidad de todos los usuarios.

#### *4.2.2.2 Impacto indirecto*

El impacto indirecto se centra en la producción y empleos generados por el beneficio indirecto de las inversiones realizadas.

De esta forma podríamos considerar un impacto indirecto en todos los lugares que se encuentren cerca de los lugares que las Administraciones consideren mejorar la accesibilidad gracias a los datos recogidos con nuestra aplicación.

#### *4.2.2.3 Impacto inducido*

El impacto inducido está relacionado con todo beneficio que se obtenga de forma directa o indirecta de las inversiones realizadas.

Como se ha dicho en el punto anterior el permitir una mayor accesibilidad a lugares que anteriormente estaban más restringidos puede generar una nueva afluencia de público a estos lugares con mejoras. De esta forma se puede fomentar el consumo en estos entornos lo que puede derivar en la generación de nuevos puestos de trabajo.

### 4.2.3 Impacto medioambiental

El impacto medioambiental es el efecto que produce la actividad humana sobre el medio ambiente. [18]

En este sentido el uso de una aplicación móvil con unas características como la nuestra no ofrece en principio ninguna posibilidad de afectar al medio ambiente. Bajo el uso propuesto para las Administraciones Públicas lo que hace es beneficiar los posibles efectos socioculturales ya que permite mejorar las relaciones sociales.

Aunque hay que tener en cuenta que el uso final que los usuarios puedan dar a la aplicación en caso de no ser correcto pueda terminar siendo perjudicial para el medio ambiente.

### 4.2.4 Plan de explotación

Con el plan de explotación de la aplicación lo que se pretende es que el proyecto genere los mayores beneficios posibles.

Debemos considerar un aspecto fundamental de cara a buscar beneficios: el precio de la aplicación.

El objetivo final de todas las aplicaciones disponibles en el mercado es llegar al mayor número de usuarios posibles, esto hace que la decisión de fijar un precio por la descarga de la aplicación sea un factor complicado.

En el caso de que nuestro objetivo sea sólo facilitar la movilidad a los usuarios que utilicen la aplicación deberíamos valorar que la aplicación fuera gratuita obteniendo nuestro beneficio en los datos que recojamos de los usuarios para poder mejorar nuestros servicios.

Por el contrario, si también queremos obtener beneficios económicos de la distribución de nuestra aplicación fijaremos un pequeño precio dado que por las características de nuestra aplicación si el precio fuera elevado el número de usuarios potenciales se vería reducido.

#### *4.2.4.1 Beneficios del proyecto*

- Las prestaciones ofrecidas por nuestra aplicación centrándonos uso final propuesto suponen una novedad, porque como se ha visto no existen a penas aplicaciones que se centren en este objetivo. Por lo tanto, al introducir nuevas alternativas en el mercado los usuarios se van a ver beneficiados al tener la posibilidad de elegir la aplicación que más les guste y mejor se adapte a lo que están buscando.
- Como se ha analizado en las secciones anteriores el uso de la aplicación puede generar nuevos empleos y no tiene riesgo para la seguridad medioambiental.
- El uso de nuestra aplicación puede extenderse dentro del ámbito de las Administraciones Públicas a cualquier territorio.
- La aplicación puede generar mejoras en la calidad de vida de las personas dado que van a tener disponible la localización de las barreras urbanas de las ciudades. Además, si contamos con las mejoras que se van a derivar de su uso por las Administraciones se van a beneficiar de las mejores de las infraestructuras de la vía pública.
- También hay que destacar que la aplicación para la detección de barreras urbanas va a ocupar una posición competitiva en el mercado dado que no tiene grandes rivales potenciales al existir escasas aplicaciones con la misma funcionalidad y que ofrezcan las mismas prestaciones.

#### *4.2.4.2 Plan de difusión*

- En el caso de que finalmente la aplicación no fuera gratuita se podría ofrecer un descuento en su precio de venta con su lanzamiento con el fin de captar más usuarios.
- Para cumplir con el objetivo de su uso dentro de las Administraciones Públicas se elaborarán una serie de jornadas de demostración para mostrar las ventajas que se pueden obtener de su uso. De estas jornadas se derivarían una serie de acuerdos para su distribución.
- Se contará con una difusión web para publicitar la aplicación.
- Se realizarán demostraciones puntuales a los usuarios potenciales.

### 4.2.5 Conclusiones del impacto socioeconómico

Tras analizar todos los aspectos que afectan al impacto socioeconómico podemos concluir que dados los beneficios que genera para la sociedad en todos los aspectos el desarrollo y uso de la aplicación para la detección de barreras urbanas tiene un alto y positivo impacto.

# 5 DISEÑO DE LA APLICACIÓN

Tras el estudio del marco en el que se va a desarrollar la aplicación mediante los puntos que componen los capítulos anteriores se han seleccionado las distintas herramientas que se van a utilizar en la aplicación. A lo largo de este capítulo se describirán los requisitos necesarios para el correcto funcionamiento de la aplicación y se explicará su arquitectura.

## 5.1 REQUISITOS DE LA APLICACIÓN

A continuación, se describirán los requisitos para la realización de la aplicación. La información correspondiente a cada tipo de requisito se recoge en una serie de tablas que contienen los siguientes campos:

- **Identificador**: se trata de una clave inequívoca que identifica cada requisito. Para cada uno de los requisitos se utilizará la siguiente nomenclatura:
  - RU-Cxx: Requisitos de usuario del tipo capacidad.
  - RU-Rxx: Requisitos de usuario del tipo restricción.
  - RS-Fxx: Requisitos de software del tipo funcional.
  - RS-NFxx: Requisitos de software del tipo no funcional.
- **Título**: se corresponde con el nombre del requisito y será complementado con la descripción de dicho requisito.
- **Descripción**: es una breve y concisa explicación del requisito que se está definiendo.
- **Prioridad**: describe el nivel de prioridad a la hora de realizar el requisito.
- **Necesidad**: define la importancia del requisito.

### 5.1.1 Requisitos de usuario

Los requisitos de usuario permiten conocer cuáles son las acciones que se pretende que cumpla la aplicación.

#### *5.1.1.1 Requisitos de capacidad*

Describen las acciones que el usuario puede hacer sobre el sistema.

| **RU-C01** | | | |
|---|---|---|---|
| Titulo | Funcionalidad principal | | |
| Descripción | La aplicación será capaz de reconocer las diferentes barreras urbanas mientras que se encuentre en funcionamiento y el teléfono se lleve en la mano. | | |
| Prioridad | ☑ Alta | ☐ Media | ☐ Baja |
| Necesidad | ☑ Esencial | ☐ Opcional | |

*Tabla 5. Requisito RU-C01*

| RU-C02 | | | |
|---|---|---|---|
| Titulo | Barrera actual | | |
| Descripción | La aplicación será capaz de indicar en todo momento en el que se estén reconociendo barreras el tipo de barrera reconocida. | | |
| Prioridad | ☑ Alta | ☐ Media | ☐ Baja |
| Necesidad | ☑ Esencial | ☐ Opcional | |

*Tabla 6. Requisito RU-C02*

| RU-C03 | | | |
|---|---|---|---|
| Titulo | Reconocimiento activo | | |
| Descripción | La aplicación indicará cuando el reconocimiento de barreras se active y se desactive. | | |
| Prioridad | ☑ Alta | ☐ Media | ☐ Baja |
| Necesidad | ☑ Esencial | ☐ Opcional | |

*Tabla 7. Requisito RU-C03*

| RU-C04 | | | |
|---|---|---|---|
| Titulo | Controles de reconocimiento | | |
| Descripción | La aplicación dispondrá de unos botones que permitan activar y desactivar el reconocimiento de barreras urbanas. | | |
| Prioridad | ☑ Alta | ☐ Media | ☐ Baja |
| Necesidad | ☑ Esencial | ☐ Opcional | |

*Tabla 8. Requisito RU-C04*

#### *5.1.1.2 Requisitos de restricción*

Indican las restricciones que tiene el sistema.

| RU-R01 | | | |
|---|---|---|---|
| Titulo | Barreras urbanas reconocidas | | |
| Descripción | La aplicación reconocerá exclusivamente las siguientes barreras urbanas: andar, subir escalera, bajar escalera, subir cuesta y bajar cuesta. | | |
| Prioridad | ☑ Alta | ☐ Media | ☐ Baja |
| Necesidad | ☑ Esencial | ☐ Opcional | |

*Tabla 9. Requisito RU-R01*

| RU-R02 | | | |
|---|---|---|---|
| Titulo | Tipos de estado | | |
| Descripción | La aplicación mostrará el estado del reconocimiento únicamente una vez al iniciar (reconocimiento iniciado) y al finalizar dicho reconocimiento (reconocimiento finalizado). | | |
| Prioridad | ☐ Alta | ☑ Media | ☐ Baja |
| Necesidad | ☐ Esencial | ☑ Opcional | |

*Tabla 10. Requisito RU-R02*

| RU-R03 | | | |
|---|---|---|---|
| Titulo | Tiempo de ejecución | | |
| Descripción | La aplicación realizará las acciones necesarias para la detección de barreras en un tiempo suficientemente pequeño como para que la barrera indicada al usuario en todo momento sea en tiempo real. | | |
| Prioridad | ☑ Alta | ☐ Media | ☐ Baja |
| Necesidad | ☑ Esencial | ☐ Opcional | |

*Tabla 11. Requisito RU-R03*

| RU-R04 | | | |
|---|---|---|---|
| Titulo | Versión del sistema | | |
| Descripción | La aplicación está realizada para la versión de Android 4.4.2 pudiendo ser utilizada en versiones posteriores. | | |
| Prioridad | ☑ Alta | ☐ Media | ☐ Baja |
| Necesidad | ☑ Esencial | ☐ Opcional | |

*Tabla 12. Requisito RU-R04*

### 5.1.2 Requisitos de software

Los requisitos de software se basan en los requisitos de usuarios y definen de forma técnica el comportamiento de la aplicación.

#### *5.1.2.1 Requisitos funcionales*

Incluyen las acciones que la aplicación va a permitir realizar a los usuarios.

| RS-F01 | | | |
|---|---|---|---|
| Titulo | Clasificador | | |
| Descripción | La aplicación contará con un método de clasificación que permitirá reconocer las diferentes barreras urbanas. | | |
| Prioridad | ☑ Alta | ☐ Media | ☐ Baja |
| Necesidad | ☑ Esencial | ☐ Opcional | |

*Tabla 13. Requisito RS-F01*

| RS-F02 | | | |
|---|---|---|---|
| Titulo | Acelerómetro | | |
| Descripción | La aplicación utilizará el acelerómetro del dispositivo para reconocer las diferentes barreras urbanas. | | |
| Prioridad | ☑ Alta | ☐ Media | ☐ Baja |
| Necesidad | ☑ Esencial | ☐ Opcional | |

*Tabla 14. Requisito RS-F02*

| RS-F03 | | | |
|---|---|---|---|
| Titulo | Sensor de gravedad | | |
| Descripción | La aplicación utilizará el sensor de gravedad para reconocer las barreras urbanas. | | |
| Prioridad | ☑ Alta | ☐ Media | ☐ Baja |
| Necesidad | ☑ Esencial | ☐ Opcional | |

*Tabla 15. Requisito RS-F03*

| RS-F04 | | | |
|---|---|---|---|
| Titulo | Almacenamiento de datos | | |
| Descripción | La aplicación contará con un sistema de almacenamiento dinámico de los datos que necesario para la detección de las barreras urbanas. | | |
| Prioridad | ☑ Alta | ☐ Media | ☐ Baja |
| Necesidad | ☑ Esencial | ☐ Opcional | |

*Tabla 16. Requisito RS-F04*

| RS-F05 | | | |
|---|---|---|---|
| Titulo | Estado del reconocimiento | | |
| Descripción | La aplicación mostrará un mensaje en pantalla cuando se inicie y se detenga el reconocimiento de barreras urbanas. | | |
| Prioridad | ☐ Alta | ☑ Media | ☐ Baja |
| Necesidad | ☐ Esencial | ☑ Opcional | |

*Tabla 17. Requisito RS-F05*

| RS-F06 | | | |
|---|---|---|---|
| Titulo | Botón para iniciar reconocimiento | | |
| Descripción | La aplicación dispondrá de un botón para iniciar y terminar el reconocimiento de barreras urbanas. | | |
| Prioridad | ☑ Alta | ☐ Media | ☐ Baja |
| Necesidad | ☑ Esencial | ☐ Opcional | |

*Tabla 18. Requisito RS-F06*

#### *5.1.2.2 Requisitos no funcionales*

Indican las condiciones especiales que debe cumplir la aplicación.

| RS-NF01 | | | |
|---|---|---|---|
| Titulo | Recogida de datos para desarrolladores | | |
| Descripción | La aplicación contará con la opción de almacenar datos para que los desarrolladores de la aplicación puedan mejorar la misma. | | |
| Prioridad | ☑ Alta | ☐ Media | ☐ Baja |
| Necesidad | ☑ Esencial | ☐ Opcional | |

*Tabla 19. Requisito RS-NF01*

| RS-NF02 | | | |
|---|---|---|---|
| Titulo | Comprobación de memoria externa | | |
| Descripción | La aplicación comprobará la disponibilidad de una memoria externa cuando sea necesario guardar datos en ella. | | |
| Prioridad | ☑ Alta | ☐ Media | ☐ Baja |
| Necesidad | ☑ Esencial | ☐ Opcional | |

*Tabla 20. Requisito RS-NF02*

### 5.1.3 Matriz de trazabilidad

Con los requisitos descritos anteriormente demandados por el usuario y los requisitos de software que se han decidido que van a desarrollarse en la aplicación se ha desarrollado una matriz de trazabilidad.

La Tabla 21 muestra cómo los requisitos de usuario están recogidos en al menos uno de los requisitos de software.

| RS\RU | RU-C01 | RU-C02 | RU-C03 | RU-C04 | RU-R01 | RU-R02 | RU-R03 | RU-R04 |
|---|---|---|---|---|---|---|---|---|
| RS-F01 | × | × | | | × | | × | |
| RS-F02 | × | × | | | × | | | |
| RS-F03 | × | × | | | | | | |
| RS-F04 | | | | | | | × | |
| RS-F05 | | | × | | | × | | |
| RS-F06 | | | | × | | | | |
| RS-NF01 | × | | | | | | | × |
| RS-NF02 | × | | | | | | | |

*Tabla 21. Matriz de trazabilidad*

## 5.2 ARQUITECTURA DE LA APLICACIÓN

Después de haber estudiado lo que los usuarios desean obtener de la aplicación se procederá al desarrollo de la misma.

El sistema desarrollado contará con una parte en la que se encuentre la interfaz donde interactuarán los usuarios y otra parte que contendrá la lógica y los datos de la aplicación. También contará con una parte dedicada a la comunicación entre la interfaz y la parte lógica.

Una vez estudiados los diferentes modelos utilizados en la ingeniería de software vemos que el que más se ajusta a nuestro sistema es el Modelo-Vista-Controlador (MVC). Este modelo consta de tres partes:

- **Modelo**: es la parte que trabaja con los datos de la aplicación. Se encarga de acceder a la información y actualizar su estado. Es el encargado de notificar a la vista los cambios que se producen en el sistema.
- **Vista**: es la parte en la que se muestran los datos por pantalla al usuario. Dado que las aplicaciones suelen tener más de una interfaz también se encarga de la navegación entre ellas. En la vista se trabaja con los datos, pero no se accede directamente a ellos, se solicitan al modelo y es éste el encargado de generar la salida que requiera nuestra aplicación.
- **Controlador**: contiene el código que responde a las acciones que se solicitan en la aplicación. Sirve de enlace entre los modelos y la vista, pero no se encarga ni de acceder a los datos ni de mostrar ninguna salida.

En la Figura 5.1 se puede ver un diagrama que representa el Modelo-Vista-Controlador.

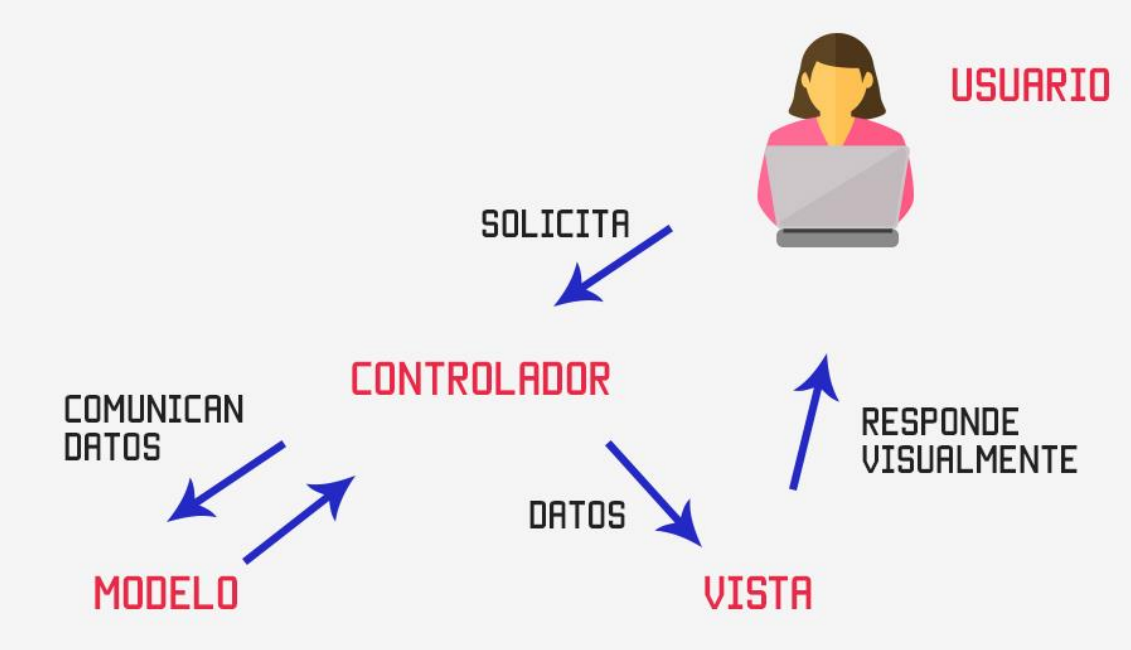


*Figura 5.1. Diagrama del sistema Modelo-Vista-Controlador. Figura extraída de Código facilito [19]*

Para entenderlo más fácilmente describiremos un ejemplo basado en nuestra aplicación:

- Cuando el usuario quiere empezar a hacer la detección de barreras urbanas lo hace mediante una acción en la pantalla de su smartphone.
- La vista identifica el evento ocurrido e informa al controlador.
- El controlador comunica los datos recibidos por la vista al modelo.
- El modelo ejecuta las acciones necesarias con los datos que ha recibido y envía una respuesta al controlador.
- La respuesta recibida por el controlador será trasladada a la vista en el formato adecuado para que sea legible por el usuario.
- La vista recibe la respuesta y se la muestra al usuario.

Para nuestra aplicación podemos identificar el MVC de la siguiente forma:

- Modelo: se centra en los ficheros que se almacenan con todos los datos recogidos de los sensores y en los métodos que permiten acceder a ellos.
- Vista: se basa en todos los ficheros que constituyen la interfaz gráfica de la aplicación.
- Controlador: se trata de la mayor parte del código Java que componen el sistema. Se compone de todos los métodos que manejan la información que tiene el sistema y los que permiten mostrar al usuario la parte de vista.

# 6 IMPLEMENTACIÓN DEL SISTEMA

En este capítulo se describirá el lenguaje de programación utilizado, así como el entorno de desarrollo utilizado. Posteriormente se pasará a detallar cómo se ha realizado la implementación de la aplicación

## 6.1 LENGUAJE DE PROGRAMACIÓN

Como se ha comentado en capítulos anteriores el sistema operativo elegido para el desarrollo de la aplicación es Android. Este sistema operativo utiliza como lenguaje de programación Java que es un lenguaje de alto nivel, a diferencia de los sistemas que desarrollaban aplicaciones inicialmente que utilizaban lenguajes de bajo nivel como C/C++, aunque las librerías de Android sí que mantienen este lenguaje.

Para crear aplicaciones Android consta de un SDK que permite desarrollar código para las aplicaciones, acceder al hardware del dispositivo o trabajar con Google Maps entre otras muchas cosas.

Además, dispone de un API muy completo para cada versión de la plataforma con información sobre todas las clases y demás atributos que tiene disponible cada nivel.

## 6.2 ENTORNO DE DESARROLLO

Como se ha comentado en el capítulo 2.1.1.1 actualmente Android dispone de un entorno de desarrollo propio llamado Android Studio que sustituye a Eclipse, utilizado anteriormente.

Android Studio está basado en IntelliJ IDEA. Es un entorno en el que se pueden desarrollar aplicaciones para todos los tipos de dispositivos Android. Además, contiene herramientas para detectar problemas de compatibilidad entre versiones y rendimiento. Es compatible con C++ y NDK y facilita la integración de Google Cloud Messaging y App Engine. Otra de sus características es que estructura de forma automática la configuración del proyecto.

## 6.3 DIAGRAMA DE CLASES

A continuación, se pasarán a detallar todas las clases que componen la aplicación con sus atributos y métodos.

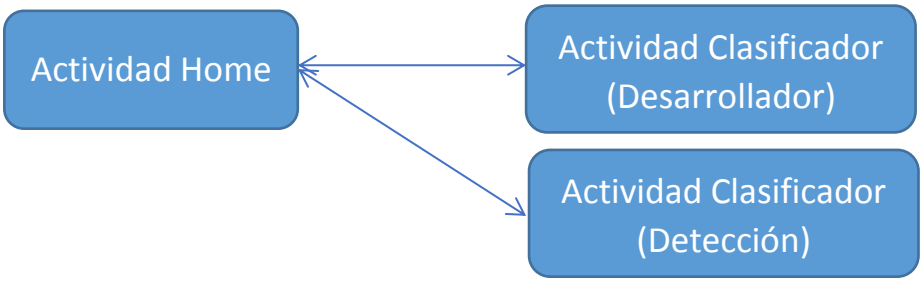


*Figura 6.1. Esquema de la navegación entre actividades*

Como se puede observar en la Figura 6.1 la aplicación consta de dos pantallas. La pantalla inicial cuenta con dos botones que permiten al usuario elegir la opción con la que realizar la detección

de barreras urbanas. Ambos botones llevan a la misma actividad (como se puede ver en la Figura 6.1) aunque la apariencia de la pantalla y las funciones disponibles serán diferentes en función de la opción elegida. El primero de los botones permite acceder a la opción de desarrollador que permite al usuario guardar datos en memoria con la finalidad de que sean utilizados por los desarrolladores a la vez que se está realizando la detección de las barreras. El segundo de los botones únicamente permite realizar la detección de barreras.

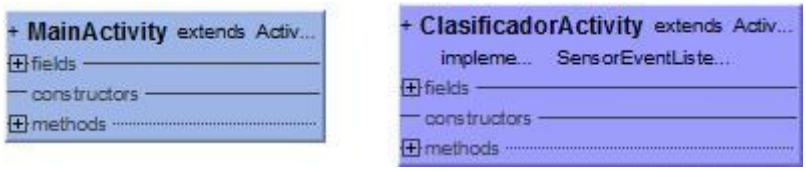


*Figura 6.2. Diagrama de clases de la aplicación*

En la Figura 6.2 se pueden observar las clases que componen la aplicación, aunque este diagrama no incluye las clases propias de Android como los ficheros XML correspondientes al desarrollo de interfaces.

Con el fin de comprender mejor el funcionamiento de la aplicación a continuación, se detallarán los aspectos fundamentales de las clases que componen la aplicación.

### 6.3.1 Activity Main

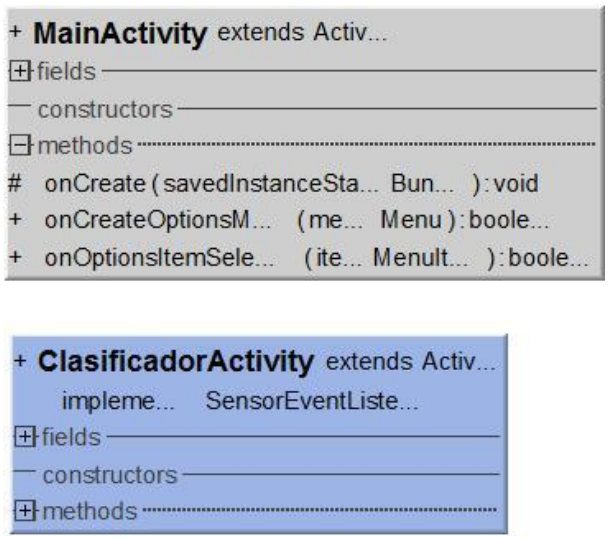


*Figura 6.3. Diagrama de clases. Detalle de Main Activity*

Como se ha dicho la Main Activity se corresponde con la pantalla inicial de la aplicación. Para que el usuario se pueda mover entre esta pantalla y las de la detección se dispone de dos botones que tienen unos escuchadores y que cuando son pulsados pasan la información sobre qué tipo de detección quiere realizar el usuario a la siguiente actividad.

Dentro de los métodos que contiene esta actividad podemos ver onCreate que está presente en todas las actividades Android ya que se utiliza para inicializar los componentes fundamentales de la actividad y definir el diseño de la interfaz de usuario.

Los métodos onCreateOptionsMenu y onOptionsItemSelected, también van a estar presentes a la Clasificador Activity, lo que permiten es generar el menú que tiene la aplicación. En este caso el menú de ambas actividades lo que contiene es una ayuda sobre la pantalla en la que se

encuentra el usuario. Cuando se pulsa aparece un diálogo con las instrucciones para utilizar dicha pantalla.

### 6.3.2 Activity Clasificador

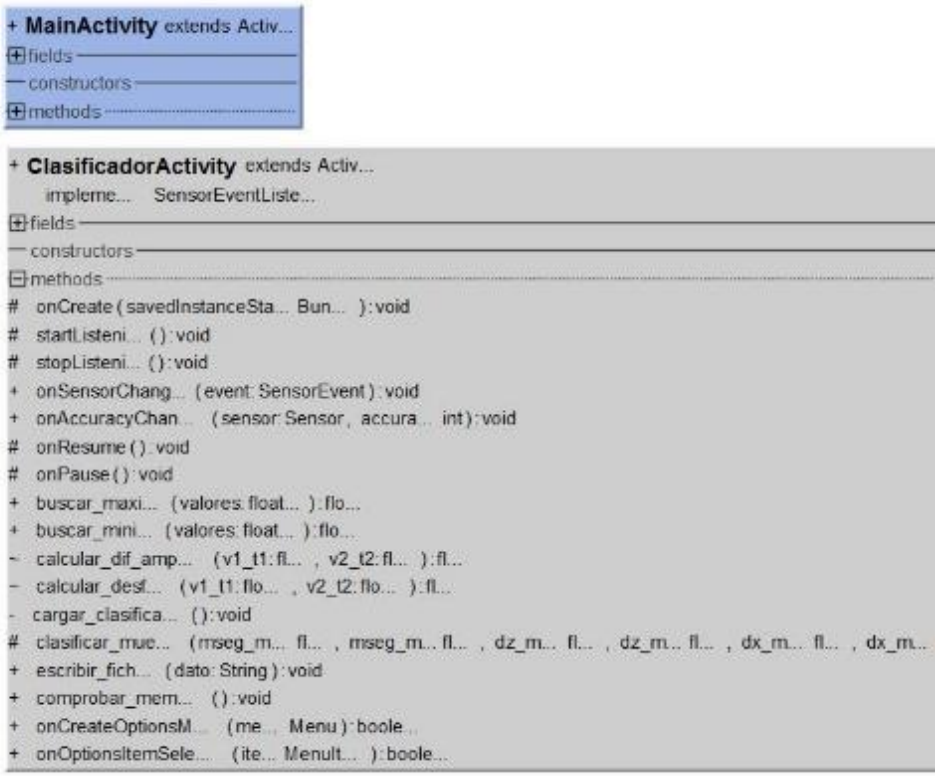


*Figura 6.4. Diagrama de clases. Detalle de Clasificador Activity*

Como se observa en la Figura 6.4 esta clase es la que compone el grueso de la aplicación. A continuación, se detallarán los métodos más importantes de esta actividad y cómo es su funcionamiento.

#### *6.3.2.1 Parte común Activity Clasificador*

En primer lugar, vamos a describir la parte común que tiene esta actividad independientemente de la opción que el usuario haya elegido como tipo de detección en la pantalla inicial.

Para poder realizar la detección de las barreras urbanas es necesario utilizar un modelo creado con Weka para poder clasificar cada nueva muestra con los datos recogidos por los sensores. Este modelo está incluido dentro de la aplicación, pero para poder utilizarlo lo primero que se hace es cargarlo (dentro del método onCreate). Una vez que está cargado se muestra un aviso por medio de un Toast al usuario.

Cuando el usuario acciona la detección de barreras urbanas mediante un botón disponible en esta pantalla se registran unos escuchadores que permiten acceder a los datos de los sensores que se van a utilizar (a su vez se muestra otro mensaje al usuario indicando que se ha iniciado el reconocimiento).

El método onSensorChanged es llamado cada que vez que uno de los sensores utilizados tiene un nuevo valor. Dentro de este método se obtienen todas las variables que se van a utilizar para clasificar qué tipo de barrera estamos detectando.

Para ello se extraen los valores de los tres ejes del acelerómetro y del sensor de gravedad. Como ya se dijo en el capítulo 2.1.2.2 el acelerómetro obtiene unos valores distorsionados por la gravedad. Por lo que esto es compensado restándole los datos que proporciona el sensor de gravedad, obteniendo así unos valores de aceleración reales. Además, también se recoge el instante en el que llegan estos nuevos datos. Después se almacenan varios de estos datos según se van recibiendo para poder buscar máximos y mínimos y calcular otros parámetros que se utilizarán para la clasificación de las muestras.

Una vez que se obtienen dos valores de máximos y mínimos se calculan:

- Diferencia de amplitud entre máximos.
- Diferencia de amplitud entre mínimos.
- Diferencia de amplitud entre máximo y mínimo.
- Desfase entre máximos.
- Desfase entre mínimos.
- Desfase entre máximo y mínimo.

Todo ello para los tres ejes en los que los sensores están recibiendo los datos. Para ello se utilizan los métodos buscar_maximo, buscar_mínimo, calcular_dif_amplitud y calcular_desfase.

Todos estos datos se pasan al método clasificar_muestra que utilizando el clasificador generado con Weka identifica y devuelve a qué tipo de barrera nos estamos enfrentando. Este valor es el que será mostrado al usuario en pantalla hasta que decida detener la detección de barreras urbanas, en este caso le aparecerá otro mensaje en pantalla para informarle que el reconocimiento de las barreras urbanas ha finalizado.

#### *6.3.2.2 Activity Clasificador. Opción desarrollador.*

Cuando el usuario selecciona la opción de desarrollador lo que hace es permitir que se almacenen datos en la memoria externa del dispositivo (en el caso de estar disponible).

Para poder almacenar los datos, en primer lugar, el usuario tiene disponible una serie de opciones con los tipos de barreras urbanas que la aplicación es capaz de detectar y que deberá seleccionar a la que se esté enfrentando en ese momento. Al cargar esta actividad se comprueba que la memoria externa esté disponible y permita la escritura en ella, esto se realiza con el método comprobar_memoria.

Una vez que se ha predicho por parte de la aplicación a la barrera urbana a la que se está enfrentando el usuario se procede a escribir en un fichero dentro de la memoria todos los datos que pueden ser utilizados por los desarrolladores. En este caso son: todas las variables que se utilizan para poder detectar las barreras urbanas junto con la predicción que ha hecho la aplicación y el dato que el usuario ha proporcionado sobre el tipo de barrera urbana que se debería estar detectando.

### 6.3.3 Presentación de la aplicación

En esta sección se presentarán todas las ventanas que pueden aparecer en el uso de la aplicación y que han sido mencionadas en la sección anterior con la descripción de las actividades que componen la aplicación.

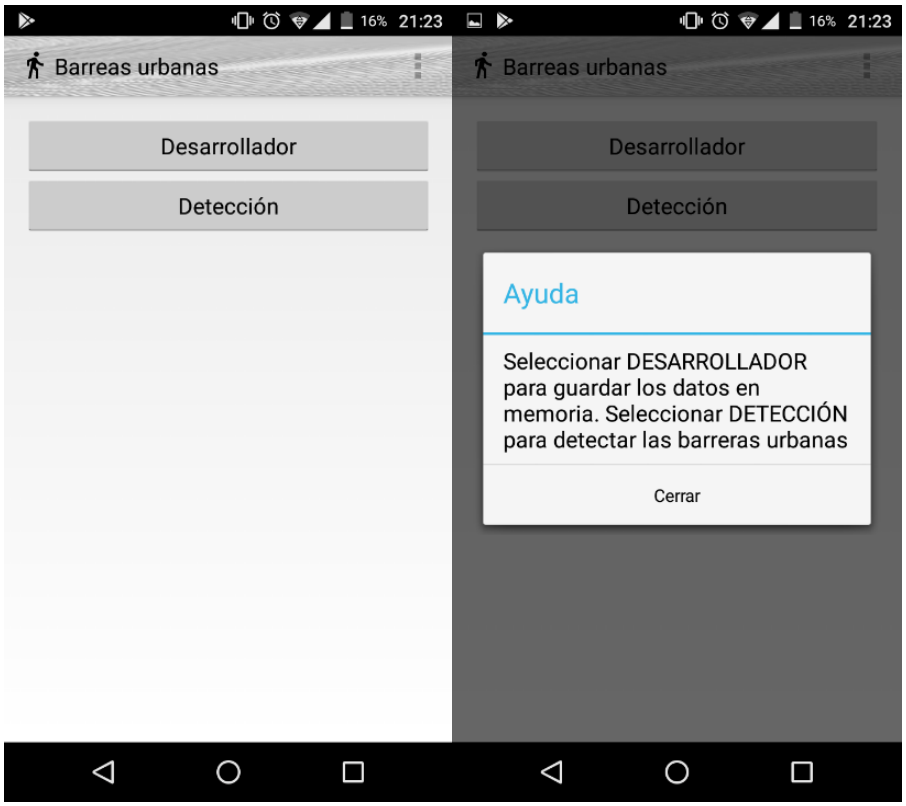


*Figura 6.5. Pantalla inicial y su ayuda*

En la Figura 6.5 se muestra la pantalla inicial junto con el mensaje de ayuda correspondiente a esta pantalla.

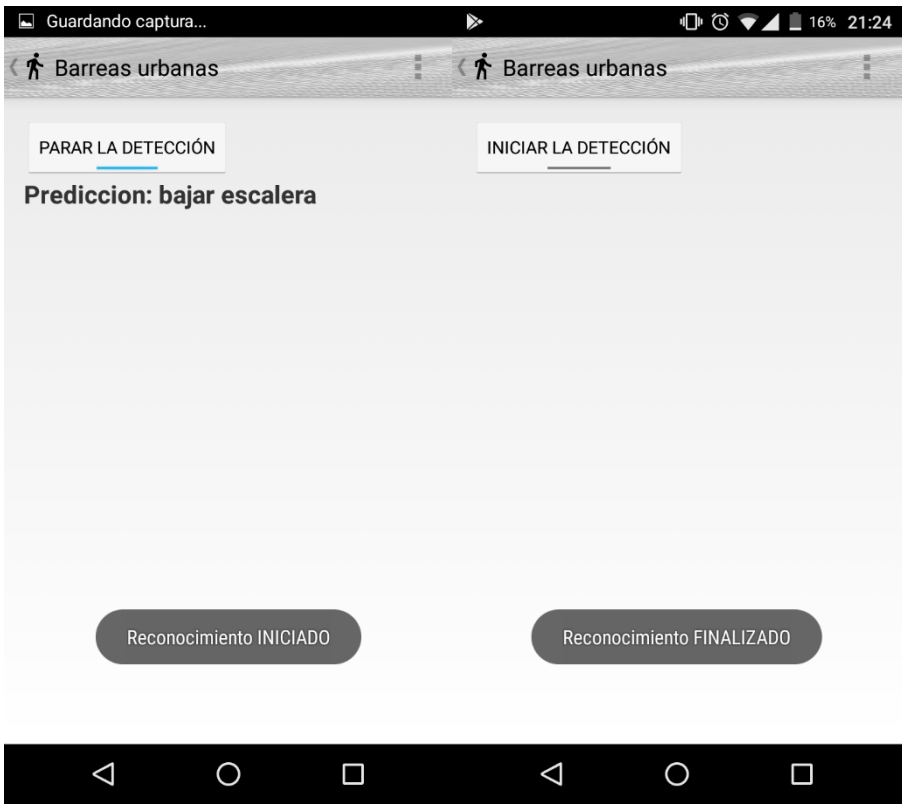


*Figura 6.6. Pantallas opción Detección*

Como se dijo en la sección anterior en la Figura 6.6 se puede ver cómo la aplicación muestra un mensaje para indicar cuando se inicia y finaliza el reconocimiento de barreras urbanas. Para realizar esta acción se utiliza el mismo botón que varía su texto en cuando el usuario lo pulsa.

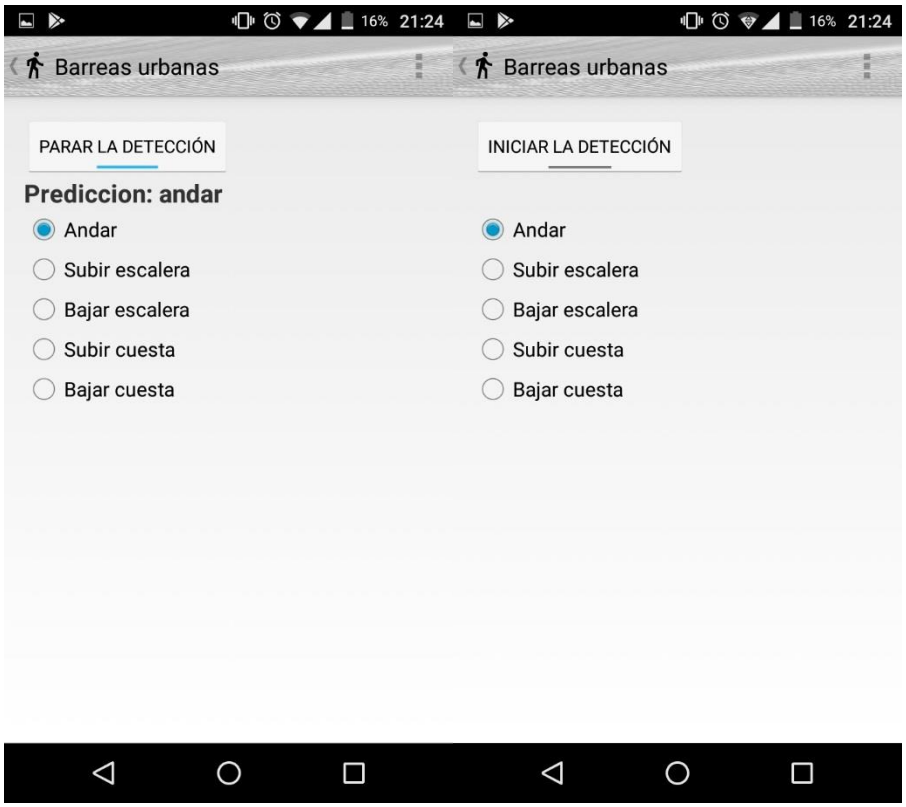


*Figura 6.7. Detalle de la opción Desarrollador*

La opción Desarrollador incluye la opción de guardar la actividad que se está desarrollando en el momento. Si se cambia de opción durante la detección se registra automáticamente y se comienza a guardar en el fichero la nueva opción seleccionada por el usuario.

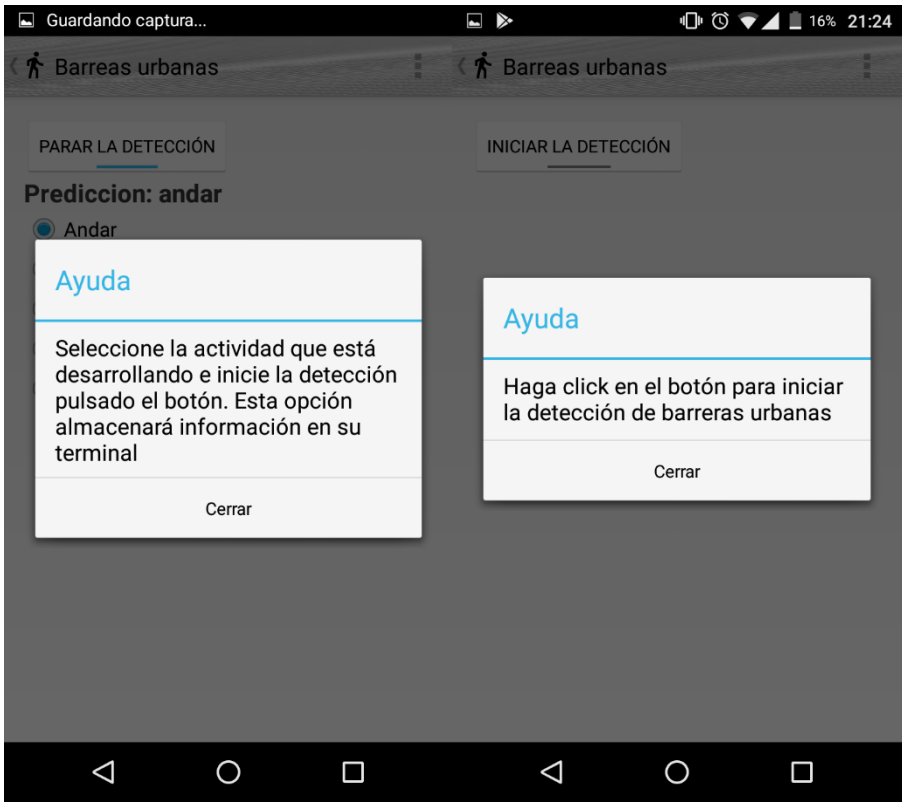


*Figura 6.8. Mensajes de ayuda de la opción Desarrollador y Detección*

Al igual que la pantalla inicial como se observa en la Figura 6.8 las opciones que permiten la detección de barreras urbanas tienen sus propios mensajes de ayuda.

### 6.3.4 Decisiones de diseño

A la hora de diseñar la aplicación final se plantean algunas cuestiones que no habían aparecido anteriormente. Estas decisiones son tomadas por el equipo desarrollador de la aplicación y serán explicadas en esta sección.

#### *6.3.4.1 Interfaz*

Al desarrollar una interfaz para cualquier dispositivo se tiene muy en cuenta el número de ventanas que utiliza y las opciones que tiene disponible, a la vez que hay que cuidar que su uso sea sencillo para los usuarios.

La primera versión desarrollada para la prueba de la funcionalidad principal de la aplicación se basaba en una sola ventana que permitía iniciar y detener la detección de barreras urbanas. Dado que no todos los usuarios van a querer almacenar datos en la memoria de sus teléfonos surgió la necesidad de dar la opción de sólo realizar la detección de barreras urbanas, de ahí que la versión final de la aplicación cuente con una primera pantalla que permita seleccionar entre las dos opciones.

La interfaz de detección cuenta únicamente con un botón de tipo toggle. Cuando se accede a la pantalla de detección en cualquiera de las opciones el botón muestra el texto “iniciar la detección” y al pulsarlo cambia su texto a “parar la detección”, alternando entre los dos mensajes según se va pulsando en el botón. Igualmente, la aplicación podría haber funcionado

utilizando dos botones simples (como en la pantalla inicial) uno que permita iniciar y otro para detener la detección.

Otra de las decisiones de diseño es la cantidad de información que el usuario va a recibir en la pantalla de su terminal referente a los datos utilizados para detectar las barreras urbanas. En un primer momento se optó por la opción de mostrar los valores obtenidos por el acelerómetro y su diferencia al eliminar la componente de la gravedad. Finalmente, esta opción quedó descartada mostrando únicamente la barrera urbana detectada, ya que el usuario no necesita ver estos otros datos que únicamente podrían confundirle.

Respecto a cómo se muestran los datos en la pantalla, como veremos en las secciones posteriores, los sensores recogen datos a su máxima velocidad. Esto hace que si mostráramos los datos al usuario a esta velocidad no fuera capaz de ver nada. Por lo tanto, se decidió que los datos se actualicen una vez por segundo en la pantalla del teléfono. De esta forma el usuario sigue percibiendo que varían de manera lo suficientemente rápida como para considerar que se está realizando un reconocimiento en tiempo real y es a la vez capaz de ver en todo momento los nuevos datos que aparezcan.

#### *6.3.4.2 Almacenamiento de datos*

Android dispone de tres opciones principales para el almacenamiento de datos: preferencias, ficheros y bases de datos. Generalmente para guardar grandes cantidades de datos se utilizan las bases de datos. En este caso el acceso a los datos está restringido a la propia aplicación, como veremos posteriormente con nuestra aplicación es necesario poder acceder a estos datos para generar el modelo de clasificador por lo que no es útil, al igual que las preferencias que sólo permiten almacenar pares clave/valor.

Por ello, la opción elegida para almacenar todos los datos necesarios son los ficheros. Con los ficheros disponemos de la opción de guardarlos en memoria interna o externa, como ya se ha dicho estos ficheros van a ser necesarios posteriormente por lo que se guardan en memoria externa cuando está disponible. No es posible guardarlos en memoria interna dado que al igual que las bases de datos sólo son accesibles para la aplicación y quedan ocultos para el resto.

#### *6.3.4.3 Acelerómetro*

El acelerómetro es el eje principal de la aplicación dado que gracias a él se obtienen todos los datos necesarios para permitir la clasificación y detección de barreras urbanas.

Los sensores en Android se manejan a partir de un Manager que captura los sensores y un Listener que recoge los nuevos valores obtenidos por los sensores.

Dentro de las posibilidades que tenemos para configurar dentro del acelerómetro es la velocidad con la que recoge los datos. Para la aplicación que estamos desarrollando es importante que los datos se registren con la mayor velocidad posible, por ello al registrar el escuchador para los sensores de la aplicación se utiliza la opción SENSOR_DELAY_FASTEST.

#### *6.3.4.4 Sensor gravity*

La configuración del sensor de gravedad utilizada es la misma que la del acelerómetro para que recoja los datos a la mayor velocidad posible. El sensor gravity estima la gravedad utilizando los datos del acelerómetro, la brújula y el giroscopio por lo que tiene una pequeña latencia desde que el acelerómetro empieza a entregar datos hasta que lo hace él. Este aspecto fue detectado

al comenzar a guardar datos en los ficheros y ha sido necesario corregirlo dentro del código para que todos los ficheros que se guarden contengan los datos de aceleración corregidos correctamente con datos del sensor de gravedad.

#### *6.3.4.5 Búsqueda de máximos y mínimos*

El cálculo de máximos y mínimos dentro de la aplicación para poder diferenciar las barreras urbanas es otra de las partes que los desarrolladores pueden diseñar. En este caso se ha realizado un estudio previo con Matlab para determinar cuál es el procedimiento que mejor se adapta a los datos que recogemos. Finalmente, la búsqueda se realiza con nueve muestras de aceleración corregidas para el eje z. Se ha escogido este eje porque es el que recoge los datos más precisos para poder realizar la detección ya que al subir una escalera, por ejemplo, va a ser el que varíe sus valores dependiendo de la variación de la gravedad al subir cada escalón. Por el contrario, los datos que recogen los ejes x e y representan en su mayor parte movimientos relativos de la posición del teléfono.

Con estos nueve datos se compara que el dato central sea superior que el resto en el caso de buscar máximos o inferior en el caso de que sea un mínimo.

#### *6.3.4.6 Clasificación con Weka*

Para decidir qué tipo de clasificación queríamos implementar se hizo un pequeño estudio previo con Matlab gracias a la herramienta Classification Learner que contiene. Utilizando los ficheros almacenados se probó a realizar la clasificación de un conjunto de muestras etiquetadas con árboles de decisión complejos, KNN o SVM. Obteniendo unos resultados similares para los árboles de decisión y SVM.

Dada la imposibilidad de trasladar los resultados que genera Matlab a la aplicación Android se decidió utilizar Weka. La ventaja de utilizar Weka como se comentó en la sección 2.1.3 es que dispone de una versión adaptada para Android. Para la clasificación con Weka se ha seleccionado el árbol J48 que implementa dicho algoritmo, éste es una implementación de código abierto en lenguaje de programación Java del algoritmo C4.5 [20].

Con los ficheros de datos obtenidos por la aplicación se ha conseguido generar un modelo con una probabilidad de muestras clasificadas correctamente de 79.9422%.

| Barrera | Andar | Subir cuesta | Bajar cuesta | Subir escalera | Bajar escalera |
|---|---|---|---|---|---|
| Andar | 1663 | 3 | 20 | 65 | 125 |
| Subir cuesta | 7 | 461 | 59 | 6 | 33 |
| Bajar cuesta | 37 | 64 | 739 | 18 | 119 |
| Subir escalera | 75 | 7 | 21 | 398 | 62 |
| Bajar escalera | 148 | 47 | 134 | 61 | 1167 |

*Tabla 22. Matriz de confusión del modelo Weka utilizado.*

En los siguientes capítulos se comentarán detalladamente los resultados obtenidos con este modelo.

Para la generación del modelo se ha utilizado una validación cruzada de 10 iteraciones para asegurar que la partición que se realiza entre los datos de entrenamiento y de test es independiente del modelo generado.

# 7 PRUEBAS Y EVALUACIÓN DE LA APLICACIÓN

En este capítulo se van a describir las pruebas realizadas para verificar el correcto funcionamiento del sistema. También se describirá el entorno de pruebas utilizado y se realizará un pequeño análisis de los resultados obtenidos.

## 7.1 DESCRIPCIÓN DEL CONJUNTO DE DATOS

Para la generación del modelo Weka utilizado en la aplicación para clasificar las nuevas muestras recibidas se ha utilizado un conjunto de entrenamiento con 5539 muestras etiquetadas. Estas muestras se obtuvieron para las diferentes barreras urbanas con una primera versión de la aplicación desarrollada, que como se ha comentado anteriormente únicamente permitía la captura y el almacenamiento de los datos.

| | Andar | Subir cuesta | Bajar cuesta | Subir escalera | Bajar escalera |
|---|---|---|---|---|---|
| Muestras | 1876 | 566 | 977 | 563 | 1557 |
| Porcentaje | 33.87% | 10.22% | 17.64% | 10.16% | 28.11% |

*Tabla 23. Número y porcentaje de muestras para cada barrera urbana.*

Como se puede observar en la Tabla 23 no todas las actividades tienen el mismo número de datos. Esto se debe a que encontrar barreras urbanas como andar es mucho más sencillo que las otras barreras.

Esto puede generar desequilibrios en el modelo generado que puede adaptarse mejor al tipo de datos con más muestras y que clasifique de peor manera las clases de las que dispone menos información.

## 7.2 ENTORNO DE PRUEBAS

Las pruebas se han realizado principalmente con un terminal, aunque para verificar el correcto funcionamiento e instalación de la aplicación en terminales con diferente versión de Android también se ha testado en un segundo terminal. Todas las pruebas para el almacenamiento de datos tanto para la generación del modelo del clasificador como para testar el funcionamiento se han realizado llevando el teléfono en la mano.

Los terminales utilizados son los siguientes:

- **Wiko Pulp** (Terminal principal)
  - Versión de Android: 5.1
  - Procesador: Octa-Core, 1.4 GHz, Cortex-A7
  - Memoria RAM: 2GB
- **BQ Aquaris U Lite** (Terminal secundario)
  - Versión de Android: 6.0.1
  - Procesador: Qualcomm® Snapdragon™ 425 Quad Core (MSM8917), 1.4 GHz
  - Memoria RAM: 2 GB

## 7.3 PRUEBAS REALIZADAS

A continuación, se detallarán todas las pruebas que se han realizado para verificar el correcto funcionamiento de todas las partes que componen la aplicación, así como las pruebas finales para testar la aplicación completa.

### 7.3.1 Prueba del acelerómetro y sensor de gravedad

Conectando el teléfono al ordenador mediante un cable USB se puede ver en Android Studio cierta información que nosotros seleccionemos utilizando Android Monitor y logcat. Este monitor permite ver los mensajes que muestra el sistema, pero también permite ver mensajes agregados por nosotros usando la clase Log. De esta forma si añadimos la clase Log, con los datos que recoge el acelerómetro y filtramos los mensajes en Android Monitor visualizaremos los datos del acelerómetro. Por otra parte, en la aplicación mostramos por pantalla los valores del acelerómetro en los tres ejes. Así comprobamos que estos valores son iguales y por lo tanto tenemos un correcto funcionamiento del sensor. Esta prueba se realiza igualmente para el sensor de gravedad.

### 7.3.2 Pruebas de almacenamiento de datos

Para comprobar el almacenamiento de datos inicialmente se tienen que dar permisos de escritura en la tarjeta de memoria externa del terminal en el caso de que se encuentre disponible.

Una vez que se han comprobado todos los permisos y la disponibilidad de la tarjeta de memoria es necesario pasar a escribir el fichero en el formato adecuado. Los ficheros se van a escribir en formato CSV por lo que se deben preparar los datos a escribir separándolos por comas. Además, para no sobrescribir los datos se deben añadir los datos nuevos al final del fichero (para ello se utiliza la opción append en lugar de write al escribir el fichero).

Con el objetivo de comprobar que los datos guardados en el fichero se almacenan en tiempo real se realizó una prueba similar a la utilizada para testar el correcto funcionamiento de los sensores. En primer lugar, se incluyó en los datos para almacenar en el fichero los valores registrados por el acelerómetro y el instante en el que se registraban dichos datos. Después, se mostraron en el ordenador los datos del acelerómetro mediante la clase Log. Para verificar que los datos se almacenan en tiempo real se comprobó que el dato almacenado en el fichero en un instante se correspondía con el mismo instante que mostraba la clase Log sobre el dato recogido por el acelerómetro.

### 7.3.3 Pruebas de validación del modelo Weka

Para conseguir generar un modelo correcto que clasifique por igual todos los tipos de datos es necesario tener un número uniforme y amplio de muestras de cada tipo. De esta forma el clasificador no se centrará en un solo tipo de muestra. Como ya hemos visto nuestro fichero utilizado para generar el modelo no cumple esta característica dado que tiene muchas más muestras de dos tipos de datos que del resto. Esto puede ser un problema para el resultado final ya que puede que no sea capaz de clasificar todas las muestras con la misma facilidad.

Al generar un modelo con el programa Weka obtenemos los siguientes datos que nos dan información sobre el mismo:

- Porcentaje de muestras clasificadas correctamente.
- Porcentaje de muestras clasificadas erróneamente.
- Estadísticas Kappa: mide lo que se ajusta la predicción a la clase real (1.0 significa ajuste total). [21]
- Error medio absoluto: es la media de la diferencia del valor que se ha obtenido en la clasificación y el valor real de la muestra.
- Raíz cuadrada del error medio: calcula la raíz cuadrada de la diferencia entre el valor obtenido en la clasificación y el valor real.
- Error absoluto relativo: es el cociente entre el error absoluto y el valor exacto.
- Raíz cuadrada del error absoluto relativo: es la raíz cuadrada del cociente entre el error absoluto.

| Resultado para validación cruzada de 10 iteraciones | | |
|---|---|---|
| Muestras correctamente clasificadas | 4428 | 79.95% |
| Muestras incorrectamente clasificadas | 1111 | 20.05% |
| Estadísticas Kappa | 0.7339 | |
| Media del error absoluto | 0.0854 | |
| Raíz del error medio absoluto | 0.272 | |
| Error relativo absoluto | 28.3035% | |
| Raíz del error relativo absoluto | 70.0138% | |

*Tabla 24. Resultados de la validación cruzada del modelo.*

En la Tabla 24 se pueden ver para estos parámetros obtenidos para nuestro modelo.

Si tenemos en cuenta los resultados de la matriz de confusión (Tabla 22) vemos que las actividades que peor se clasifican son bajar escalera que se confunde con la acción de bajar cuesta, esto es posible dado que los valores que se obtienen para ambas barreras urbanas pueden ser similares. Bajar escalera también se confunde mucho con la acción de andar, esto es posible porque el movimiento relativo del terminal puede ser parecido en ambos casos y llevar a error.

### 7.3.4 Prueba del modelo Weka dentro de Android

La carga del modelo generado utilizando Weka es una de las partes más delicadas dado que puede fallar por muchos motivos. La principal característica que hay que destacar es que el modelo generado tiene que hacerse con la misma versión de Weka que se va a utilizar en la aplicación. Por ello es necesario generar el modelo del clasificador con el fichero .jar que también está incluido dentro de la aplicación y que contiene la versión de Weka adaptada para ser utilizada en Android. Si el modelo se ha generado correctamente no habrá ningún problema al cargarlo dentro de la aplicación.

Para comprobar el funcionamiento del modelo se introdujeron una serie de muestras de las que se conocía su clase original y se procedió a su clasificación con el modelo Weka. El resultado generado se mostró mediante la clase Log en Android Studio para verificar su correcto resultado.

### 7.3.5 Evaluación del sistema completo

Para evaluar la calidad de la aplicación finalizada se han realizado una serie de pruebas para comprobar el funcionamiento en los diferentes tipos de barreras urbanas. Las pruebas se han realizado utilizando la aplicación en la opción de desarrollador y variando el tipo de barrera urbana durante el almacenamiento de datos.

Los resultados obtenidos almacenan los valores reales indicados por el usuario a través de la aplicación y los valores predichos por la propia aplicación.

Con el fin de obtener resultados estadísticos los resultados obtenidos son posteriormente tratados para calcular el tanto por ciento de acierto de la aplicación.

En las siguientes tablas se muestran los resultados obtenidos para las pruebas realizadas. En la segunda fila se muestra la barrera urbana que se estaba detectando y el número de muestras totales de esta barrera. Las siguientes filas contienen las muestras que se han detectado de cada tipo de barrera urbana. Por último, las dos filas finales muestran el porcentaje de acierto de cada barrera urbana individualmente y el de la prueba completa.

| **Test 1 (1488 muestras)** | | | |
|---|---|---|---|
| Predicción/Real | Andar (316 muestras) | Bajar escalera (589 muestras) | Subir escalera (583 muestras) |
| Andar | 158 | 151 | 234 |
| Subir escalera | 31 | 29 | 153 |
| Bajar escalera | 102 | 285 | 128 |
| Subir cuesta | 0 | 10 | 0 |
| Bajar cuesta | 25 | 144 | 67 |
| Acierto (%) | 50 | 48.38 | 26.24 |
| Acierto total (%) | | | 40.05 |

*Tabla 25. Test de la aplicación final 1.*

| **Test 2 (3052 muestras)** | | |
|---|---|---|
| Predicción/Real | Andar (2197 muestras) | Bajar cuesta (855 muestras) |
| Andar | 1107 | 576 |
| Subir escalera | 212 | 81 |
| Bajar escalera | 626 | 149 |
| Subir cuesta | 21 | 0 |
| Bajar cuesta | 231 | 49 |
| Acierto (%) | 50.38 | 5.7 |
| Acierto total (%) | | 37.88 |

*Tabla 26.Test de la aplicación final 2*

| **Test 3 (1512 muestras)** | | | |
|---|---|---|---|
| Predicción/Real | Andar (890 muestras) | Bajar escalera (265 muestras) | Subir escalera (357 muestras) |
| Andar | 420 | 91 | 68 |
| Subir escalera | 94 | 41 | 52 |
| Bajar escalera | 323 | 115 | 123 |
| Subir cuesta | 0 | 5 | 25 |
| Bajar cuesta | 53 | 13 | 89 |
| Acierto (%) | 47.19 | 43.40 | 15.57 |
| Acierto total (%) | | | 38.82 |

*Tabla 27. Test de la aplicación final 3.*

| Test 4 (854 muestras) | | |
|---|---|---|
| Predicción/Real | Andar (560 muestras) | Bajar escalera (294 muestras) |
| Andar | 364 | 92 |
| Subir escalera | 44 | 21 |
| Bajar escalera | 136 | 141 |
| Subir cuesta | 0 | 7 |
| Bajar cuesta | 16 | 33 |
| Acierto (%) | 65 | 47.96 |
| Acierto total (%) | | 59.13 |

*Tabla 28. Test de la aplicación final 4.*

| Test 5 (3283 muestras) | | | |
|---|---|---|---|
| Predicción/Real | Andar (1363 muestras) | Bajar cuesta (557 muestras) | Subir cuesta (1363 muestras) |
| Andar | 704 | 33 | 2 |
| Subir escalera | 243 | 29 | 9 |
| Bajar escalera | 338 | 1 | |
| Subir cuesta | 10 | 92 | 569 |
| Bajar cuesta | 68 | 151 | 665 |
| Acierto (%) | 51.65 | 27.11 | 41.75 |
| Acierto total (%) | | | 43.37 |

*Tabla 29. Test de la aplicación final 5.*

#### *7.3.5.1 Resultados de la evaluación*

Como se puede observar los resultados de la aplicación final no son muy correctos. Si analizamos cada tipo de barrera urbana de forma individual vemos que algunas tienen un porcentaje de éxito similar al de la aplicación completa, mientras que hay que hay otras que obtienen resultados muy insatisfactorios que hacen que baje considerablemente el porcentaje de acierto de la aplicación total.

Estos resultados pueden ser debidos a que los datos utilizados para generar el modelo Weka que se usa para clasificar las muestras nuevas no eran suficientes y por lo tanto el clasificador no es capaz de identificarlas correctamente.

El otro problema principal es que la posición en la que se lleva el teléfono en las pruebas tiene una posición relativa. Es decir, si el smartphone se llevaba en una posición en los ficheros que se almacenaron para generar el modelo del clasificador, para obtener unos resultados correctos al 100% para realizar las pruebas el teléfono debería llevarse exactamente en la misma posición que cuando se tomaron los datos del clasificador. Dado que es prácticamente imposible mantener el smartphone exactamente siempre en la misma posición en todas las pruebas realizadas y, que a la vez esta coincida con la utilizada para generar el clasificador, es difícil conseguir mejores resultados de evaluación de la aplicación.

Las posibles mejoras que permitirían obtener mejores resultados se analizarán en el siguiente capítulo.

# 8 CONCLUSIONES Y FUTURAS LÍNEAS

En este capítulo se tratarán las conclusiones extraídas de la realización de este trabajo y se plantearán unas líneas de trabajo futuras para seguir desarrollando la aplicación una vez finalizado el mismo.

## 8.1 CONCLUSIONES

Con la elaboración de este trabajo lo que se busca es ofrecer a los usuarios una herramienta con la que poder realizar una detección en tiempo real de distintos tipos de barreras urbanas. Esta idea surge del convencimiento de que con los sensores disponibles en los teléfonos y en especial con el acelerómetro se pueden utilizar para detectar muchas cosas. También surge por la seguridad de que se pueden implementar técnicas para la clasificación de patrones dentro de una aplicación móvil.

Además de estas ideas previas para la realización de esta aplicación, la selección del sistema operativo Android está orientada a profundizar y ampliar conocimientos adquiridos durante la carrera.

La aplicación se encuentra programada en su mayoría en lenguaje Java orientado a Android, por lo que es posible utilizar todos los recursos que ofrece el API de Android. Gracias a ello se han podido desarrollar diferentes Activities y capturar los datos proporcionados por los sensores.

El desarrollo de esta aplicación está orientado a integrarse posteriormente en una aplicación de mayor envergadura con un objetivo concreto en el que es necesario la detección de barreras urbanas, como algunos de los ejemplos que han propuesto a lo largo de este trabajo.

Una vez finalizada la primera versión de la aplicación podemos concluir que este trabajo aporta como punto distintivo de otras aplicaciones la utilización de algoritmos de inteligencia artificial por medio del clasificador J48 que permite el reconocimiento de patrones entre diferentes barreras urbanas. Gracias a la obtención de datos previos se puede realizar un entrenamiento que permita la utilización de un clasificador que cumplirá con el requisito de obtener resultados en tiempo real.

## 8.2 LÍNEAS DE TRABAJO FUTURAS

En esta sección se van a plantear unas líneas que debería seguir el trabajo futuro para la mejora de la aplicación una vez vistas las carencias de la misma.

- En primer lugar, se deberían valorar las diferentes opciones en las que los usuarios pueden llevar el teléfono situado. De esta forma se podrían obtener unos resultados que no sean variables con la posición en la que se coloca el smartphone para recoger los datos. Por ejemplo, se podría llevar el teléfono sujeto al brazo o al abdomen dado que así no se variaría su posición durante todo el tiempo, aunque estas posiciones tienen el inconveniente de que es difícil manipular el smartphone, por lo tanto, habría que evaluar cuál es la mejor opción posible.
- En segundo lugar, habría que generar una base de datos lo suficientemente grande utilizando muchos usuarios que recojan datos con diferentes modelos de terminales. De esta forma se podrían utilizar todos esos datos para generar un modelo de clasificador

con unas prestaciones mucho mejores. De esta forma si la probabilidad de muestras clasificadas correctamente del modelo ronda el 100% los resultados finales de la aplicación tendrán una probabilidad de éxito muy elevada. En este caso también habrá que tener en cuenta que no se caiga en el sobreajuste para que el clasificador no esté muy ajustado a los datos de entrenamiento y luego no sea capaz de realizar una correcta clasificación de nuevas muestras.

Una vez concluidos estos puntos principales, se podrían seguir desarrollando las siguientes mejoras:

- **Inclusión de otras barreras urbanas**: si se considera necesario se podrían incluir dentro de las barreras a detectar otro tipo de barreras urbanas que no estuvieran incluidas dentro de las utilizadas en la primera versión de la aplicación. Todo ello requeriría conseguir muchos datos de estas barreas para poder incluirlas dentro del modelo de clasificación.
- **Ampliar las funcionalidades de la aplicación**: como en los ejemplos de futuros usos comentados a lo largo del trabajo se podría incluir un mapa que recoja la situación de las barreras urbanas registradas por los usuarios.
- **Ampliación de sistemas operativos**: una vez se tenga una versión que registre unos resultados satisfactorios se podría evaluar la opción de portar la aplicación a otros sistemas operativos como iOS.
- **Nuevos modelos de clasificación**: se podrían evaluar otros modelos de clasificación diferentes del algoritmo J48 con el fin de obtener mejores prestaciones, como una mayor velocidad, precisión o la opción de que el usuario fuera capaz de entrenar un nuevo modelo desde su terminal.
- **Conexión con redes sociales**: es algo que suelen contener la mayoría de aplicaciones que se encuentran disponibles hoy en día en el mercado y que podría ser útil para que los usuarios compartieran información entre ellos.
- **Portabilidad de la aplicación entre distintos dispositivos**: dada la gran variedad de dispositivos que pueden incluir un sistema operativo Android sería adecuado contar con la opción de disponer de la misma aplicación adaptada tanto para tablets como para wearables.
- **Traducción**: para que la aplicación se pueda comercializar de forma global y se puedan obtener datos en cualquier lugar sobre las barreas urbanas sería conveniente que la aplicación cuente con la opción de ser traducida a cualquier idioma.

# GLOSARIO

- **Acelerómetro**. Es un aparato que sirve para medir aceleraciones. Permite medir en los tres ejes. Cuando el dispositivo se encuentra en reposo mide la gravedad.
- **Activity**. Una actividad es una acción que el usuario puede hacer. También representa la clase del API de Android que se encarga de crear la ventana en la que se colocan todos los elementos que constituyen la interfaz de usuario.
- **API**. Interfaz de Programación de Aplicaciones. Es un conjunto de librerías que contienen clases, paquetes y métodos entre otros que pueden usarse dentro de un sistema operativo como Android.
- **App**. Del inglés application. Es una aplicación informática que se utiliza dentro de teléfonos móviles inteligentes, tablets y otros dispositivos.
- **C/C++.** Lenguajes de programación orientados a la manipulación de objetos.
- **CSV.** Del inglés comma-separated-values. Es un tipo de documento caracterizado porque los datos se presentan en forma de tabla y las columnas que la forman se separan por comas o punto y coma.
- **Hardware**. Son las partes físicas de un sistema informático, por ejemplo, un monitor, un procesador o una memoria.
- **Interfaz**. Es la comunicación entre dos sistemas que permite pasar información entre ellos, por ejemplo, referido a la comunicación entre el usuario y el equipo.
- **Java**. Lenguaje de programación orientado a objetos. Se utiliza para programar aplicaciones Android.
- **KNN**. Del inglés k nearest neighbors. Es un método de aprendizaje automático que se utiliza para la clasificación de elementos mediante un entrenamiento de muestras cercanas en el espacio.
- **Log**. Clase de Android que permite visualizar mensajes que permiten controlar el comportamiento del sistema.
- **Matlab**. Es un software matemático que permite realizar cálculos técnicos. Permite la manipulación de matrices, representar funciones y datos.
- **Memoria RAM**. Memoria de acceso aleatorio (del inglés Random Access Memory). Se utiliza como memoria de trabajo para el sistema operativo, en ella se cargan todas las instrucciones que tiene que ejecutar el procesador.
- **Paquete**. Es un contenedor de clases de Java que permite agrupar las distintas partes de un programa que tienen elementos comunes.
- **SDK**. Kit de desarrollo de software. Es un conjunto de herramientas de software que permite crear una aplicación informática.
- **Sistema operativo.** SO o OS en inglés, es un conjunto de programas que controlan la unidad central, la memoria y los dispositivos de entrada y salida de un sistema informático.
- **Smartphone**. Es un teléfono móvil construido sobre una plataforma informática móvil con capacidad de almacenar datos y con conectividad.
- **Software**. Es el término que describe los componentes no físicos de un sistema informático como los programas o el sistema operativo.
- **SVM**. Del inglés Support Vector Machine, máquina de vectores soporte. Es un conjunto de algoritmos de aprendizaje supervisado. Se puede utilizar para resolver problemas de clasificación y regresión entrenando un modelo a partir de un conjunto de muestras de

entrenamiento podemos etiquetar clases y entrenar una SVM para generar un modelo capaz de predecir una nueva muestra.

- **Toast**. Es un pequeño mensaje emergente que puede aparecer en las pantallas de los smartphones durante la ejecución de una aplicación para mostrar alguna información.
- **Toggle**. Es un tipo de botón que pueden contener las aplicaciones. Se caracteriza porque puede cambiar el texto que contiene según el usuario lo pulse.
- **USB**. Del inglés Universal Serial Bus, es un bus que define los cables, conectores y protocolos usados para conectar y comunicar un ordenador con los distintos periféricos.
- **Usuario**. Es el individuo que utiliza un sistema y realiza diferentes operaciones con él.
- **Wearable**. Es una tecnología que se puede vestir incluido dentro de la ropa. Estos dispositivos permiten realizar varias tareas al mismo tiempo. Dentro de estas tecnologías se incluyen los relojes inteligentes, las pulseras de actividad y las gafas inteligentes.
- **XML**. Del inglés eXtensible Markup Language. Es un metalenguaje que permite definir lenguajes de marcas, se utiliza para almacenar información de forma legible.
- **JAR**. Del inglés Java ARchive. Es un tipo de archivo que permite ejecutar aplicaciones escritas en lenguaje Java.

# REFERENCIAS


[1] Wikipedia, la enciclopedia libre, 2017a. Barrera arquitectónica. En: *Wikipedia* [en línea]. Disponible en: https://es.wikipedia.org/wiki/Barrera_arquitect%C3%B3nica [consulta: agosto 2017].

[2] IDC, 2017. Worldwide Smartphone OS Market Share (Share in Unit Shipments). En: *IDC* [en línea]. Disponible en: https://www.idc.com/promo/smartphone-market-share/os [consulta: agosto 2017].

[3] Eclipse, 2017. Eclipse for Android Developers. En: *Eclipse* [paquete de descarga en línea]. Disponible en: http://www.eclipse.org/downloads/packages/eclipse-android-developers/neonm6 [consulta: agosto 2017].

[4] Android Studio, 2017. Android Studio IDE oficial para desarrolladores. En: *Developers Android* [en línea]. Disponible en: https://developer.android.com/studio/index.html?hl=es-419 [Consulta: septiembre 2017].

[5] Developers Android, 2017a. Versiones de la plataforma. En: *Developers Android* [en línea]. Disponible en: https://developer.android.com/about/dashboards/index.html?hl=es-419 [Consulta: agosto 2017].

[6] Apple Inc., 2017. 89% of devices are using iOS 10. En: *Support App Store* [en línea]. Disponible en: https://developer.apple.com/support/app-store/ [Consulta: agosto 2017].

[7] Mobile Nations LLC,2017. OS Versions – Worldwide. En: *Windows Central* [en línea]. Disponible en: https://www.windowscentral.com/windows-10-mobile-jumps-14-percent [Consulta: agosto 2017].

[8] Developers Android, 2017b. Sensors Overview. En: *Developers Android* [en línea]. Disponible en: https://developer.android.com/guide/topics/sensors/sensors_overview.html [Consulta: agosto 2017].

[9] Weka, 2017. Weka 3: Data Mining Software in Java. En: *Weka. The University of Waikato* [en línea]. Disponible en: http://www.cs.waikato.ac.nz/ml/weka/ [Consulta: septiembre 2017].

[10] GitHub Inc., 2017. Weka for Android. En: *GitHub* [repositorio en línea]. Disponible en: https://github.com/rjmarsan/Weka-for-Android [Consulta: septiembre 2017].

[11] DUDIC, 2017. Accieo. En: *Play Store* [en línea]. Disponible en: https://play.google.com/store/apps/details?id=dudic.accieo [Consulta: agosto 2017].

[12] Politecnico di Milano, 2017. MEP APP. En: *Play Store* [en línea]. Disponible en: https://play.google.com/store/apps/details?id=com.polimi.mep [Consulta: agosto 2017].

[13] Cruz Pérez, Luis, 2017. Zaragoza Accesible. En: *Play Store* [en línea]. Disponible en: https://play.google.com/store/apps/details?id=es.zgzappstore.equipoa.handicapp [Consulta: agosto 2017].

[14] EUR Lex, 2017. Protección de los datos personales. En: *EUR-Lex* [en línea]. Disponible en: http://eur-lex.europa.eu/legal-content/ES/TXT/?uri=LEGISSUM%3Al14012 [Consulta: agosto 2017].

[15]480, 2016. Cómo cumple una app con la ley de protección de datos. En: *480* [en línea]. Disponible en: http://www.cuatroochenta.com/como-cumple-una-app-con-la-ley-de-proteccion-de-datos/ [Consulta: agosto 2017].
[16]González, Ana, 2016. Guía de Protección de Datos para desarrolladores de aplicaciones móviles. En: *Ayuda Ley Protección Datos* [en línea]. Disponible en: https://ayudaleyprotecciondatos.es/2016/06/06/normativa-lopd-aplicaciones-moviles/ [Consulta: agosto 2017].
[17]Ministerio de energía, turismo y agenda digital, 2017. Ley de Servicios de la Sociedad de la Información y del Comercio Electrónico. En: *Ministerio de energía, turismo y agenda digital. Gobierno de España* [en línea]. Disponible en: http://www.lssi.gob.es/paginas/Index.aspx [Consulta: agosto 2017].
[18]Wikipedia, la enciclopedia libre, 2017b. Impacto ambiental. En: *Wikipedia* [en línea]. Disponible en: https://es.wikipedia.org/wiki/Impacto_ambiental [Consulta: septiembre 2017].
[19]Hernández, Uriel, 2015. MVC (Model, View, Controller) explicado. En: *Código facilito* [en línea]. Disponible en: https://codigofacilito.com/articulos/mvc-model-view-controller-explicado [Consulta: septiembre 2017].
[20]Wikipedia, la enciclopedia libre, 2017c. C4.5. En: *Wikipedia* [en línea]. Disponible en: https://es.wikipedia.org/wiki/C4.5 [Consulta: septiembre 2017].
[21]Universidad Carlos III de Madrid, 2003. Minería de datos. En: *it.uc3m* [en línea]. Disponible en: http://www.it.uc3m.es/jvillena/irc/practicas/03-04/18.mem.pdf [Consulta: septiembre 2017].

# BIBLIOGRAFÍA ADICIONAL

A continuación, se listan los documentos que no han sido referenciados durante la memoria pero que también han sido consultados durante la realización del trabajo.

- Campo Vázquez, Celeste y García Rubio, Carlos, 2016. Diapositivas de la asignatura Aplicaciones Móviles. En: *AulaGlobal* [en línea]. Disponible en: http://aulaglobal.uc3m.es/
- Google, 2017a. Subir una aplicación. En: Google support [en línea]. Disponible en: https://support.google.com/googleplay/android-developer/answer/113469?hl=es
- Microsoft, 2017. Regístrate como desarrollador de aplicaciones. En: *Developer Microsoft* [en línea]. Disponible en: https://developer.microsoft.com/es-es/store/register
- Google, 2017b. Centro de políticas para programadores. En: *Google Play* [en línea]. Disponible en: https://play.google.com/intl/es-419_ALL/about/privacy-security/user-data/
- OTT CSIC CV, 2000. Guión para la redacción de planes de difusión y explotación de los proyectos de investigación. En: *CSIC* [en línea]. Disponible en: https://www.dicv.csic.es/pdf/ott/planes.pdf

# ANEXO I: RESUMEN EXTENDIDO

## CONTEXTO

El amplio desarrollo de las tecnologías móviles ha generado un aumento en el mercado de los smartphones. Esto ha derivado en un aumento de la necesidad de todo tipo de aplicaciones móviles como registro de entrenamientos o navegación.

Las barreras urbanas consisten en cualquier obstáculo que impida la movilidad para determinados grupos de población. Este trabajo pretende mejorar la calidad de vida de estos grupos de población utilizando una tecnología extendida dentro de la sociedad.

## MOTIVACIÓN

La principal motivación para la realización de este proyecto es boom que existe en las aplicaciones móviles. Además del reto que supone la implementación completa de una aplicación desde el diseño hasta el desarrollo final.

Otra de las motivaciones para la realización de este trabajo es poder ayudar a la población mejorando la accesibilidad mediante la detección de las barreras urbanas.

Actualmente existen algunas aplicaciones similares al objetivo de nuestro proyecto, aunque se centran en aspectos muy concretos o en localizaciones muy específicas. La aplicación desarrollada busca poder implantarse en un ámbito mucho más amplio.

## OBJETIVOS

El objetivo de este trabajo es desarrollo de una aplicación capaz de detectar diferentes tipos de barreras urbanas en tiempo real utilizando los datos proporcionados por los sensores presentes en el teléfono.

Cuando se desarrolla una aplicación se necesita crear un plan para los siguientes objetivos:

- **Plan de acción**: se estudian las tecnologías seleccionadas para la elaboración del proyecto. En concreto se trabajará con Android Studio, dado que es la herramienta oficial para el desarrollo de aplicaciones Android y se estudiarán los sensores necesarios para la implementación de la aplicación.
- **Diseño de la aplicación**: en esta fase se realiza un primer diseño de la aplicación y se modifica en función de las necesidades y los requisitos planteados. Se define la arquitectura que va a utilizar la aplicación.
- **Desarrollo**: en este punto se toman todas las decisiones en torno a las variables que presenta la aplicación basándose en los requisitos.

Este trabajo es una primera versión de una aplicación que detecta barreras urbanas y que posteriormente podría incorporarse dentro de una aplicación más compleja en la que sea necesaria esta detección previa.

## TECNOLOGÍAS UTILIZADAS

Tras realizar un análisis de las posibles tecnologías que se pueden utilizar para el desarrollo de aplicaciones móviles se ha decidido lo siguiente:

- **Sistema operativo**: los principales sistemas operativos móviles que se encuentran actualmente en el mercado son Android, iOS y Windows Phone. Teniendo en cuenta la evolución del número de usuarios en los últimos años y las facilidades para el desarrollo de aplicaciones que ofrece cada uno se ha decidido utilizar el sistema operativo Android. Windows Phone queda descartado por el mercado tan reducido que tiene. IOS, a pesar de contar con un amplio mercado tiene muy limitada la posibilidad de desarrollar aplicaciones dado que es necesario disponer de diferentes tecnologías de la marca Apple para ello.
- **Entorno de desarrollo**: el desarrollo de aplicaciones Android dispone de dos opciones principales, Elipse y Android Studio. El entorno elegido es Android Studio dado que es el reconocido oficialmente.
- **Sensores**: los sensores utilizados en la aplicación son el acelerómetro y el sensor de gravedad necesario para conseguir unos datos de aceleración que no estén afectados por la componente de la gravedad.
- **Weka para Android**: Weka es una colección de algoritmos para realizar minería de datos. Se ha seleccionado para utilizarse dentro de la aplicación dado que dispone de una versión adaptada para Android y permite realizar una clasificación de muestras en tiempo real.

## MARCO REGULATORIO E IMPACTO SOCIOECONÓMICO

El **marco regulatorio** aplicable a las aplicaciones móviles se centra en proteger la privacidad de los usuarios. En primer lugar, se recogen una serie de leyes y directivas para la protección de los datos de carácter personal que puedan recoger las aplicaciones. Estas leyes se encargan de determinar cómo se van a tratar los datos recogidos, cuál va a ser la finalidad para la que se utilizan estos datos y cómo se va a preservar la privacidad de los usuarios. Estos aspectos quedan recogidos a nivel europeo en la Directiva 95/46/CE y a nivel nacional en la Ley Orgánica de Protección de Datos de Carácter Personal. A nivel nacional también se tiene que cumplir la Ley de Servicios de la Sociedad de la Información y Comercio Electrónico que regula los diferentes aspectos del comercio electrónico.

Por otro lado, se tienen que cumplir las políticas de privacidad. En este caso la aplicación desarrollada debería tener su propia política de privacidad en la que se expliquen todos los aspectos que recogen las leyes de protección de datos personales. También se deben cumplir las políticas de privacidad impuestas por Google para la publicación de aplicaciones dentro de su tienda.

El **impacto socioeconómico** trata de evaluar cómo va a afectar el uso de nuestra aplicación al bienestar de la sociedad y a su economía. Los usos que se pueden dar a nuestra aplicación se centran en eliminar las barreras urbanas que pueda haber en una localización para facilitar la accesibilidad de todos los colectivos. De esta forma se mejora la calidad de vida de ciertos grupos de población y puede conllevar a una mejora de la economía de estas localizaciones. Por lo tanto, el impacto de nuestra aplicación es positivo para sociedad.

## ARQUITECTURA

La aplicación se basa en un Modelo-Vista-Controlador porque permite separar las distintas partes de una forma clara. Por un lado, se encuentra la parte que se muestra al usuario, por otro lado, se encuentra la parte encargada de trabajar con los datos y llevar el peso de la aplicación y por último tenemos una tercera parte que se encarga de conectar la parte que se muestra al usuario con la que procesa los datos.

Si vemos como se organiza una aplicación una parte está compuesta por el código Java que se distribuye en clases y luego tenemos la parte que se componen las distintas pantallas que se muestran en la aplicación que son ficheros XML. Cada pantalla se corresponde con una clase Java.

## REQUISITOS

Los requisitos tanto de usuario como de software permiten identificar todos aquellos elementos que la aplicación debe contener.

- La aplicación debe realizar el reconocimiento de diferentes tipos de barreras urbanas: andar, subir escaleras, bajar escaleras, subir y bajar cuestas.
- El reconocimiento se debe realizar en tiempo real.
- Se deben utilizar los sensores presentes en el teléfono, especialmente el acelerómetro, para obtener datos con los que realizar el reconocimiento.
- La aplicación tiene que poder guardar ficheros con los datos recogidos en el dispositivo.
- Se tienen que incluir dentro de la aplicación técnicas de aprendizaje automático que permitan clasificar nuevos datos.
- La aplicación tiene que permitir iniciar y detener la detección de barreras urbanas e informar al usuario de que se ha iniciado/detenido el reconocimiento.
- La aplicación tiene que mostrar en todo momento en el que se encuentre activo el reconocimiento el tipo de barrera urbana que se está detectando en ese instante.

## IMPLEMENTACIÓN

La aplicación desarrollada hace uso de varias tecnologías presentes en los smartphones y de un sistema de clasificación incorporado dentro de la misma.

La aplicación consta de dos pantallas una inicial que permite elegir entre dos opciones de detección de barreras urbanas y una segunda pantalla que presenta una apariencia diferente en función de la opción seleccionada.

La Figura 0.1 muestra las transiciones entre las distintas pantallas de la aplicación.

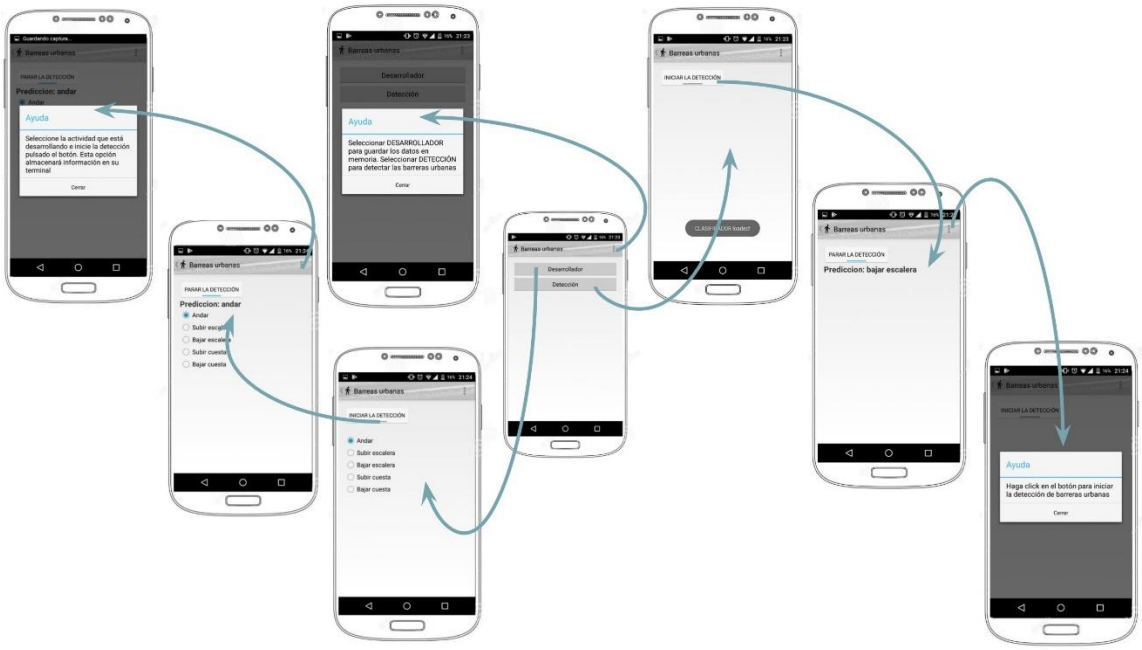


*Figura 0.1. Evolución entre las pantallas de la aplicación*

El desarrollo interno que realiza la aplicación para la detección de barreras urbanas es el siguiente.

En primer lugar, se accede a los sensores del teléfono, en concreto el acelerómetro y el sensor de gravedad. De cada uno de ellos se extraen las componentes x, y, z y se captura el instante en que recogen nuevos valores. Los valores de la aceleración se corrigen con los proporcionados por el sensor de gravedad para eliminar la componente de la gravedad que está incluida en los valores recogidos por el acelerómetro.

Con los datos obtenidos se buscan valores máximos y mínimos con los que buscar diferencias de amplitud y desfases entre máximos/mínimos consecutivos y entre máximos y mínimos.

Todas estas variables son almacenadas en ficheros en la memoria externa del teléfono para poder utilizarlos posteriormente.

Una vez que se llega a este punto se consiguen muchos datos etiquetados para los diferentes tipos de barreras urbanas. Estos ficheros son utilizados para entrenar un clasificador Weka y generar un modelo que se implantará dentro de la aplicación y permitirá realizar el reconocimiento de barreras urbanas en tiempo real.

La aplicación permite sólo realizar el reconocimiento de barreras urbanas con la opción Detección y también permite almacenar ficheros mientras se realiza la detección con la opción Desarrollador.

## VALIDACIÓN

Para comprobar el correcto funcionamiento de la aplicación es necesario realizar una serie de pruebas que permitan verificar que todos los puntos desarrollados cumplen su función.

En este caso la validación se divide en dos partes:

- **Validación de los elementos unitarios de la aplicación**. Consiste en comprobar que todos los elementos que componen la aplicación funcionan de forma correcta. En este caso las pruebas realizadas se basan en conectar el smartphone con el ordenador y comprobar que los resultados del teléfono son iguales a los que se registran en el ordenador utilizando la clase Log y visualizándola en Android Studio. Estas pruebas unitarias se centran en evaluar los sensores, el almacenamiento de ficheros en memoria externa y el funcionamiento del modelo Weka dentro de la aplicación.
- **Validación de la aplicación final**. Para ello se han recogido datos para los distintos tipos de barreras urbanas y se ha almacenado el tipo de barrera y la predicción que realizaba la aplicación. Posteriormente se ha evaluado el porcentaje de acierto que se obtiene.

El test realizado para el conjunto de la aplicación final no ha obtenido unos resultados muy satisfactorios. Esto es debido a que el modelo generado para la clasificación dentro de la aplicación no contenía el mismo número aproximado de muestras para todas las barreras urbanas. Esto hace que la clasificación sea mucho más eficaz para las barreras de las que inicialmente se disponía de más muestras.

El otro problema es que todas las pruebas se realizaron con el teléfono en la mano. Esto afecta a los resultados porque la posición en la que se sitúa el teléfono no es exacta en todas las muestras. Estas variaciones relativas de la posición afectan mucho a un correcto funcionamiento del clasificador.

## CONCLUSIONES

Con la creación de esta aplicación queda claro que se puede ver cómo es posible detectar una gran posibilidad de acciones gracias a los sensores presentes en los smartphones y especialmente con el acelerómetro.

También se puede ver cómo es posible llevar las herramientas de aprendizaje automático a las aplicaciones gracias a los algoritmos proporcionados por Weka, realizando un entrenamiento previo de estos algoritmos que permita generar un modelo que trasladar a las aplicaciones.

La motivación de realizar todas las fases en el desarrollo de la aplicación: diseño, implementación, evaluación y presentación, me ha permitido adquirir una mayor habilidad con esta tecnología. Además de trabajar con ciertas características disponibles como los sensores y los ficheros que no había llegado a utilizar durante la carrera.

La elección del sistema operativo Android vino determinado por el conocimiento previo sobre esta materia y por la facilidad que tiene para el desarrollo de nuevas aplicaciones en comparación con los otros sistemas operativos.

La aplicación está dirigida a poder localizar las diferentes barreras urbanas dentro de un territorio. Esto puede resultar útil tanto a los usuarios que necesiten transitar por lugares accesibles como a las Administraciones Públicas que podrán utilizarla para mejorar la accesibilidad de las ciudades teniendo un registro de donde se encuentran localizadas.

Después de este desarrollo queda claro que el punto más importante a la hora de elaborar cualquier proyecto es una buena organización. Aunque en este caso las aplicaciones van evolucionando según las necesidades que van apareciendo. Un buen diseño será aquel que permita obtener una reducción de costes con una correcta implementación. De esta forma se conseguirá obtener un resultado exitoso.

## FUTURAS LÍNEAS

Dados los resultados que se han obtenido con las pruebas realizadas en la aplicación final hay que trabajar en dos aspectos fundamentales antes de poder centrarse en otros aspectos que se pueden implementar posteriormente.

- Estudiar la posición en la que llevar el teléfono durante la recogida de datos. Vistos los problemas que implica en la clasificación llevar el teléfono en una posición que tiene variaciones relativas se deberán realizar pruebas para obtener una posición en la que llevar el teléfono que no afecte a las mediciones realizadas.
- En segundo lugar, se debe obtener una base de datos lo suficientemente grande que permita que se genere el mejor clasificador posible.

Una vez que se hayan cubierto estos aspectos tendremos la mejor versión posible de la aplicación iniciada con este trabajo. En este punto se podrían realizar una serie de mejoras para obtener un mejor producto final.

- Inclusión de otras barreras urbanas que se puedan detectar.
- Buscar otros aspectos que se puedan incluir en la aplicación que mejoren su uso y funcionalidad. Por ejemplo, incluir la opción de guardar las barreras urbanas localizadas dentro de un mapa.
- Trasladar la aplicación a otros sistemas operativos.
- Utilizar otros modelos de clasificación para obtener mejores prestaciones o incluso permitir que sea el usuario el que entrene su propio clasificador desde el teléfono.
- Conexión de la aplicación con redes sociales.
- Portabilidad de la aplicación entre distintos dispositivos basados en el sistema operativo Android como tablets y especialmente wearables.
- Traducción de la aplicación a otros lenguajes.

# ANEXO II: EXTENDED ABSTRACT

## CONTEXT

The high development of mobile technologies is generating a huge market around smartphones. This situation generates the need of all kind of mobile applications, like the ones for sports tracking or navigation.

Urban barriers are all the objects placed in the street or the buildings. Those barriers increase difficulties for some sections of society to move freely everywhere. This project tries to increase life quality of this people by using an extended technology in current society.

## MOTIVATION

The main motivation to make this project is the boom that involves the market around mobile applications. In addition, the complete development of an application supposes a challenge because you need to make it from the design to the final product.

Another motivation is that this application can help society to increase the access to all places in a city by detecting all the urban barriers that it contains.

Nowadays there are some mobile applications with a similar objective to our project, but the main purpose for all of them is to detect barriers just for one city. The application we are developing looks for detecting the barriers in any location.

## OBJECTIVES

The main objective of this work is to develop a mobile application that could detect some kinds of urban barriers in real time by using the data provided by the sensors placed in the smartphones. This application can detect the following urban barriers: go walking, go upstairs, go down stairs, go up a hill and go down a hill.

When you think about how to develop an application, you need to draw a plan with the next steps:

- **Action plan**: you need to study the technologies chosen for the project. In this case we must work with Android Studio because is the program we can use to develop Android applications. Also, we need to study the sensors we need for the application.
- **Mobile application design**: in this case we need to make a first design of the app. When moving forward on the development we could improve this initial design by including new options and additional user requirements. This point defines the architecture of the app.
- **Development**: at this point we will make the decisions about all the variables that we can modify to accomplish the requirements.

This project makes a first version of an app that can detect some kinds of urban barriers. In following versions, we could use this app inside other more complex applications. This second application will need a previous detection of urban barriers that can be made with our app.

## TECHNOLOGIES USED

After studying all the technologies that can be used to develop a mobile application we decided to focus in the next points:

- **Operating system**: there are three main options about operating systems that are competitive in the market. These options are Android, iOS and Windows Phone. After analyzing the evolution of the mobile sales and the options that each one provides to develop new apps we decided to use the Android OS. The Windows Phone option provides good possibilities to develop new apps but it has a minority market so the app couldn't arrive to many users. The iOS technology has the opposite situation to the Windows Phone. In this case iOS have a good market place with lots of users but the options to developing new apps are very closed, because you need to have the whole Apple software.
- **Development IDE**: we have two options about the IDE for developing Android apps. We could use Android Studio or Eclipse. In this case Android Studio is the official IDE so we chose this option. Eclipse was used before Android Studio appeared.
- **Sensors**: the sensors we need to use in the application are the accelerometer and the gravity sensor. We need the gravity sensor to eliminate the gravity component that has the accelerometer data.
- **Weka for Android**: Weka is a collection of algorithms to make data mining. In this case we choose to use Weka because it provides a version that we can include in the Android app to classify the new data obtained with the sensors. It allows us to do the classification in real time.

## REGULATORY FRAMEWORK AND SOCIOECONOMIC IMPACT

The **regulatory framework** that can be applied to the mobile applications makes a huge effort to protect the privacy off the app users.

The first thing we need to know is that there are some directives and laws that can be applied to the apps. These laws try to protect the personal data that the apps obtain from their users. These laws pretend to know how developers will use the data they are saving from the users. Also, they want to know what is the final purpose to obtain this data and how the developers are going to safe the personal identity of the users.

In the European Union these terms are focused in the Directive 95/46/CE. In Spain we have the Ley Orgánica de Protección de Datos. Also, in Spain we have the Ley de Servicios de la Sociedad de la Información y Comercio Electrónico that controls the electronic commerce.

On the other hand, the apps need to have some privacy policy. In this case the application that we are developing needs to have their own privacy policy where we can explain all the terms that are required the laws. Also, we need to obey the privacy policy from Google because it is mandatory to publish our app in their store.

The **socioeconomic impact** tries to evaluate how is our application going to affect the society wellness and the economy. The uses we think are going to be applied to our app are based in identifying and removing all the urban barriers that can be found in a location. This action could improve accessibility some areas so that could also improve the life of certain social groups.

Also, this action could improve economy in these locations. That's why our app could generate a positive impact on society.

## APPLICATION ARCHITECTURE

The application uses a Model-View-Controller because this model allows us to separate the parts of the project in an easy way. The first section is focused on the screens we will show to the user. Then we have another section that works with the data and is the most important part of the app. The last section is the one that connects the part that we show to the user and the part that process all the data.

If we see the internal app organization we have the Java code that has class distribution and the part that contains the screens that the user can see, this part is organized in XML files. Every screen match with a Java class.

## REQUIREMENTS

All the requirements, the user and the software ones, allows us to know all the elements that the application should contain.

- The app should make the urban barriers recognition for different kind of barriers: walking, go upstairs, go down stairs, go up a hill and go down a hill.
- The recognition should be done in real time.
- We should use the sensors that are available in the smartphone, in this case the accelerometer and the gravity sensor. With these sensors we should obtain all the data to make the recognition.
- The application should be able to save files in the smartphone memory card. These files should contain the data to make the recognition of the urban barriers.
- The app should contain the machine learning algorithms to classify new data.
- It should allow the user start and stop the recognition of the urban barriers and should notify when the recognition starts and stops.
- The app should show the urban barrier detected when the recognition is on.

## IMPLEMENTATION

The mobile application that we are developing uses some different technologies that are present in the smartphones and a classification algorithm that is get into the app.

The application contains two different screens. The first one asks the user to choose the kind of urban barrier recognition he wants to start. The second screen shows a different appearance according to the urban barrier recognition that the user had chosen.

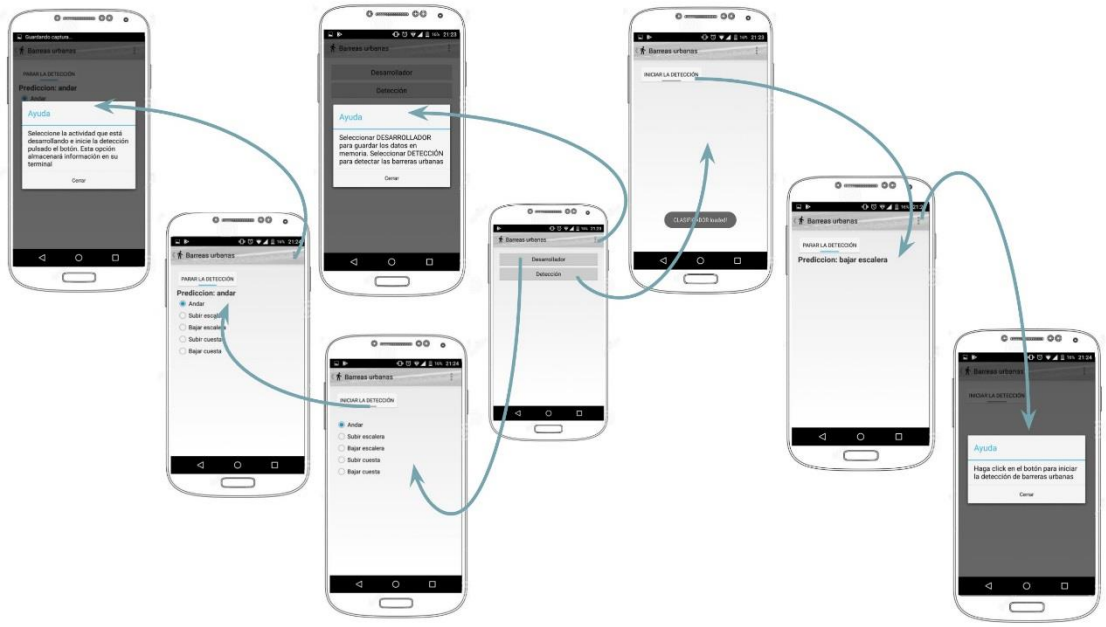

*Figure 0.2. App navigation between screens.*

The Figure 0.2 shows the navigation between the different screens that contains the app.

The internal process that allows the app to detect the urban barriers is the following one.

The first thing that we need to do is to have access to the sensors of the phone. In this case we want to get data from the accelerometer and the gravity sensor. From each one we extract the x, y and z components and we save the time when a new data is captured from the sensors.

The acceleration data should be corrected because they contain some information from gravity. So, as we want to work with a real acceleration we need to correct this data with the information that the gravity sensor provides to us.

With all this data we are going to look the maximum and minimum data to compare them and look for the amplitude and the delay between two maximum, two minimum and one maximum and one minimum that are consecutive.

All these variables are saved in a file in the external memory card of the phone, because we are going to use it later.

When the application is in that point we need to obtain lots of data with the label of the urban barrier that the user has in front of him.

This files with the data are going to be used to train a classifier with Weka and generate a model that is going to be run inside the smartphone. This model allows us to make the real-time recognition of the urban barriers.

One important thing about the real-time recognition is that we need to use the sensor in the fastest way that it can capture data. So, if we show to the user the information of the prediction that is making the app with this speed he won't be able to see anything because the speed of the letters in the screen is too high. For this reason, we decide only to show the prediction once per second. In this case the user can see the information and he can still thinking that the recognition is being doing in real time.

The application allows to make the recognition of the urban barriers with the Detection ("Detección") option and allows to save files and make the recognition with the detection data with the Developer option ("Desarrollador").

## VALIDATION

We need to check if the application works like we want it to, so we need to make some test to validate that all the points the app contains works fine.

In this case we can do two different kind of validations:

- **Validation of all the individual parts of the app**. This part is to check that all the elements that we have developed individually work fine. In this case all the tests that we have done have the same pattern. We connect the smartphone with the computer to check if the result that we have in the phone is the same that the one we have in the computer. We use the class Log to see in Android Studio the messages with the information we are looking for. The main purpose of these tests is to check the sensors, the process of saving data in the external memory of the phone and the classification made with the Weka model are working properly.
- **Complete application validation**. In this case we need to save lots of information about the different urban barriers. This data should contain the information of the barrier that the user has in front of him and the prediction that the classifier makes. After that we can evaluate the results and calculate the average of well predicted samples.

With the results of the final test we can say that the final application does not get such well results as we might expect. This could be the consequence of two things.

The first thing is the model generated with Weka. If the model we have included in the app doesn't contain a similar quantity of samples for each urban barrier, then the result won't be satisfactory. If we have more samples for some kinds of urban barriers we are going to predict well these barriers and then we have a worst prediction for the barriers with less samples.

The other problem is the position of the phone during the test. All the samples for these tests has been taken with the phone in the hand. We have discovered that this position is not good because the phone is not going to be in the exactly same position all the time. These tiny variations in the phone position could affect the model we're creating and could affect the prediction that is making the classifier.

## CONCLUSIONS

With the application developed we can confirm the previous idea that anything can be predicted with the sensors placed in the smartphones and specially with the accelerometer.

Also, we can see how is possible to use the machine learning in the mobile applications with the Weka algorithms. We should train this algorithms before the model generation and then we can use these models in the applications.

The main motivation for the project is being able to do all the parts related to develop a new Android application: design, implementation, evaluation and presentation of the final project.

This project has allowed me to increase my knowledge about this technology and to work with some options like sensor and files that I've had never used before.

The election of the Android operating system had been determinate by previous knowledge about this science and for all the opportunities that we have for developing new applications compared to the other operating systems.

This application has been developed for locating all the urban barriers placed in a city. This action can be useful for the users when they need to walk in easily accessible places. Also, can be useful for the Government because they have a technology that provides information about what places can be improved.

After this project we can affirm that the most important thing in the mobile application developing is the previous planning. Although, in these cases the applications can be modified when new necessities appear. A good design would be able to incorporate these changes with no more costs and with a good implantation. In this way we would obtain a successful result.

## FUTURE LINES

We have seen the main issues in the developed project. The first thing we should complete are two main tasks.

- First, we should investigate which is the best position we can find to situate the phone during the tests. This position should be the same in all the measuring because we have seen that if we can not do this point the result of the app is disappointing.
- Second, we must create a huge data base with more or less the same number of samples for each urban barrier in this way we can get the best classifier we are able.

When these main problems are solved, we will have the best version of the application initiated in this project. After that we can evolve the app in other ways to obtain a complex final product.

- We can include other urban barriers in the options for detection.
- We can include some options to make a more efficient app. For example, we can add a map where we can place the urban barriers detected.
- We can develop the app for other operating systems.
- We can introduce new classification models which obtain better results. In addition, we can try to include inside the app the option for training the model inside the smartphone.
- We can connect the app with the social networks.
- We can adapt the app to other Android formats like tablets and especially to Android Wear.
- We can translate the application into other languages.