



TRABAJO FIN DE GRADO

INGENIERÍA EN INFORMÁTICA

Sistema de posicionamiento en interiores basado en WiFi

Autor:

Javier Bueno López

Director:

Antonio Cañas Vargas



Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Granada, Septiembre de 2020

Sistema de posicionamiento en interiores basado en WiFi

Autor:

Javier Bueno López

Director:

Antonio Cañas Vargas

Granada, Septiembre de 2020

Sistema de posicionamiento en interiores basado en WiFi

Javier Bueno López (alumno)

Palabras clave: SWADroid, Android, Java, Localización

Resumen

El trabajo de fin de grado que a continuación se presenta consiste en el desarrollo de un módulo para una aplicación. Este módulo permitirá conocer la ubicación aproximada de un usuario dentro de un edificio perteneciente a una institución (por ejemplo, una facultad o una escuela), siempre que este haya activado y dado permisos a dicha aplicación.

La localización se establecerá en función a distintos parámetros obtenidos de los puntos de acceso inalámbricos distribuidos por las distintas dependencias de los edificios que constituyen la institución. La aplicación estará integrada con un servicio web que se ejecutará en el servidor de una plataforma docente. La base de datos de dicha plataforma almacenará las ubicaciones de los puntos de acceso y del usuario y permitirá a otros usuarios conocer dónde se encuentra el primero, siempre que este lo desee.

Aunque el profesor/estudiante no desactive su localización, se garantiza que nunca se guardará ni se informará de la ubicación si se sale del edificio. Además, sólo se guardarán los datos de ubicación actual en un periodo breve para que puedan ser consultados en tiempo real, es decir, no se conservará ningún histórico de ubicaciones.

Indoor positioning system using WiFi

Javier, Bueno López (student)

Keywords: SWADroid, Android, Java, Location

Abstract

The final degree project presented below consists in the development of a module for an application of a teaching platform. This module will allow knowing the approximate location of a user inside a building (for example, a faculty or a school), as long as the user has activated and given permissions to this application.

The location will be established according to different parameters obtained from the wireless access points distributed by the different locations of the building. The application will be integrated with a web service that will run on the teaching platform's server. The database of this platform will store the locations of the access points and the user and will allow other users to know where the first one is located, whenever they wish.

Even if teachers or students does not disable their location, it is guaranteed that their location will never be saved or reported if they leave the building. In addition, only the current location data will be saved in a short period so that it can be consulted in real time, that is, no location history will be kept.

Yo, **Javier Bueno López**, alumno de la titulación del Grado en Ingeniería Informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 77147399V, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Javier Bueno López

Granada a 1 de Septiembre de 2020.

D. **Antonio Cañas Vargas**, Profesor del Departamento de Arquitectura y Tecnología de Computadores de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Sistema de posicionamiento en interiores basado en WiFi***, ha sido realizado bajo su supervisión **Javier Bueno López**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 1 de Septiembre de 2020.

El director:

Fdo: Antonio Cañas Vargas

Agradecimientos

A mi tutor ayudarme con la realización de este trabajo.

A mi familia y mis amigos, por todos los buenos consejos y el apoyo recibido durante estos años.

Contenido

1	Introducción.....	18
1.1	Motivación.....	18
1.2	Objetivos.....	19
1.3	Planificación.....	19
1.3.1	Planificación temporal.....	19
1.3.2	Planificación de los prototipos.....	21
1.4	Presupuesto.....	23
2	Estado del arte.....	26
2.1	Análisis del dominio.....	26
2.1.1	Localización.....	26
2.1.2	Aplicación Android.....	29
2.1.3	Infraestructura ETS de Ingenierías informática y de Telecomunicación.....	33
2.2	Tecnologías disponibles.....	35
2.2.1	Códigos NFC/NFT/QR.....	35
2.2.2	Bluetooth.....	35
2.2.3	Sistemas de navegación inercial.....	35
2.2.4	LiFi.....	36
2.2.5	Sonidos.....	36
2.2.6	Wifi.....	36
2.3	Trabajos relacionados.....	37
2.4	Aplicaciones similares.....	39
3	Propuesta de solución.....	42
3.1	Metodología de trabajo.....	42
3.1.1	Planificación.....	43
3.1.2	Implementación.....	44
3.1.3	Revisión.....	44
3.1.4	Retrospectiva.....	44
3.2	Requisitos.....	45
3.2.1	Requisitos funcionales.....	45
3.2.2	Requisitos no funcionales.....	45
3.2.3	Requisitos de información.....	45
3.3	Recursos empleados.....	46
3.3.1	Lenguaje de programación.....	46

3.3.2	Entorno de desarrollo integrado	47
3.3.3	Sistema de control de versiones.....	47
3.3.4	Base de datos.....	48
3.3.5	Despliegue	49
3.4	Diseño de la solución	50
3.4.1	Algoritmo de localización basado en la pérdida básica de transmisión en el espacio libre.....	50
3.4.2	Recolección de información	54
3.4.3	Desarrollo API propia	57
3.4.4	Comunicación con SWAD	58
3.4.5	Diseño	61
3.5	Implementación.....	64
3.5.1	Compartir ubicación	67
3.5.2	Buscar usuario	72
3.5.3	Base de datos.....	77
3.5.4	Algoritmo	78
4	manual de usuario	81
4.1	Introducción.....	81
4.2	Funciones.....	81
4.2.1	Habilitar ubicación	81
4.2.2	Compartir ubicación	83
4.2.3	Buscar la ubicación de un usuario	85
5	Conclusiones y trabajos futuros	87
5.1	Trilateración.....	87
5.2	Adición de centros	88
6	Bibliografía	90
7	Índice de figuras.....	94
8	Índice de tablas.....	96

1 INTRODUCCIÓN

En este capítulo se realizará una introducción al problema que se a abordado en el trabajo de fin de grado. Consta a su vez de cuatro secciones: motivación, objetivos, planificación temporal y presupuesto.

En la sección de motivación se hablará de las problemáticas que dieron origen a este trabajo, así como la posible solución a nivel informativo. Posteriormente, se definirán los objetivos que se deben alcanzar con la realización de este trabajo. Para lograr estos objetivos será necesaria una planificación previa y la definición de los prototipos funcionales que se irán desarrollando para comprobar el progreso del proyecto a niveles generales. Finalmente, se realizará una estimación presupuestaria de los costes derivados de la realización de este proyecto.

1.1 MOTIVACIÓN

Actualmente, si un alumno quiere ponerse en contacto con un profesor de un mismo centro educativo existen dos opciones. La primera sería escribir a esta persona para concertar una cita. La segunda sería acudir directamente al despacho del profesor en cuestión. Si elegimos la primera opción tendremos que esperar a que el profesor responda. Si elegimos la segunda nos arriesgaremos a que el profesor haya tenido que abandonar su despacho por algún motivo. Este caso es el origen de la idea que supuso el desarrollo de este trabajo de fin de grado.

El ámbito de actividad de un sistema de este tipo no está reducido a alumnos y profesores, sino que cualquier usuario de la plataforma docente podrá encontrar a cualquier otro usuario que lo permita. La intención no es ubicar a usuarios para que sean rastreados con otras finalidades que no sea agilizar las comunicaciones. Es por este motivo que solo se compartirán ubicaciones de los usuarios cuando ellos lo deseen.

Además, en este trabajo no se profundizará en la ubicación de forma exacta de usuarios ya que no es el objetivo, sino más bien en establecer una ubicación aproximada. Para ello hará uso de los sensores de los dispositivos móviles actuales y de la infraestructura típica de la que hacen uso las Universidades.

Finalmente, este módulo se comunicará con una plataforma docente desde la que se podrán ir viendo las ubicaciones de los usuarios que decidan compartirla. En conclusión, se podrá usar el móvil para compartir la ubicación y esta podrá ser consultada por otros usuarios tanto desde la aplicación móvil como desde la plataforma docente en la que se llevará a cabo un desarrollo de forma paralela al de la aplicación.

1.2 OBJETIVOS

El desarrollo de este proyecto supone la cumplimentación de una serie de objetivos que se irán alcanzando a medida que se vaya progresando en la ejecución del mismo. Entre esta serie de objetivos podemos distinguir dos tipos: los principales y los secundarios. Los principales son aquellos objetivos que resultan indispensables para resolver los problemas presentados en la sección anterior. Los secundarios derivan de los principales y por lo general los complementan o mejoran.

El objetivo principal de este proyecto es la implementación de un módulo para SWADroid que permita a los usuarios de esta plataforma compartir su ubicación dentro de la Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada.

Como objetivos secundarios se podría destacar la ampliación de esta cobertura a más Universidades y centros en los que la plataforma SWAD también estuviese en uso. Otros objetivos secundarios serían la realización de una interfaz para el usuario sencilla e intuitiva o agilizar comunicaciones entre miembros de una misma institución mediante el uso de sus dispositivos móviles.

1.3 PLANIFICACIÓN

Para planificar el proyecto por un lado se realizará la planificación temporal con el desglose por cada mes y por otro la planificación de los prototipos que se irán desarrollando.

1.3.1 Planificación temporal

Para realizar una planificación temporal se dividirán las tareas entre los meses de forma esquemática.

Noviembre

- Reunión con el tutor y formalización de la idea.
- Investigación sobre herramientas, algoritmos, tecnologías y aplicaciones disponibles.
- Clonación de SWADroid de GitHub.
- Definir siguientes pasos.

Diciembre

- Desarrollo de una aplicación de prueba para testear opciones disponibles.
- Creación de un nuevo repositorio.
- Creación de un nuevo proyecto Android.
- Recolección de datos de puntos de acceso distribuidos por las dependencias de la Escuela.
- Preparación de base de datos con puntos de acceso para probar la aplicación.
- Comienzo de la documentación del proyecto.
- Decisión del algoritmo tras realizar pruebas.
- Retrospectiva.

Enero

- Migración de aplicación propia a módulo de SWADroid.
- Formación sobre SWADroid.
- Realizar cambios y sincronizaciones periódicas con upstream.

Febrero

- Continuar con la migración del proyecto propio.
- Documentación.
- Retrospectiva.

Marzo

- Creación de microservicio MacLocation con una API que gestione la base de datos emulando el comportamiento que debería tener la plataforma docente.
- Mejorar aspecto visual del módulo IndoorLocation para SWADroid.

Abril

- Despliegue de microservicio MacLocation.
- Integración entre SWADroid y MacLocation.
- Retrospectiva.

Mayo

- Comunicar al tutor el estado del proyecto.
- Implementación de funciones en SWAD.
- Refactorización del código.
- Integración con API SWAD.

Junio

- Documentación del trabajo.
- Mejoras en vistas.
- Frecuencia de actualización.
- Retrospectiva.

Julio

- Continuar con la documentación.

Agosto

- Continuar con la documentación.
- Retrospectiva

Septiembre

- Pruebas reales del sistema en la Escuela.
- Recopilación de fallo.

1.3.2 Planificación de los prototipos

Esta planificación conlleva el desarrollo de 3 prototipos funcionales más pequeños cuya finalidad es realizar un seguimiento real de trabajo y poder recopilar incidencias que solucionar en la planificación de los siguientes meses en función a un orden de prioridad.

A continuación, paso a enunciar los prototipos que se han generado.

Indoor Positioning

En este primer prototipo se buscaba realizar una **aplicación propia sencilla** con la que poder escanear las redes wifi cercanas y mostrar parámetros relevantes de las mismas que sean útiles para el posterior desarrollo de la aplicación. El código de este proyecto se aloja en un repositorio de GitHub.

El desglose en tareas es el siguiente:

- Escanear redes wifi cercanas al accionar un botón.
- Mostrar MAC de cada punto de acceso.
- Mostrar SSID de cada punto de acceso.
- Mostrar intensidad de señal de cada punto de acceso.
- Algoritmo sencillo.

MacLocation

En este proyecto se buscó de **simular un entorno** en el que funcionaría la aplicación final. Para ello se recopilaban todos los datos de los puntos de acceso del edificio de la Facultad haciendo uso de la primera aplicación para posteriormente realizar un microservicio desde el que poder añadir puntos de acceso, modificarlos y eliminarlos. Por último, se desplegó una API a la que se realizan peticiones desde el cliente de SWADroid. El código de este proyecto se aloja en un repositorio de GitHub.

El desglose en tareas es el siguiente:

- Recolección de datos de los puntos de acceso de la ETSIIT.
- Desarrollo de una API con información de los puntos de acceso de la ETSIIT.
- Adaptación del código propio al cliente SWADroid.
- Pruebas reales en la Escuela.
- Establecer tiempo de escaneo programable.

SWADroid

Este último prototipo tiene como objetivo **incorporar las funcionalidades** de los anteriores prototipos en el cliente de SWAD para Android. Para ello es necesario realizar un conjunto de funciones a las que realizar peticiones de forma similar a como se hacía antes, pero con diferentes datos.

El desglose en tareas sería el siguiente:

- Integración con la API de SWAD.
- Mejorar el aspecto visual.
- Realización de test unitarios y de integración.
- Pruebas reales en la Escuela.

1.4 PRESUPUESTO

Este trabajo sigue la filosofía de sus predecesores en cuanto a usar únicamente software libre. Para el caso de SWADroid se aplica la licencia **GNU General Public License v3.0** [1].








							
Type	Proprietary	Permissive	Permissive	Permissive	Copyleft	Copyleft	Copyleft
Provides copyright protection	✗ FALSE	✓ TRUE	✓ TRUE	✓ TRUE	✓ TRUE	✓ TRUE	✓ TRUE
Can be used in commercial applications	✗ FALSE	✓ TRUE	✓ TRUE	✓ TRUE	✓ TRUE	✓ TRUE	✓ TRUE
Provides an explicit patent license	✗ FALSE	✓ TRUE	✗ FALSE	✗ FALSE	✗ FALSE	✗ FALSE	✗ FALSE
Can be used in proprietary (closed source) projects	✗ FALSE	✓ TRUE	✓ TRUE	✓ TRUE	✗ FALSE	✗ FALSE partially	✗ FALSE for web
Popular open-source and free projects		Kubernetes Swift Firebase	Django React Flutter	Angular.js JQuery, .NET Core Laravel	Joomla Notepad++ MySQL	Qt SharpDevelop	SugarCRM Launchpad

Figura 1: Licencias de código abierto más populares. Recuperada de artículo web [2]

Al ser un desarrollo realizado para el cliente SWADroid la licencia que se aplicaría sería la misma.

En cuanto a costes de personal los gastos de tener una persona contratada se podrían desglosar de la siguiente manera:

Sueldo bruto anual	20.000€
Retenciones por IRPF	2.338€
Cuotas a la Seguridad Social	1.270€
Sueldo neto anual	16.392€
Tipo de retención sobre la nómina	11,69%
Sueldo neto mensual	1.366€

El resto de gastos se dividen entre el material inventariable y el no inventariable.

Material inventariable	Coste
1 ordenador portátil para el desarrollo y documentación.	1100€
1 teléfono móvil Android para realizar pruebas con SWADroid	300€
3 rúters para realizar pruebas	140€
Total	1540€

Material no inventariable	Coste
Elaboración de encuestas de satisfacción	1000€
Total	1000€

Gasto	Coste
Material inventariable	1540€
Material no inventariable	1000€
Personal	20.000€
Total	22540€

2 ESTADO DEL ARTE

En este capítulo se realizará un estudio sobre el estado tecnológico de la localización en interiores en la actualidad, así como la plataforma e infraestructura donde va a ser implementado. Para ello se analizará el dominio del problema y se buscarán todas las tecnologías disponibles para implementar un sistema de este tipo. Finalmente se observarán trabajos similares de los que poder tomar ejemplo.

2.1 ANÁLISIS DEL DOMINIO

Esta sección está dedicada a conocer el ámbito en el que opera el problema descrito, así como todos los conceptos interrelacionados con el mismo. Esta sección contará a su vez de tres apartados que se consideran los pilares fundamentales de conocimiento previo a la realización del proyecto software. Estos tres pilares son: la localización, la aplicación Android y la infraestructura en la que se implementará el sistema.

2.1.1 Localización

La geolocalización consiste en denotar la posición de un objeto o persona en un contexto geográfico. Esto se consigue mediante el uso de dispositivos que combinan **capacidad de procesamiento** con uso de **sensores** para la obtención de información de su entorno a partir de la cual pueden deducir la posición de una entidad. Los sistemas de localización que hacen uso del **GPS** son muy populares, tanto es así que el tamaño del mercado fue estimado en 37.9 USD billones en 2017 [3]. El crecimiento al que este mercado ha sido sometido se debe a la penetración de los dispositivos con GPS integrado como por ejemplo móviles o vehículos.

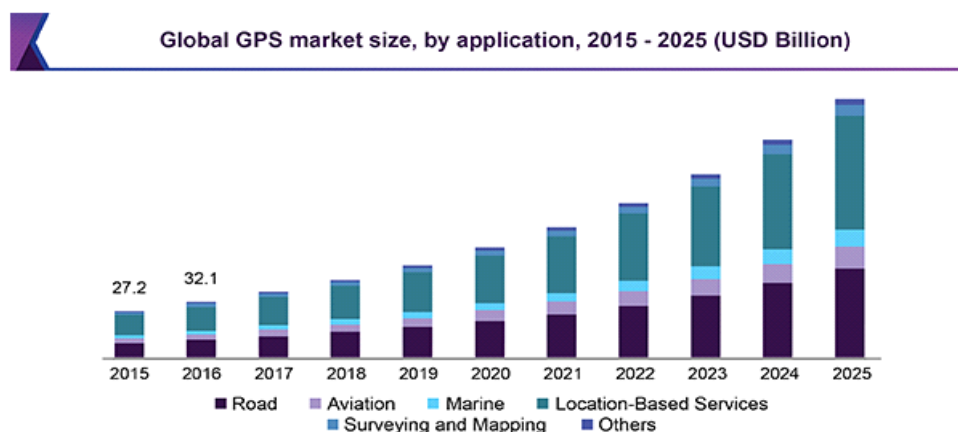


Figura 2: Tamaño del mercado del GPS desde 2015 a 2020. Recuperado de GrandViewResearch [4]

Estos dispositivos hacen uso de satélites que orbitan alrededor de la Tierra y que suministran datos basados en su posición espacial que es enviada por parte de los mismos dispositivos hasta unos receptores. Los receptores son estaciones en la Tierra que se encargan de interpretar los datos y haciendo uso algoritmos de trilateración son capaces de obtener como resultado la posición de un usuario.

La trilateración es una forma de obtener la posición de un objeto en el espacio que ya usaban cartógrafos alrededor del 1600 para realizar mediciones como por ejemplo la altura de un acantilado. Posteriormente **Willebrord Snell** descubrió que con 3 puntos se podía localizar un punto en un mapa, esto se consigue con la superposición de las circunferencias definidas por cada punto en el mapa con su correspondiente radio. [5]

El GPS o sistema de posicionamiento global siempre estuvo en mente de las grandes potencias de finales de los 80-90. Siendo de esta manera Estados Unidos el primero en poner en prueba su **GNSS** (Sistema global de navegación por satélite) en la Guerra del Golfo. Actualmente podríamos destacar tres sistemas GNSS:

- **GALILEO**: Es el sistema europeo de radionavegación y posicionamiento por satélite. [6]
- **GLONASS**: Desarrollado por la URSS y actualmente administrado por la Federación Rusa. [7]
- **NAVSTAR GPS**: Desarrollado por el Departamento de Defensa de Estados Unidos y actualmente propiedad de la Fuerza Espacial de EEUU. [8]

El GPS hace uso de la triangularización que es un sistema empleado para equipos de navegación más antiguos. Se basa en el área que ocupan las ondas de un emisor que dibujan una circunferencia en condiciones ideales.

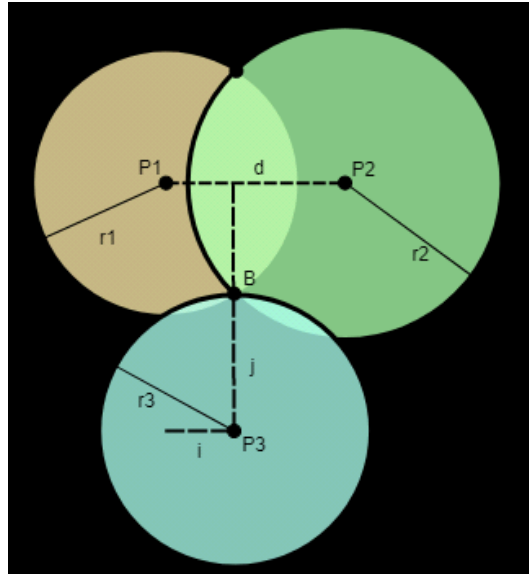


Figura 3: Triangularización ejemplificada sobre un plano bidimensional. Recuperado de Wikipedia [9]

Supongamos que queremos conocer la posición de B a partir de las circunferencias con centro en P1, P2 y P3 respectivamente. La posición vendrá determinada por la intersección de estas 3. Obsérvese que los sistemas de localización pueden precisar más la posición si se añaden más circunferencias que contengan al punto B. Hay que concretar que este ejemplo se basa en latitud, longitud, pero los que se usan en la realidad hacen uso de 3 dimensiones. Sin embargo, el principio que se usa es el mismo.

Esto a nivel satelital se traduce en que el satélite será el centro de la esfera y los receptores GPS determinarán el radio de esta esfera. En ese momento el receptor obtendrá la distancia de al menos dos esferas más y realizará los cálculos matemáticos apropiados para determinar la posición. Para establecer cuál es el radio de cada circunferencia se medirá el tiempo que tarda en llegar la información desde cada satélite al receptor.

Este sistema de tan avanzada edad ha ido pasando por diferentes etapas de desarrollo hasta llegar a la actualidad como lo que conocemos por **GPS III** siendo previsiblemente lanzado el tercer satélite de esta generación en abril de 2020. Con esta última generación de satélites se obtendrán mejoras sustanciales respecto al anterior llegando a obtener una precisión mejor a 1-5 m (generación anterior) y proporcionando soporte a los posibles cambios futuros que permitan satisfacer la carga de usuarios hasta 2030.

La localización en interiores va al alza siguiendo el compás a los sistemas GPS anteriormente descritos. La diferencia que hay entre unos y otros sistemas es el **espacio en el que actúan**, uno trata de localizar a nivel terrestre a una persona mientras que otro lo hace dentro de un lugar cerrado. El modelo de funcionamiento que se sigue es el mismo que el anteriormente descrito para el GPS, pero con hardware distinto. No obstante, adelante se profundizará más sobre esta idea. Este tipo de localización permite determinar la posición de usuarios dentro de lugares cerrados donde el GPS es incapaz de hacerlo.

Actualmente existe una gran infraestructura de red que permite la conexión de millones de dispositivos a internet. Este número no deja de aumentar llegando a la previsión de que en 2025 habrá 100.000 millones de dispositivos conectados a la red. Todo este crecimiento se lleva produciendo desde la aparición del estándar **802.11** [10] que recoge la interconexión de dispositivos de manera inalámbrica y fue anunciado en 1997. Este protocolo y sus sucesores conformaron una revolución tecnológica nunca vista hasta entonces y que hoy conocemos como Internet. La interconexión de redes supone la adquisición de hardware específico para unir estas redes, en el caso de las conexiones inalámbricas hablamos de rúters a los que se conectaran los dispositivos que a su vez dispongan de adaptadores de red que permitan conexiones inalámbricas (implementen el protocolo 802.11). De esta manera queda conformada una infraestructura que puede tener otras aplicaciones aparte de la anteriormente descrita. Aquí es donde entra en juego el posicionamiento en interiores, existen muchas formas de implementar un sistema de este tipo. La infraestructura jugará un papel fundamental a la hora de tomar cualquier tipo de decisión. En los siguientes capítulos explicaré todas las **alternativas** disponibles.

2.1.2 Aplicación Android

Como ya se ha mencionado anteriormente el sistema de localización se integrará en una plataforma docente. La plataforma elegida es SWAD y el módulo estará disponible para la aplicación que implementa alguna de sus funcionalidades **SWADroid**. Esto supone una restricción de diseño ya que al trabajar sobre un proyecto ya existente hay que adaptarse a las tecnologías escogidas por el desarrollador o grupo de desarrolladores iniciales. En este caso la aplicación está escrita en Java y se comunica con la API del servidor usando **SOAP** como protocolo de comunicación, lo que significa que se usará **XML** como protocolo de intercambio. Ahora que he descrito de forma general la estructura y funcionamiento de la aplicación pasaré a explicar las características que se pueden emplear para afrontar el desarrollo del módulo de localización.

SWADroid hace uso del nivel de API 29 (Android 10) esta es la última versión del SDK. Esto nos permite usar la API de **Wifi RTT (Round-Trip-Time)** siempre que los rúters implementen RTT. Esta solución permite a partir de tres medidas a rúters cercanos realizar un algoritmo de multilateración para determinar la distancia de un dispositivo con una precisión de 1 a 2 metros. En principio parece una solución candidata a un posible desarrollo final ya que además es bastante simple de implementar. Sin embargo, tiene un problema fundamental, necesita que los dispositivos móviles que se usen implementen el protocolo **802.11mc** [11] [12]. El número de dispositivos que actualmente soportan este protocolo es bastante reducido de momento y además no dispongo de ninguno de ellos que me permita realizar pruebas por la facultad. Este motivo si bien no descarta la solución descrita la deja con pocas posibilidades de llegar a ser la final ya que la intención es que el sistema de localización esté disponible para el mayor número de dispositivos Android posibles.

Otra opción que evade el problema anteriormente explicado sería poner en práctica un modelo similar al de la trilateración empleado por el GPS usando la potencia de señal y una aproximación para la pérdida de señal. Esto sería fácil de realizar en Android. Habría que escanear las redes cercanas (en la primera opción igual) y comparar la dirección física de los resultados con una lista con los puntos de acceso del centro y sus ubicaciones.

Para escanear las redes cercanas podemos usar un **Broadcast Receiver** [13] que no es más que un receptor de transmisiones. Una vez se reciba el conjunto de redes candidatas habrá que filtrarlas para obtener un conjunto de redes formado por redes únicamente registradas en la base de datos. Sobre este conjunto final realizaremos las operaciones de cálculo pertinentes para obtener la ubicación. Se podrían usar cuantos puntos de acceso se quisiese, pero esto aumentaría el tiempo de cómputo y derivaría en una peor experiencia de usuario. Además, la finalidad de este proyecto no es revelar la ubicación de forma exacta y precisa, esto además generaría cierto nivel de desconfianza en los usuarios. Se pretende dar la ubicación de manera aproximada, así que con uno o dos puntos de acceso podría ser suficiente.

Ahora analizaré la estructura del proyecto de SWADroid.

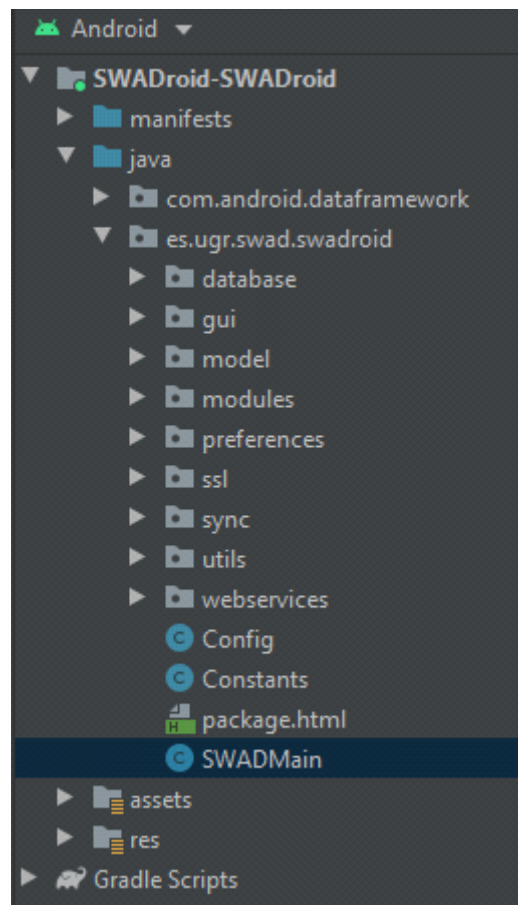


Figura 4: Organización en directorios de SWADroid. Recuperada de Android Studio con el proyecto de SWADroid abierto.

/manifests: Directorio común a todos los proyectos Android. Es un fichero de definición de *Activities* y de los permisos que necesitará la aplicación entre otros. Contiene un solo fichero **AndroidManifest.xml**.

/java: Común a todos los proyectos en *Java*. Dentro se encuentran los paquetes y el código fuente de la aplicación funcione.

/assets: Contiene los iconos de *font awesome* y *MathJax* necesario para representar fórmulas matemáticas.

/res: Contiene los diseños de *Activities* y diseños propios como botones, listas, cuadros de texto, etc.

Si avanzamos un nivel de profundidad en el directorio donde se encuentra el código fuente de java podemos encontrar dos paquetes: *com.android.dataframework* y *es.ugr.swad.swadroid*

En la primera podemos encontrar las clases usadas para comunicarse con la base de datos. En la segunda podemos encontrar una jerarquía de directorios:

/database: Clase complementaria a la del otro paquete de la base de datos.

/gui: Contiene diseños gráficos para barras de progreso, diálogos, fuentes especiales, etc.

/model: Dentro se encuentran todos los modelos necesarios para las funcionalidades de SWADroid.

/modules: Es el directorio más complejo, en él se encuentran todos los módulos que implementa la aplicación.

/preferences: Contiene clases que definen las preferencias globales para la aplicación como tiempo de sincronización para las notificaciones, por ejemplo.

/ssl: Compone las utilidades SSL para establecer una comunicación segura con la API de SWAD.

/sync: Código correspondiente para la realización de sincronizaciones llevadas a cabo de forma periódica.

/utils: Principalmente compuesto de clases criptográficas usadas para realizar cifrado de datos.

/webservices: En este paquete se encuentra un cliente SOAP que se usa para realizar peticiones a la API de SWAD.

2.1.3 Infraestructura ETS de Ingenierías informática y de Telecomunicación

La infraestructura que se usará de prueba resulta fundamental para el desarrollo del módulo de la aplicación. En este caso se usará la ETSIT como entorno de pruebas. Cuenta con un sistema de cableado estructurado que sigue la normativa definida en el documento que podemos encontrar en CSIRC [14]. Antes de 2004 la red de la UGR funcionaba haciendo uso de **CVI-UGR** que ya ofrecía una cobertura al 100% de los campus. En 2005 se incrementaron las prestaciones de la red añadiendo **VPN**, **EDUROAM** y **802.1x/WPA** entre otras mejoras, además se migró de CVIUGR a **CVIUGR2**.

Estas redes están compuestas por rúters CISCO entre los que se destacan [15]:

- Cisco Aironet 1100 Series Access Point
- Cisco Aironet 1300 Series Access Point
- Cisco Aironet 1230 Series Access Point

En las especificaciones de estos rúters podemos además encontrar el rango que alcanzan tanto en interior como en el exterior.

Item	Specification			
	802.11a: <ul style="list-style-type: none"> • 90 ft (27 m) at 54 Mbps • 225 ft (69 m) at 48 Mbps • 300 ft (91 m) at 36 Mbps • 350 ft (107 m) at 24 Mbps • 400 ft (122 m) at 18 Mbps • 450 ft (137 m) at 12 Mbps • 475 ft (145 m) at 9 Mbps • 500 ft (152 m) at 6 Mbps 	802.11g: <ul style="list-style-type: none"> • 90 ft (27 m) at 54 Mbps • 95 ft (29 m) at 48 Mbps • 100 ft (30 m) at 36 Mbps • 140 ft (43 m) at 24 Mbps • 180 ft (55 m) at 18 Mbps • 210 ft (64 m) at 12 Mbps • 220 ft (67 m) at 11 Mbps • 250 ft (76 m) at 9 Mbps • 300 ft (91 m) at 6 Mbps • 310 ft (94 m) at 5.5 Mbps • 350 ft (107 m) at 2 Mbps • 410 ft (125 m) at 1 Mbps 	802.11a: <ul style="list-style-type: none"> • 170 ft (52 m) at 54 Mbps • 350 ft (107 m) at 48 Mbps • 550 ft (167 m) at 36 Mbps • 700 ft (213 m) at 24 Mbps • 800 ft (244 m) at 18 Mbps • 875 ft (267 m) at 12 Mbps • 925 ft (282 m) at 9 Mbps • 950 ft (290 m) at 6 Mbps 	802.11g: <ul style="list-style-type: none"> • 110 ft (34 m) at 54 Mbps • 200 ft (61 m) at 48 Mbps • 225 ft (69 m) at 36 Mbps • 325 ft (99 m) at 24 Mbps • 400 ft (122 m) at 18 Mbps • 475 ft (145 m) at 12 Mbps • 490 ft (149 m) at 11 Mbps • 550 ft (168 m) at 9 Mbps • 650 ft (198 m) at 6 Mbps • 660 ft (201 m) at 5.5 Mbps • 690 ft (210 m) at 2 Mbps • 700 ft (213 m) at 1 Mbps
	Ranges and actual throughput vary based upon numerous environmental factors so individual performance may differ.			

Figura 5: Rango de rúters CISCO. Recuperada de Barcodesinc [52]

Todos estos rúters son los que se ubican por las distintas dependencias de la facultad. La instalación de los rúters se realizó mediante anclajes fijos, en falsos techos, candados y a altura no accesible, de esta manera se garantiza la seguridad de los mismos sin perder efectividad.

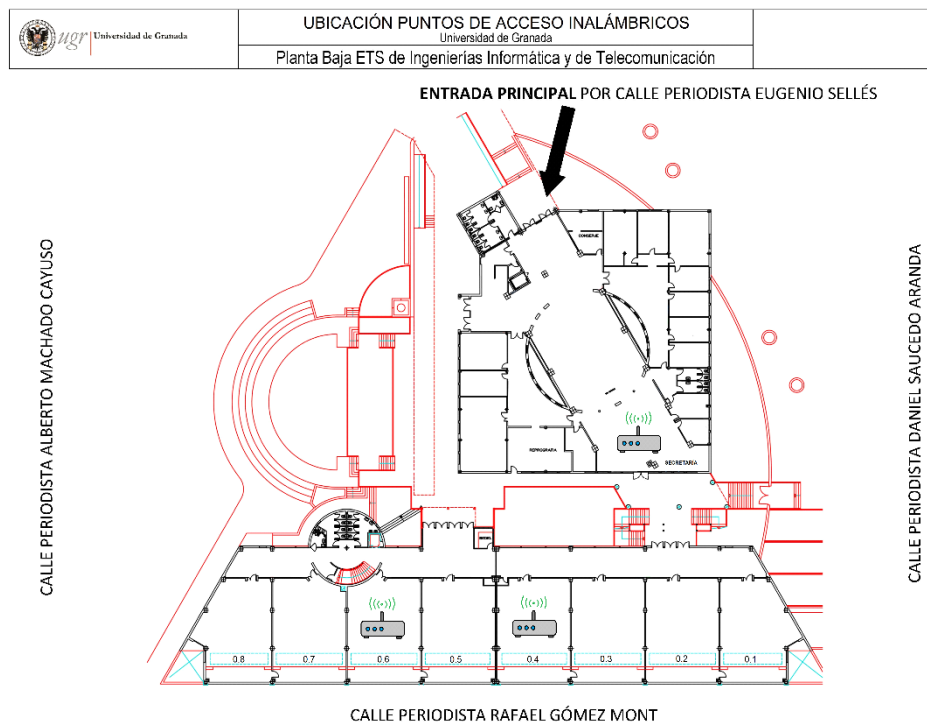


Figura 6: Ubicación de los puntos de acceso en Planta 0. Imagen reutilizada [16]

2.2 TECNOLOGÍAS DISPONIBLES

A continuación, se debatirán las tecnologías que se pueden aplicar en este proyecto y se tratará de comparar las ventajas e inconvenientes que aportan cada una.

2.2.1 Códigos NFC/NFT/QR

Todos estos métodos se utilizan de la misma manera a pesar de ser de distinta naturaleza. El procedimiento consiste en acercarse a un lugar en el que poder escanear o registrar un código y a partir de ahí establecer la ubicación. Es un sistema que requiere de la interacción de la persona para poder realizar el proceso de localización. Existen otras formas de realizar este procedimiento de forma más transparente como camuflar esos códigos en imágenes u otro tipo de elementos que componen el diseño del edificio.

2.2.2 Bluetooth

El comportamiento de estos sistemas es similar al de la localización mediante puntos de acceso, sin embargo, existen otro tipo de inconvenientes aparte de los ya existentes para la localización por puntos de acceso. El mayor de ellos es la inexistencia de una infraestructura que incluya emisores Bluetooth ya que apenas son usados para este tipo de fines. Estas balizas Bluetooth no tienen un coste demasiado elevado, además usan una pila de 3V que les concede una autonomía de unos 3 años al ser BLE, tienen un alcance de unos 60 metros. Sin embargo, hay que modificar ligeramente la estructura interior de los edificios para fijarlos en paredes y hay que asumir un gasto moderado para cubrir un edificio medianamente grande. Tanto Apple como Google proporcionan protocolos que permiten realizar geolocalización usando estas balizas. Presentaron errores de seguridad importantes en el pasado que parecen haber sido subsanados en las últimas actualizaciones.

2.2.3 Sistemas de navegación inercial

Este tipo de sistemas hacen uso de sensores para establecer la posición de un dispositivo dentro de un edificio. Más concretamente hacen uso del giroscopio y del acelerómetro. La principal desventaja de estos sistemas es que resulta difícil establecer cuál es el punto inicial desde el que empezar a trazar la trayectoria que se sigue. Además, un fallo en una medición de los sensores (algo relativamente común) hace que sea de obligado uso otro sistema complementario para calibrar automáticamente sensores. Los grandes problemas de este sistema son la descalibración de los sensores y la complejidad.

2.2.4 LiFi

Consiste en la modularización de la luz led de forma invisible para el ojo humano de forma que la cámara de un dispositivo móvil sea capaz de interpretarla y determinar la posición. Esta solución resulta muy curiosa y realmente podría dar buenos resultados, pero sin embargo presenta un grave problema. Si llevamos el dispositivo móvil en el bolsillo u oculto no podrá usar la cámara. Otro inconveniente sería la adquisición y preparación del hardware necesario para llevar a cabo esta técnica.

2.2.5 Sonidos

Se hace uso del micrófono del móvil que se encarga de interpretar los sonidos que percibe de su entorno. Como se puede deducir esto tiene muchos problemas ya que no en todos los sitios se pueden emitir sonidos y además el sonido ambiente puede falsear esas lecturas de sonido.

2.2.6 Wifi

Existen muchas formas de obtener la ubicación usando wifi. Una de ellas sería aprovechando el **RTT** o **round time trip** que es el tiempo que tarda en llegar y volver un paquete, este método se implementó en el protocolo 802.11mc. Este protocolo está disponible para los móviles Android desde la actualización Android P, lo que supone que dispositivos móviles con una versión anterior no podrán hacer uso de este protocolo.

Otra forma de obtener la ubicación mediante wifi es usando la potencia de señal a los rúters más cercanos. Esta opción no es tan precisa como la anterior, pero si se puede implementar en cualquier tipo de dispositivo.

Tecnología	NFC/NFT/QR	Bluetooth	Navegación inercial	Lifi	Sonidos	Wifi RTT	Wifi FSPL
Hardware del dispositivo	Cámara	Adaptador Bluetooth	Giroscopio y acelerómetro	Cámara	Micrófono	Adaptador de red	Adaptador de red
Material adicional	Etiquetas	Balizas	—	Luces LED	Altavoces	Rúters	Rúters
Interacción con el medio	✓	—	—	—	—	—	—
Coste	→	→	↓	→	→	↓	↓
Modificación del entorno	↓	↑	↓	→	↑	↓	↓
Mantenimiento	↓	↓	↑	→	↓	↓	↓

Tabla 1: Comparación de los distintos sistemas empleados para la localización. Elaboración propia.

2.3 TRABAJOS RELACIONADOS

En esta sección se exponen trabajos relacionados con la localización en interiores que puedan servir de ejemplo para la realización de este proyecto.

Visitas guiadas por museos

Hace menos de un año la UJI diseñó un asistente virtual con geolocalización avanzada para las visitas guiadas. Este sistema tiene el objetivo de ayudar a realizar visitas guiadas por un museo, definir puntos de interés, incluir información adicional, etc. Para realizar la geolocalización hace uso de tecnología wifi, balizas Bluetooth y GPS. Estos elementos actúan de forma cooperativa para poder establecer con detalle posición y orientación del usuario. Incluye además un plano del recorrido seguido por el museo. [17]

Control de errantes

Otro campo sobre el que se han realizado sistemas es sobre salud. Actualmente existen muchos sistemas de localización para ancianos y discapacitados, existen tanto trabajos como empresas que se encargan de ello. Entre sus funciones se concibe localizar, registrar y detectar los riesgos para personas con demencia en una residencia. La mayoría de ellos hacen uso de ultrasonidos y de radiofrecuencias que son imperceptibles para los pacientes de la residencia como forma de posicionamiento.

Marketing basado en localización

El marketing digital también hace uso de la localización en interiores. En la tesis se realizó un sistema de posicionamiento en interiores haciendo uso de balizas Bluetooth. El sistema está integrado en una aplicación Android que hace uso de la localización del usuario para mostrar en su pantalla marketing de proximidad. De esta forma el vendedor podrá aumentar sus ganancias y los clientes podrán mejorar su experiencia al realizar compras.

Localización de personal a bordo

Otro proyecto que resulta interesante comentar es. Esta vez se usó como recinto sobre el que localizar un buque de la Armada, más concretamente el patrullero “Tabarca”. La finalidad era el **control del personal a bordo** del buque. Para ello se combinaron dos tecnologías ya comentadas anteriormente para lograr mayor precisión para resolver situaciones más complejas. Las tecnologías elegidas son wifi y Bluetooth BLE. [18]

Detección de infectados COVID-19

Un último ejemplo de trabajo relacionado más actual es el que se ha hecho para localizar a personas infectadas y personal potencialmente infectadas durante la pandemia del COVID19. Estos sistemas actúan de muchas formas y no siempre en interiores, pero el principio es el mismo. Detectan los lugares en los que estuvo una persona infectada a partir de wifi o Bluetooth principalmente y manteniendo un historial de los dispositivos conectados a esa red en esa fecha son capaces de determinar las personas con las que pudo estar en contacto el infectado. Para establecer con precisión la distancia que mantuvo con esas personas se emplea el principio de triangularización explicado anteriormente. [19]

2.4 APLICACIONES SIMILARES

FIND

Es un framework Open Source que nos permite ubicar dispositivos en interiores sin incluir sensores y haciendo uso de la red y Bluetooth. Proporciona una API en la que podemos almacenar, borrar y consultar las distintas ubicaciones. Para dispositivos móviles incluye una aplicación desde la que se pueden introducir las ubicaciones directamente de cada lugar para posteriormente ser consultadas. Usa algoritmos de machine learning para conseguir establecer la posición.



Figura 7: Icono FIND. Recuperada de FIND [20]

SITUM

Es una alternativa de pago que además ofrece soporte para Android mediante un SDK. Además de proporcionar localización y seguimiento en interiores aporta información geográfica. Se puede visualizar el plano del edificio y seguir el lugar en el que está un dispositivo. Incluye documentación para desarrolladores muy completa.



Figura 8: Icono situm. Recupera de Situm [21]

Aplicaciones usadas en residencias y hospitales

Existen muchas aplicaciones para residencias, todas comparten la estructura principal. Funcionan por radiofrecuencia y hacen uso de pulseras que se colocan sobre las manos o las piernas de los pacientes, de forma que si uno de ellos se acerca a una zona catalogada como no permitida se genera una alerta al personal sanitario. Esto libera de carga a todo el personal ya que no se tienen que tener cuidados tan precisos con estas personas y es más fácil identificar situaciones de riesgo. Entre estas aplicaciones podemos encontrar: Accessor, Kimaldi, Ibernex, Neat...

3 PROPUESTA DE SOLUCIÓN

En este capítulo se discutirá la solución escogida y los motivos que desencadenan la elección de la misma. Además, se establecerá una metodología de trabajo para el desarrollo de esta propuesta de solución que irá conllevando la creación de pequeños prototipos funcionales que muestren el progreso de la misma. Por otro lado, se elegirán una serie de herramientas las cuales serán utilizadas para el desarrollo de esta propuesta. Finalmente se expondrá el algoritmo usado, así como su justificación e implementación en la aplicación.

3.1 METODOLOGÍA DE TRABAJO

Como metodología de trabajo he usado **SCRUM**. Este tipo de metodología es ágil y se usa para trabajar normalmente en equipos. Incluye entregas parciales que sirven para obtener resultados rápidamente de forma que en el siguiente paso se puedan añadir requisitos o definir más los que previamente se especificaron. El procedimiento que he seguido se podría representar de la siguiente manera:

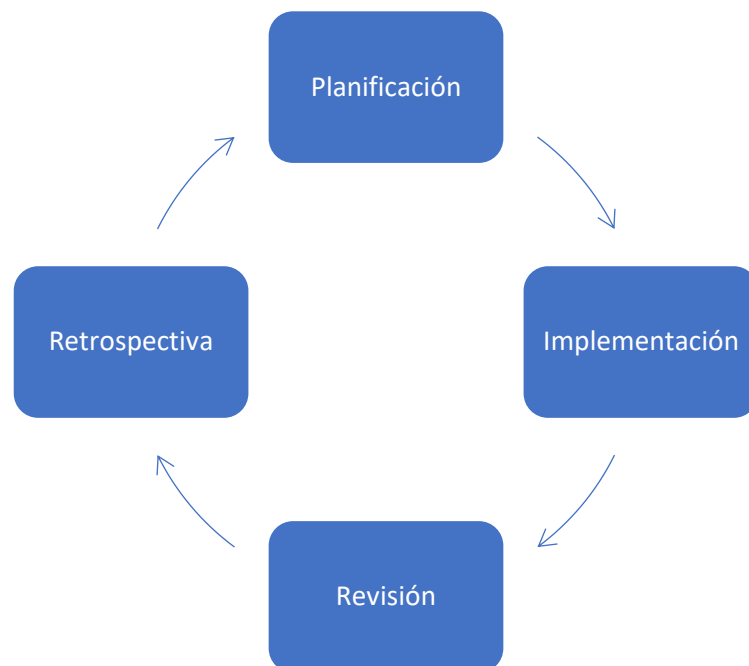


Figura 9: Metodología de trabajo seguida en SCRUM. Elaboración propia.

A continuación, se detallan cada una de las fases de esta metodología, así como la forma en que han sido aplicadas al proyecto.

3.1.1 Planificación

En cuanto a la selección de requisitos se realizó un procedimiento coordinado con los stakeholders para decidir cuáles se iban a pedir para la siguiente iteración, así como la asignación de una serie de prioridades a los mismos. Los requisitos al principio no tienen por qué estar muy bien definidos ya que al aplicar una metodología ágil pueden redefinirse o eliminarse en futuras iteraciones. Estas reuniones se han llevado a cabo a través de la plataforma de software libre Jitsi.

A partir de estos requisitos el equipo se reúne y establece las siguientes tareas en función a la complejidad y el esfuerzo. De esta manera cada uno se asigna su propio trabajo apoyándose de la valoración del equipo. Para este proyecto, las decisiones que se tomaron abarcaban principalmente iteraciones para desarrollo de software de la aplicación móvil y para el crecimiento de la API del servicio web de la que hace uso. Para definir todas estas tareas se hace uso de un **Scrum Taskboard**. Llegados a este punto podemos elegir varias opciones, Trello o GitHub entre otras. Se realizaron unas pequeñas pruebas piloto para cada prototipo y finalmente se decidió usar *Trello*. La decisión fue tomada basándose en la experiencia del equipo y la facilidad de uso de una plataforma frente a otra.

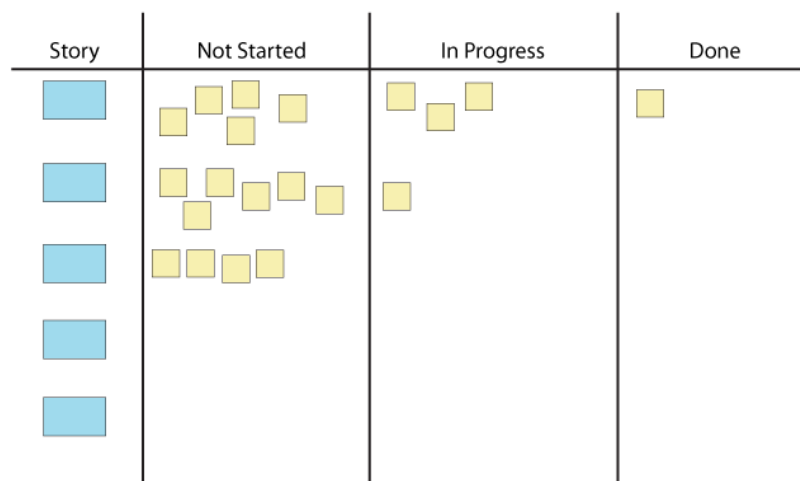


Figura 10: Scrum Taskboard. Recuperada de AGILE FOR ALL. [56]

3.1.2 Implementación

En esta etapa se toma ese conjunto de tareas priorizadas y se implementan. Además, se van detectando nuevos requisitos y riesgos que se transmiten al resto del equipo y los stakeholders de modo que serán tenidos en cuenta para futuras iteraciones o se redefinirán en esa misma iteración en el caso de ser necesario.

3.1.3 Revisión

Se entrega el producto al cliente y este decidirá cuáles son los siguientes cambios que quiere añadir en función del resultado obtenido. Normalmente, estos entregables son prototipos o pequeñas aplicaciones funcionales que permitan conocer el estado del producto. En este proyecto se realizaron concretamente tres aplicaciones funcionales mediante el prototipado rápido. En el resto de iteraciones se aplicaban mejoras sobre uno de estos prototipos.

3.1.4 Retrospectiva

El equipo se reúne para poner en común qué se ha hecho bien, que se ha hecho mal y que se debería mejorar. Con estas reuniones se consigue escalar las dificultades y los riesgos de forma que sean más fáciles de resolver para el equipo de desarrollo. Este rol normalmente suele corresponder a una sola persona. Finalmente se repite el proceso hasta que se consiga el producto final.

3.2 REQUISITOS

En este apartado haré referencia a los requisitos finales, ya que durante el proceso de desarrollo se han ido definiendo nuevos, cambiando antiguos y eliminando otros.

3.2.1 Requisitos funcionales

- RF1. Integración con la API SWAD.
 - 1. Guardar ubicaciones.
 - 2. Consultar ubicaciones.
 - 3. Enviar actualizaciones de ubicación de forma periódica.
- RF2. Dejar de compartir ubicación si el usuario cuando el usuario lo desee.
- RF3. Establecer frecuencia de actualización.
- RF4. Determinar la ubicación de un dispositivo dentro del edificio.
- RF5. Visualización de la información en listado.

3.2.2 Requisitos no funcionales

- RNF1. Escanear los puntos de acceso más cercanos.
- RNF2. La aplicación funcionará en la mayor parte de dispositivos Android.
- RNF3. La aplicación debe solicitar los permisos necesarios para su funcionamiento.
- RNF4. Las vistas serán sencillas e intuitivas.

3.2.3 Requisitos de información

- RI1. El módulo almacenará de forma local las ubicaciones del usuario.
- RI2. El módulo se encargará de enviar la información de cada usuario para ser almacenada en SWAD.

3.3 RECURSOS EMPLEADOS

En este apartado trataré de describir los recursos empleados para la realización de este trabajo, así como una justificación de la elección de los mismos.

3.3.1 Lenguaje de programación

Al ser un proyecto desarrollado en Android de forma nativa las opciones se reducen a dos lenguajes: Java y Kotlin.

- **Java:** En sus inicios conocido como *OAK*, nació como un lenguaje de programación parecido a C++ pero que fuese orientado a objetos y además usase una máquina virtual propia. Tiene licencia GNU GPL. [22]
- **Kotlin:** Presentado como un Java mejorado en 2016 y siendo desarrollado por JetBrains. Es un lenguaje que soluciona muchos detalles de Java (null, casting...). Tiene licencia Apache 2. [23]

Se decidió usar **Java** debido a que el equipo tiene más experiencia. Además, SWADroid lanzó su primera versión el *3 de noviembre de 2010*, aún no existía Kotlin y la única opción era usar Java para desarrollo nativo Android. Se consideró por lo tanto oportuno continuar usando por lo tanto Java.

Por otro lado, para realizar un microservicio para el segundo prototipo se debe elegir un framework que permita de forma rápida y sencilla realizar un prototipo funcional válido que emule un ambiente de trabajo. Se han considerado cuatro framework populares con los que se podría realizar esta API a nivel de backend. A continuación, se enumeran los mismos, así como la decisión final y el porqué de esta.

- **Flask:** Es un microframework de Python, está caracterizado por su alto grado de sencillez. [24]
- **Django:** Más completo que Flask, pero igualmente usa Python incorpora funciones para formularios, autenticación, etc. [25]
- **Spring Boot:** Usa Java y es muy completo. La ventaja que ofrece es su curva de aprendizaje. [26]
- **Laravel:** Hace uso de PHP y también es muy completo como el resto de opciones. Emplea el patrón de diseño MVC. [27]

Finalmente se decidió usar Flask ya que la curva de aprendizaje es poco pronunciada y además implica un desarrollo más rápido frente al resto de opciones. Por último, añadir, que los demás frameworks incluían una serie de utilidades que no se necesitaban puesto que el desarrollo que se iba a hacer era sencillo y a nivel de aprendizaje esencialmente.

3.3.2 Entorno de desarrollo integrado

Como la aplicación final es nativa y Android se pueden considerar dos opciones principalmente: Android Studio y Eclipse. A continuación, paso a compararlas.

- **Android Studio:** Nace en 2013 como reemplazo a Eclipse como IDE de desarrollo oficial para Android. Está basado en IntelliJ IDEA. [28]
- **Eclipse:** Fue sustituido por Android Studio, sin embargo, consume menos recursos que su sucesor lo cual lo hace apto para su uso en algunos dispositivos. [29]

Finalmente se decidió usar Android Studio ya que se consideró que existe una serie de ventajas de un IDE frente a otro que hacen indispensable el uso del mismo. Desde la refactorización de código hasta detección de errores y warnings, Android Studio proporciona mejoras sustanciales; esto se debe principalmente a que **indexa todo el proyecto** mientras que Eclipse no lo hace. Sin embargo, esto como ya se ha concretado anteriormente concurre en que Android Studio consume mayor número de recursos que Eclipse por lo que será necesarios equipos para el desarrollo de software con un hardware medio, siendo casi imposible trabajar con equipos informáticos muy económicos.

3.3.3 Sistema de control de versiones

Como no podría ser de otra forma se ha elegido **Git** como sistema de control de versiones. Fue diseñado por *Linus Torvalds* y lanzado en 2005 como sustituto a los antiguos sistemas de control de versiones. Actualmente existen muchas plataformas online en las que poder alojar proyectos que usen Git, entre ellas podemos destacar las siguientes:

- **GitHub:** Fue adquirida el 4 de junio de 2018 por Microsoft. Es posiblemente el sitio que aloja más proyectos. Incorpora Wiki para cada proyecto, páginas web, tableros Kanban... [30]
- **GitLab:** Es más empleado para proyectos DevOps ya que incluye funciones que GitHub no incluye de forma directa. [31]
- **Bitbucket:** Producto de Atlassian lo que significa que tiene una alta sinergia con Jira y Confluence que son herramientas destinadas a la documentación y planificación de proyectos. [32]

Se ha optado por usar GitHub ya que SWADroid está desarrollado en esta plataforma. Personalmente he trabajado en todas y considero que las mejores son tanto GitHub como BitBucket, la primera por la gran comunidad que hay detrás y la segunda por las integraciones que he comentado anteriormente.

3.3.4 Base de datos

Otro punto fundamental para la realización de este proyecto es la elección del tipo de sistema gestor de base de datos. Para empezar, se pueden diferenciar dos tipos de base de datos principalmente, las relacionales y las no relacionales.

Gestores de bases de datos no relacionales

Hay muchas opciones a tener en cuenta, para reducir por popularidad se han elegido dos. Cada uno de ellos fue usado para el desarrollo de un prototipo distinto.

- **Firebase:** Se uso para el primer prototipo por facilidad de uso y porque incluía un Tier gratuito para estudiantes. [33]
- **MongoDB:** Se uso para el segundo prototipo ya que el equipo contaba con conocimientos previos sobre este sistema de base de datos. Para alojar la base de datos usé MongoDB Atlas que incluye un cliente bastante intuitivo llamado *Compass*. [34]

3.3.4.1 Gestores de bases de datos relacionales

Si hablamos de SGBD podemos encontrar una amplia gama sobre la que elegir. Como se viene haciendo con anterioridad se elegirán los más populares para someterlos a una comparativa.

- **MySQL:** Es el sistema más extendido, es muy sencillo de usar y tiene interfaces bastantes buenas como MySQLWorkbench o PhpMyAdmin. [35]
- **MariaDB:** Proporciona una alternativa de código abierto alternativa al ecosistema MySQL de Oracle. [36]
- **PostgreSQL:** Rinde mejor en bases de datos más extensas y proporciona más funciones que el resto. [37]

Este sistema gestor de bases de datos relacionales es usado por la plataforma docente que proporciona la API de modo que todos los puntos de acceso se encuentran almacenados ahí.

3.3.5 Despliegue

En esta parte se hará referencia únicamente al despliegue del microservicio realizado con el microframework Flask. Para ello se discutirán unas cuantas opciones que personalmente considero que son las más destacables basándome en la facilidad de uso.

- **Heroku:** Es un PaaS que permite realizar despliegues de manera muy sencilla mediante su *CLI de Heroku*. [38]
- **AWS:** Ofrece muchos servicios y muchas opciones para el despliegue de aplicaciones, si pasas los límites establecidos para un Tier gratuito cobran automáticamente la diferencia. [39]
- **GCloud:** Es una plataforma que reúne todas las aplicaciones de Google destinadas al desarrollo web. Su principal ventaja es la calidad de la documentación y la comunidad que hay detrás [40]

Se eligió Heroku porque el equipo tenía experiencia previa con el despliegue de aplicaciones con este PaaS, por otro lado, es la opción que permite desplegar aplicaciones de la forma más sencilla y sin apenas realizar ningún tipo de configuraciones. Si no se hubiese elegido Heroku se habría generado un contenedor Docker [41] de la aplicación realizada con Flask y se desplegaría ese contenedor en cualquiera de las otras opciones comparadas anteriormente. Para mantener esta aplicación actualizada se tagearía y se subiría a DockerHub de forma que cualquiera pudiese hacer uso de ella y además se automatizasen los despliegues con un sistema de continuous developing como TravisCI o GitHub Actions.

3.4 DISEÑO DE LA SOLUCIÓN

3.4.1 Algoritmo de localización basado en la pérdida básica de transmisión en el espacio libre

La pérdida básica de transmisión en el espacio libre es la atenuación que sufre una señal al propagarse desde un punto a otro. Podemos encontrar la información referente al cálculo de la atenuación en el espacio libre en la especificación **ITU P.525** [42]. Más concretamente la fórmula que más interesante resulta es la recogida en el apartado 2.2 donde se abordan los enlaces punto a punto:

$$L_{bf} = 32,4 + 20 \log(f) + 20 \log(d) \quad (1)$$

L_{bf} : Pérdida básica de transmisión en el espacio libre (dB)

f: frecuencia (MHz)

d: distancia (km)

Esta fórmula nos permite calcular la pérdida básica de transmisión en el espacio libre a partir de la frecuencia y la distancia. Sin embargo, también podríamos obtener la distancia si supiésemos la pérdida básica o pudiésemos calcularlo de alguna manera. Aquí es dónde se encuentra la técnica usada por muchos sistemas para calcular distancias. Para explicarlo haré uso de la siguiente imagen.

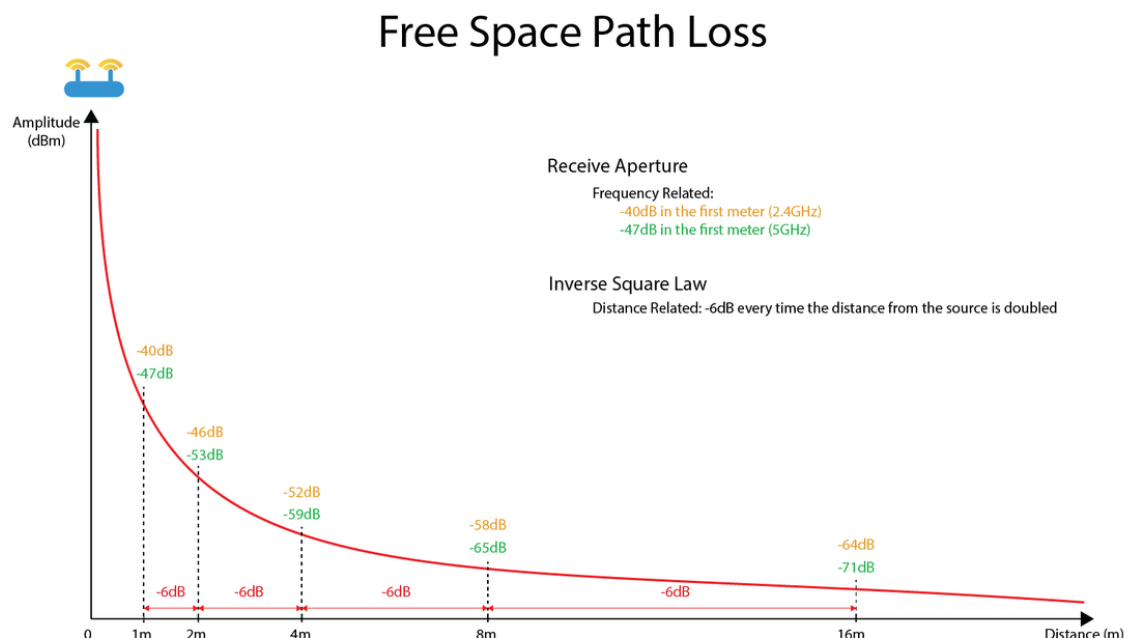


Figura 11: Ejemplo de amplitud frente a distancia para FSPL. Recuperada de cleartosend.net [43]

Como podemos ver, a medida que aumenta la distancia la atenuación va incrementando. Esto nos permite poner valor con cierto error para obtener el valor de la atenuación. Los rúters inalámbricos actuales incluyen antenas que soportan frecuencias de 2,4 y 5 GHz. Existen diferencias para el cálculo de la atenuación para cada frecuencia ya que trabajan con distintas longitudes de onda (12 y 6 m respectivamente). A priori esto muestra que el receptor de 2,4 GHz tiene mayor apertura de recepción y por lo tanto le permitiría obtener mejores resultados en cuanto a atenuación que el receptor de 5 GHz.

Para demostrar esto y calcular la atenuación media para cada receptor aplicaré la fórmula a todos los canales de cada frecuencia y deduciré la atenuación media para cada frecuencia.

Para 2,4 GHz con 1 m de distancia.

CANAL	FRECUENCIA (MHz)	ATENUACIÓN (dB)
1	2412	40,09755
2	2417	40,11553
3	2422	40,13348
4	2427	40,15140
5	2432	40,16927
6	2437	40,18711
7	2442	40,20491
8	2447	40,22268
9	2452	40,24041
10	2457	40,25810
11	2462	40,27576
12	2467	40,29338
13	2472	40,31097
	<i>Total</i>	522,66056

Tabla 2: Atenuación en frecuencias de 2,4 GHz a 1 m de distancia

$$\bar{x} = \frac{\sum_{i=1}^n \text{atenuación}_i}{n} = \frac{522,66056}{13} = 40,2046582839832 \text{ dB}$$

Ahora vamos con los cálculos para 5 GHz con 1 m de distancia.

CANAL	FRECUENCIA (MHz)	ATENUACIÓN (dB)
32	5160	46,70299
34	5170	46,71981
36	5180	46,73660
38	5190	46,75335
40	5200	46,77007
42	5210	46,78675
44	5220	46,80341
46	5230	46,82003
48	5240	46,83663
50	5250	46,85319
52	5260	46,86971
54	5270	46,88621
56	5280	46,90268
58	5290	46,91911
60	5300	46,93552
62	5310	46,95189
64	5320	46,96823
68	5340	47,00083
	<i>Total</i>	843,217008908991

Tabla 3: Atenuación en frecuencias de 5 GHz a 1 m de distancia

$$\bar{x} = \frac{\sum_{i=1}^n \text{atenuación}_i}{n} = \frac{843,217008908991}{68} = 46,8453893838328 \text{ dB}$$

Podemos concluir por lo tanto que usando 2,4 GHz obtenemos mayor precisión en las medidas ya que la atenuación es menor.

Nótese que en (1) se usa la constante 92,45, este valor viene determinado por las unidades que se estén empleando siendo por lo tanto este valor el correspondiente para trabajar con GHz y metros. Sin embargo, trabajar con estas unidades en este proyecto no tiene sentido así que se utiliza 27.55 [44]

$$L_{bf} = 27,55 + 20\log(f) + 20\log(d) \quad (2)$$

$$d = 10^{\frac{L_{bf} - 27,55 - 20\log(f)}{20}} \quad (3)$$

Sustituimos L_{bf} por 40.2 dB (valor medio de atenuación para frecuencias de 2.4 GHz)

$$d = 10^{0,6325 - \log(f)} \quad (4)$$

Hemos encontrado una fórmula que nos permite relacionar distancia con frecuencia. Hay que tener en cuenta que se han realizado una parte de asunciones que hacen que esta fórmula no sea del todo precisa como por ejemplo la existencia de paredes u otros obstáculos que pueden incrementar esta atenuación. No obstante, se realizará una validación posterior con pruebas reales para calcular el error promedio en la medida de distancias; este error en principio tendrá bastante margen ya que el objetivo de este trabajo no es determinar de forma precisa la ubicación de personas dentro de un edificio sino hacerlo de forma aproximada.

3.4.2 Recolección de información

Para poner en marcha el algoritmo anteriormente descrito es necesario tener un conjunto de datos con información sobre los puntos de acceso. De este modo se podría saber la ubicación específica de cada punto de acceso y determinar a la distancia que se encuentra un dispositivo de dicho punto.

Teniendo esto en cuenta, se realizó un prototipo como culminación a la primera iteración cuya finalidad era mostrar los datos de las redes wifi cercanas. El siguiente paso consistió en ir por todas las aulas y pasillos de la ETSIT para obtener la ubicación exacta de cada punto de acceso, así como su dirección física.



Figura 12: Aplicación para la recolección de datos. Recuperada de aplicación IndoorPositioning [45].

Estos datos se fueron anotando manualmente para posteriormente digitalizarlos y materializarlos en una base de datos. De esta manera se compusieron los datos usados para la primera base de datos que se iba a probar, **Firestore**.

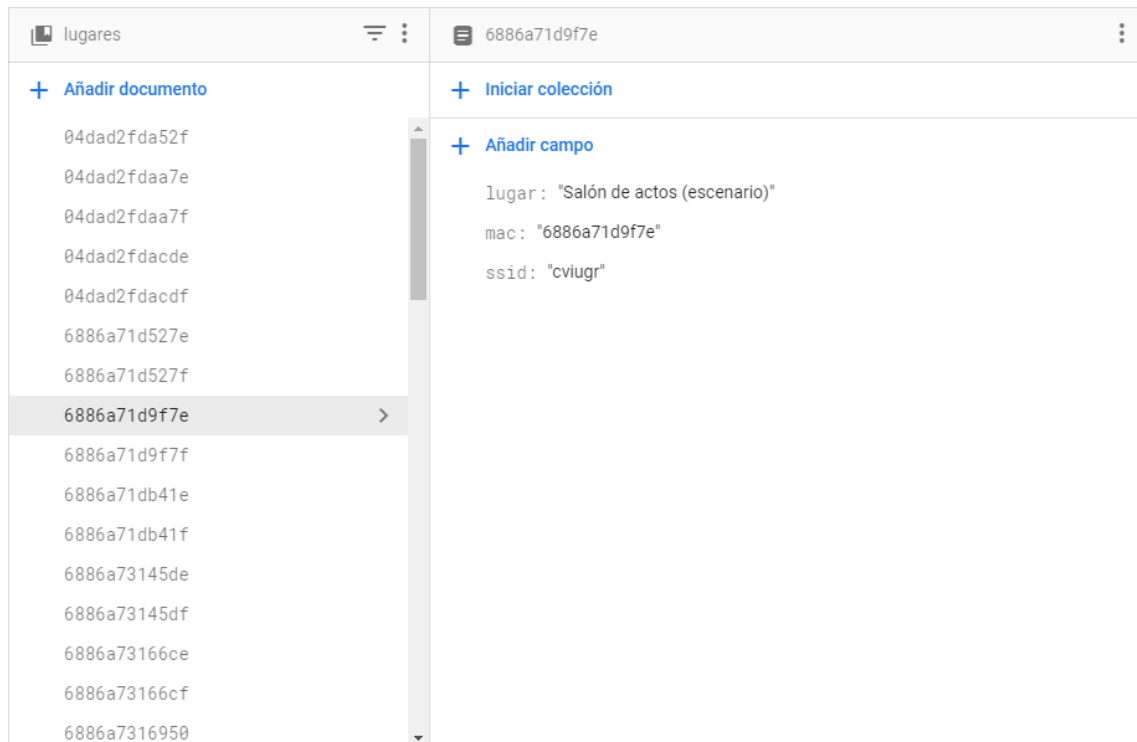


Figura 13: Base de datos con los puntos de acceso en Firebase. Recuperada de la BD en Firebase.

A continuación, se mostrarán todos los datos recopilados después de esta primera fase junto con otra fase realizada al final del proyecto para completar y verificar que los datos son correctos.

MAC	LOCALIZACIÓN
f0:7f:06:67:d5:ff	0.1
cc:d5:39:89:de:bf	0.4
cc:d5:39:89:e7:0f	0.7
68:86:a7:57:72:2f	1.6
68:86:a7:31:66:cf	1.8
68:86:a7:31:69:5f	Pasillo 1ª planta aulas
68:86:a7:1d:b4:1f	2.1
68:86:a7:57:ec:8f	2.4
cc:d5:39:89:e1:3f	2.8
68:86:a7:1d:b4:1f	Cuarto técnico
68:86:a7:57:73:af	3.10
68:86:a7:31:45:df	3.4
68:86:a7:57:cc:ff	3.7
68:86:a7:31:6a:ff	Delegación de estudiantes
84:b8:02:3b:31:8f	Vestíbulo A1-A2
68:86:a7:1d:52:7f	-1.1
cc:d5:39:89:96:1f	Cafetería
68:86:a7:1d:9f:7f	Salón de actos
b4:e9:b0:32:c6:8f	Hall
04:da:d2:fd:87:cf	Pasillo despachos dirección
b4:e9:b0:32:c8:3f	Sala de Reuniones
68:86:a7:1d:41:2f	Biblioteca
b4:e9:b0:32:dc:4f	Pasillo 1ª planta aulas de libre acceso
b4:e9:b0:32:bb:1f	Pasillo 2ª planta derecha ATC-TSTC cerca ascensor
04:da:d2:fd:aa:7f	Pasillo 2ª planta derecha TSTC-Álgebra
f0:29:29:0e:4d:ff	Pasillo 2ª planta izquierda ATC
04:da:d2:fd:a5:2f	Pasillo 3ª planta derecha LSI ascensor
b4:e9:b0:32:cf:4f	Pasillo 3ª planta derecha LSI aulas
f0:29:29:0e:45:ff	Pasillo 3ª planta izquierda LSI
b4:e9:b0:32:ce:7f	Pasillo 4ª planta derecha DECSAI ascensor
b4:e9:b0:32:c8:1f	Pasillo 4ª planta derecha DECSAI aulas
04:da:d2:fd:ac:df	Pasillo 4ª planta izquierda DECSAI
68:86:a7:cb:49:50	Despacho 5.1 TSTC
68:86:a7:cb:49:5f	Sala de Usos Múltiples

Tabla 4: Ubicación de routers inalámbricos identificados por su MAC. Elaboración propia.

El código correspondiente a este desarrollo se puede encontrar en un repositorio de GitHub con el nombre IndoorPositioning [46].

3.4.3 Desarrollo API propia

Para desarrollar la API propuesta para el segundo prototipo usé **Flask** por sencillez y rapidez de implementación. Esta API hace uso de **JWT** [47] como método de autenticación e incluye un tipo de usuario que realiza la función de administrador con posibilidad de realizar llamadas a endpoints ocultos para el resto de usuarios. Se debería incluir una función de refresco del token y así evitar realizar la operación de inicio de sesión cada x tiempo. Esta funcionalidad no se llegó a implementar debido a que no era necesaria para el prototipo funcional y tampoco se iba a continuar desarrollando el microservicio puesto que había otras prioridades.

El estado final de la API del microservicio es la siguiente:

MÉTODO	URI	PARÁMETROS	RESULTADO	DESCRIPCIÓN
POST	/login	username: String password: String		Permite al usuario logearse como administrador
GET	/user		User	Verifica la identidad
GET	/ap		ApList: List<AccessPoint>	Muestra todos los puntos de acceso en la base de datos
GET	/ap/<mac>	mac: String	Ap: AccessPoint	Muestra el punto de acceso con esa <mac>
POST	/ap	Ap: AccessPoint		Crea un nuevo punto de acceso
DELETE	/ap/<mac>	mac: String		Borra un punto de acceso existente con <mac>
PUT	/ap	Ap: AccessPoint	Ap: AccessPoint	Actualiza un punto de acceso existente

Tabla 5: API de MacLocation [48]. Elaboración propia.

Todas estas funciones trabajan sobre una base de datos no relacional alojada en un servicio en la nube de forma gratuita. La base de datos elegida fue MongoDB como ya se comentó anteriormente y para establecer la comunicación con ella se hizo uso de la librería **MongoClient**. La conexión se realiza de forma sumamente sencilla, en este caso se hizo uso de la URI como se indica en la documentación.

```
uri =  
"mongodb://user:password@example.com/?authSource=the_database&authMechanism=SCRAM-SHA-256"  
  
client = MongoClient(uri)
```

Por otro lado, un ejemplo de la consulta de un punto de acceso usando la MAC del mismo como identificador se haría de la siguiente manera.

```
# Get location of AP  
  
@app.route('/ap/<mac>', methods=['GET'])  
  
def getByMac(mac):  
    ap = db.find_one({'mac': mac})  
    return jsonify(ap), 200
```

Como se puede observar, se hace uso de la función *find_one* para buscar el elemento. Finalmente, se envía el elemento en formato JSON, para ello se hace uso de la función *jsonfy*. El resto del código de este proyecto se puede encontrar en el repositorio de GitHub MacLocation. [49]

3.4.4 Comunicación con SWAD

Para poder conocer la ubicación de un usuario será necesario tener en una base de datos el conjunto de puntos de acceso conocidos para posteriormente realizar consultas por clave primaria (dirección física) que revelen esta información. Por lo tanto, se ha decidido que la plataforma de SWAD sea la encargada de mantener el control sobre esta base de datos y proporcione una serie de métodos que permitan realizar operaciones de consulta a nivel de API. Cabe destacar que la plataforma permite tanto la adición como edición, así como borrado de ubicaciones o direcciones físicas. Esta funcionalidad está limitada a usuarios que sean administradores de centro y permite que en el futuro se puedan incluir salas o actualizar existentes.

Plataforma > España > ugr.es > ETSIT > Titulación > Asignatura

E. T. S. de Ingenierías Informática y de Telecomunicación

INICIO CENTRO EVALUACION ARCHIVOS USUARIOS MENSAJES ANALISIS PERFIL

Centro > Edificios
Edificios del centro educativo

Salas

Nueva sala

Cód. Edificio Planta Tipo Nombre breve Nombre completo Aforo Dirección MAC

Sin edifi 0 Sin tipo

Crear sala

	Cód. Edificio	Planta	Tipo	Nombre breve	Nombre completo	Aforo	Dirección MAC	
	30	Aulas	0	Aula	0.1	Aula 0.1 (Rector Lorenz)	85	f0:7f:06:67:d5
	31	Aulas	0	Aula	0.2	Aula 0.2	108	
	32	Aulas	0	Aula	0.3	Aula 0.3	108	
	33	Aulas	0	Aula	0.4	Aula 0.4	108	cc:d5:39:89:de
	34	Aulas	0	Aula	0.5	Aula 0.5	108	
	35	Aulas	0	Aula	0.6	Aula 0.6	108	
	36	Aulas	0	Aula	0.7	Aula 0.7	105	cc:d5:39:89:de
	37	Aulas	0	Aula	0.8	Aula 0.8	82	
	253	Aulas	0	Conserj	Conserjería planta baja	Conserjería planta baja	2	
	1	Aulas	1	Aula	1.1	Aula 1.1	60	
	2	Aulas	1	Aula	1.2	Aula 1.2	78	
	3	Aulas	1	Aula	1.3	Aula 1.3	78	

Figura 14: Creación y modificación de aulas en SWAD. Recuperada de SWAD [54]

En definitiva, toda esta información resulta fundamental para su representación en la aplicación y es por ello que se generaron unas funciones con las que interactuar, más adelante se profundizará en ellas. Concretando, la aplicación obtendrá la información de SWAD y realizará un procesamiento para determinar la distancia a cada punto de acceso haciendo uso del algoritmo anteriormente descrito.

Para comunicarse con las funciones de SWAD debemos conocer el estado de la API en primer lugar. La API de SWAD usa SOAP como protocolo de comunicación, esto supone ciertos requisitos que se deben cumplir a la hora de realizar la comunicación con la misma. Se tendrá que envolver cada petición con un encapsulado específico que se puede encontrar en la propia documentación de SWAD; más concretamente un fichero **WSDL** que facilita información sobre cómo se debe conformar la estructura del mensaje.

Para facilitar la comprensión de un fichero complejo WSDL hay varias alternativas, plugins para IDEs, herramientas (en su mayoría de pago), etc. Finalmente me decidí por usar una herramienta online a la que se le pasa la **url** y automáticamente te proporciona de forma visual el encapsulado de cada función de la API. Un ejemplo del envoltorio que se debe realizar sería algo tal que así:

```
<v:Envelope xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
xmlns:d="http://www.w3.org/2001/XMLSchema"
xmlns:c="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:v="http://schemas.xmlsoap.org/soap/envelope/">

  <v:Header />

  <v:Body>

    <n0:functionName xmlns:n0="urn:swad">

      <param>PARAM-VALUE</param>

    </n0:functionName>

  </v:Body>

</v:Envelope>
```

En *functionName* indicaremos el nombre de la función y en *param* el nombre de los parámetros necesarios, así como sus valores. Para realizar pruebas de estas llamadas a la API y cerciorarse de que todo funciona correctamente haré uso de la herramienta gratuita **Postman**.

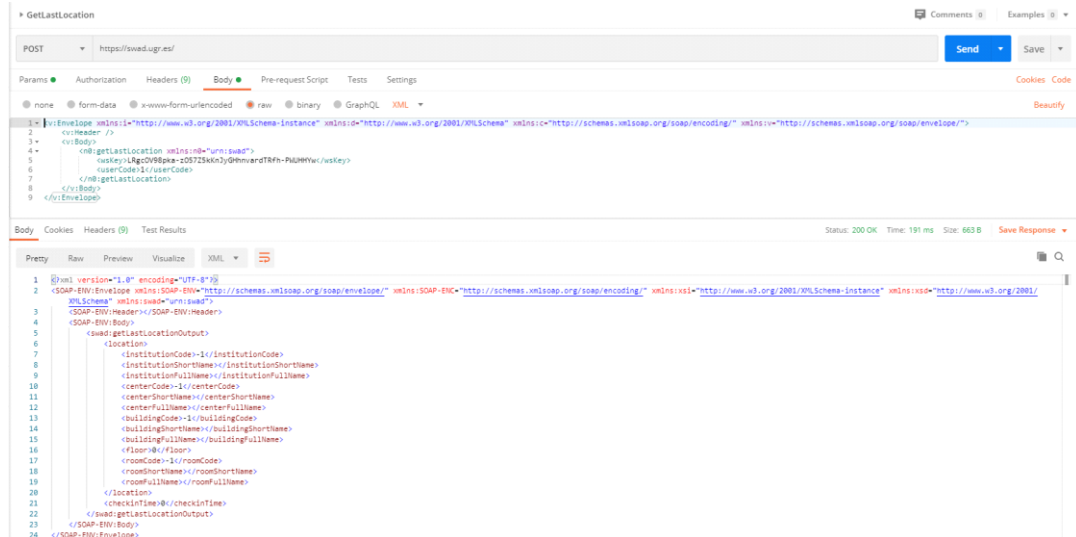


Figura 15: Ejemplo de llamada a una función de la API de SWAD. Recuperada de Postman [50]. Elaboración propia.

Como podemos observar se obtiene respuesta con la estructura de datos definida en la propia API. En este caso la función o endpoint es [GetLastLocation](#).

getLastLocation

Returns the more recent location (institution, center, building and room) of a given user. A user can only consult the location of another user if the intersection of the centers of their courses is not empty. Both users do not have to share any course, but at least some course of each one has to share center.

- Call parameters:
 - wsKey**: string, identifier returned by `loginByUserPasswordKey` or `loginBySessionKey`.
 - userCode**: unique identifier of the user returned, for example, by `findUsers`
- Returns a data structure with the following fields:
 - location**: a data structure, corresponding to a location, with the following fields:
 - institutionCode**: integer, unique identifier of the institution.
 - institutionShortName**: string.
 - institutionFullName**: string.
 - centerCode**: integer, unique identifier of the center.
 - centerShortName**: string.
 - centerFullName**: string.
 - buildingCode**: integer, unique identifier of the building.
 - buildingShortName**: string.
 - buildingFullName**: string.
 - floor**: signed integer.
 - roomCode**: integer, unique identifier of the room ($\leq 0 \Rightarrow$ the user has no registered her/his last location, or you don't have permission to see it; $> 0 \Rightarrow$ room found).
 - roomShortName**: string.
 - roomFullName**: string.
 - checkinTime**: integer of 32 bits with sign, [Unix time](#).

Figura 16: Función `getLastLocation` de la API de SWAD. Recuperada de SWAD API [50]

3.4.5 Diseño

Para llevar a cabo la implementación en SWADroid antes he realizado diagramas software que recogen cuál debería ser el funcionamiento del sistema para posteriormente implementarlo.

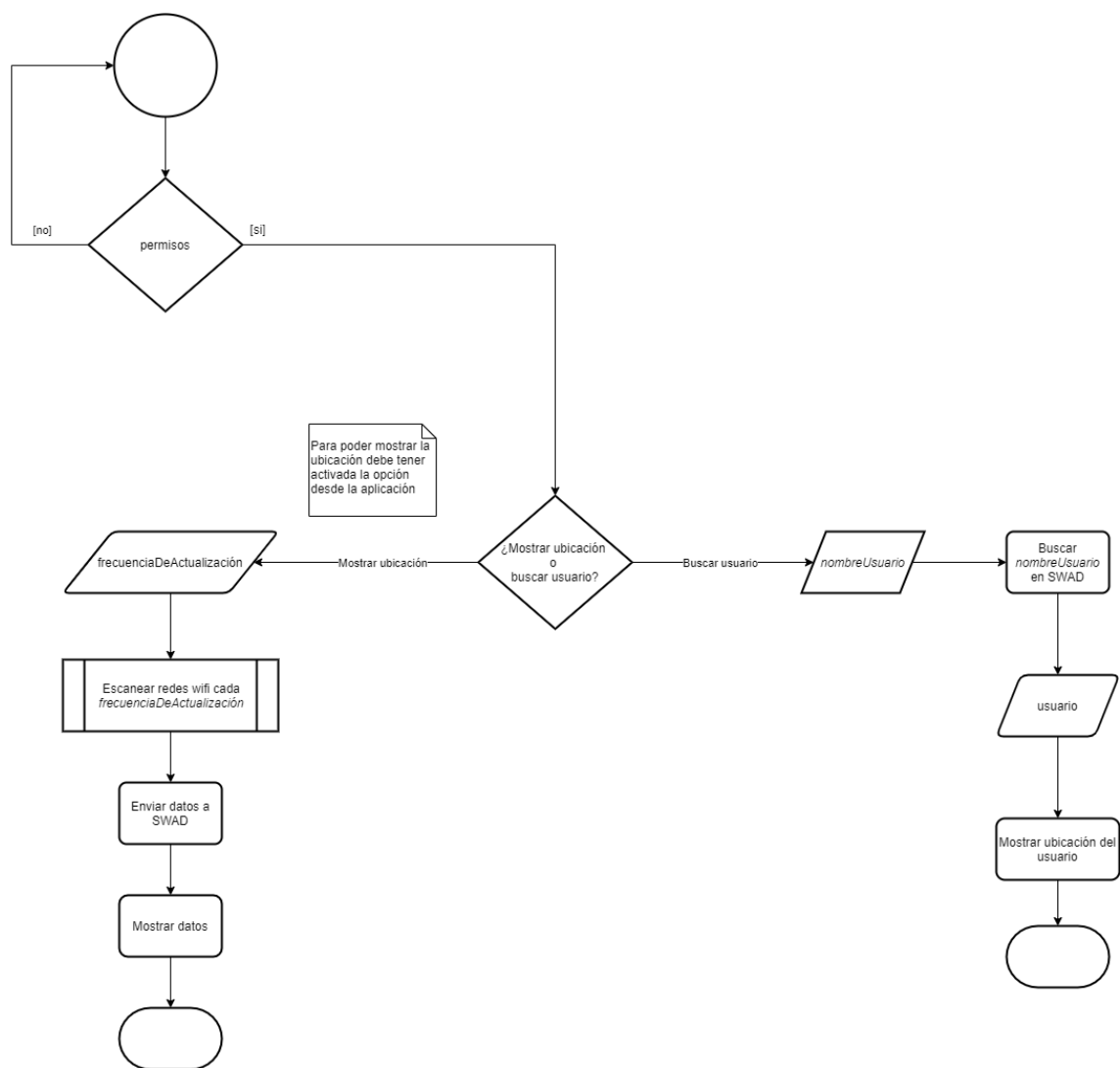


Figura 17: Diagrama de actividad. Realizado con DrawIO [55]. Elaboración propia.

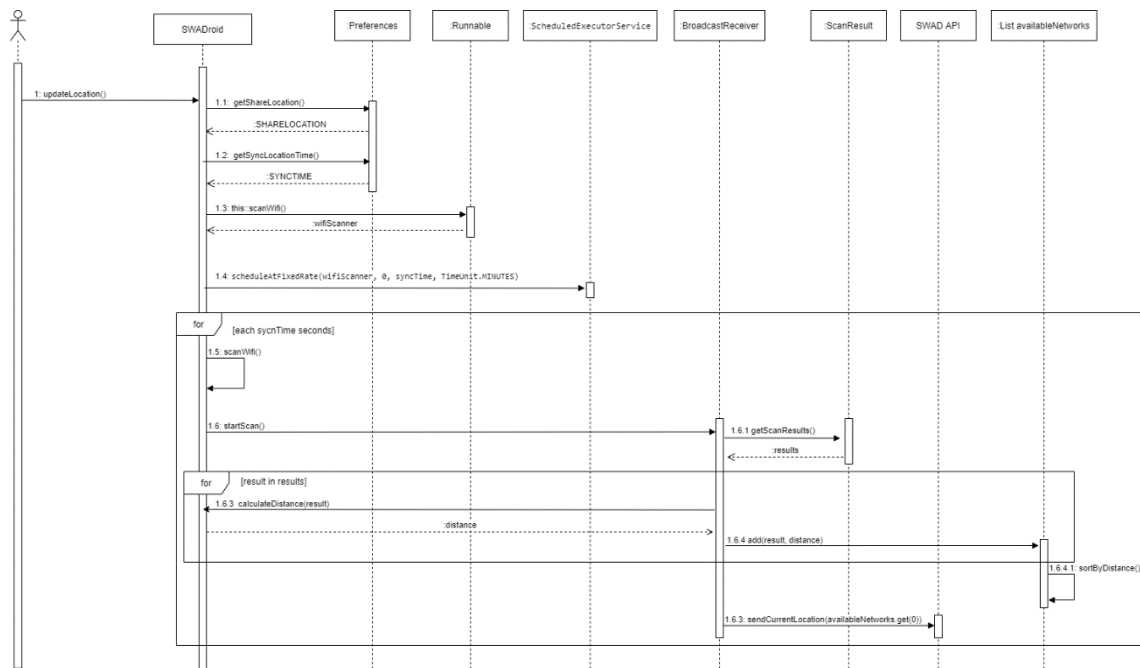


Figura 19: Diagrama de secuencia. UpdateLocation. Realizado con DrawIO [55] Elaboración propia.

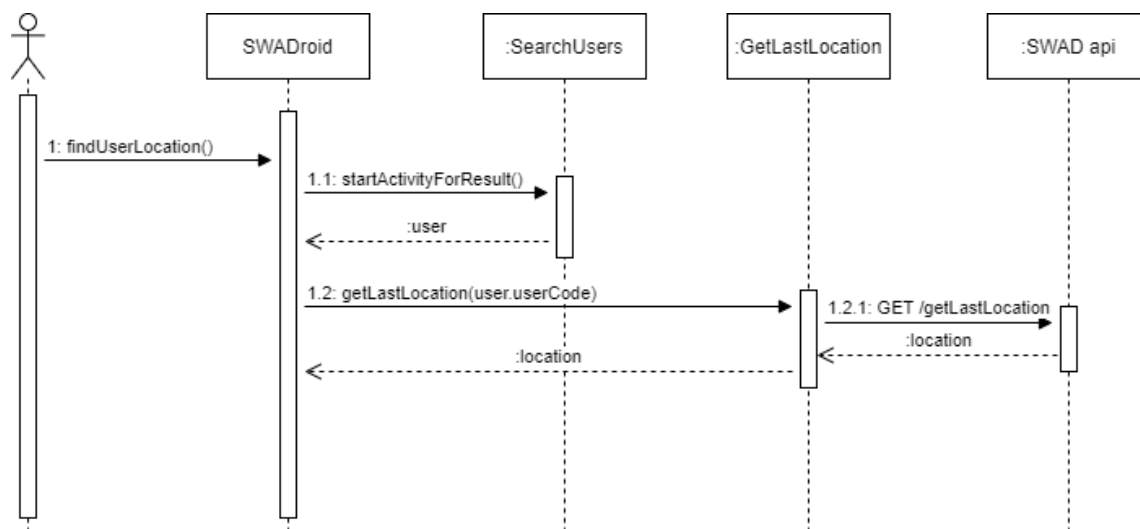


Figura 18: Diagrama de secuencia. FindUserLocation. Elaboración propia. [55]

Cabe destacar que el sistema solo hará uso de una única dirección de un punto de acceso para determinar la ubicación de un dispositivo dentro del edificio. La razón de esto radica en que establecer una posición intermedia (entre dos puntos de acceso o más) requiere que se haga uso de un algoritmo de triangulación de la posición mediante el uso de más rúters cercanos. En consecuencia, se añade una gran complejidad al sistema que además necesitaría de más utilidades como GPS para garantizar la ubicación precisa. Se ha tomado entonces la decisión de diseño que establece una única pareja individual ubicación-MAC. Por ello, la ubicación de posiciones intermedias será el objetivo de trabajos futuros que mejoren este sistema base de localización en interiores.

3.5 IMPLEMENTACIÓN

Para llevar a cabo la implementación de este módulo se tomaron como referencia los realizados anteriormente por otros estudiantes. De este modo la estructura empleada para el desarrollo del módulo constará de tres piezas fundamentales a nivel de diseño sobre las que se implementará la parte lógica descrita de forma teórica en este documento.

Estas tres piezas que componen el patrón de diseño de la aplicación son el **modelo**, las **vistas** y las **actividades**. Dentro de las actividades se pueden distinguir a su vez dos tipos: actividades de módulo y actividades empleadas para realizar peticiones a la API de SWAD. Resulta evidente pensar que esta disposición esté un poco desfasada pero actualmente sería necesaria una refactorización completa del código si se quisiese actualizar. No obstante, implementar módulos y otro tipo de funcionalidades resulta una tarea bastante sencilla y esto se debe en gran parte al esfuerzo dedicado por anteriores alumnos en preservar la estructura y seguir unas buenas prácticas de diseño. Por estos motivos se decidió continuar con este patrón que, si bien tiene inconvenientes, también presenta ventajas.

Como se comentó en el párrafo previo, la actividad principal que será en **núcleo** de este módulo se situará sobre el directorio *modules*. El nombre elegido es IndoorLocation y la elección de este nombre se debe a que es suficientemente descriptivo y además se diferencia de otra clase denominada Location empleada para otras finalidades distintas a la de la localización en interiores.

Esta clase IndoorLocation será la que implemente la lógica descrita en los diagramas de diseño anteriores. Su vista queda definida en el fichero *indoor_location.xml* (este fichero se encuentra dentro de *res* y a su vez *layout*).

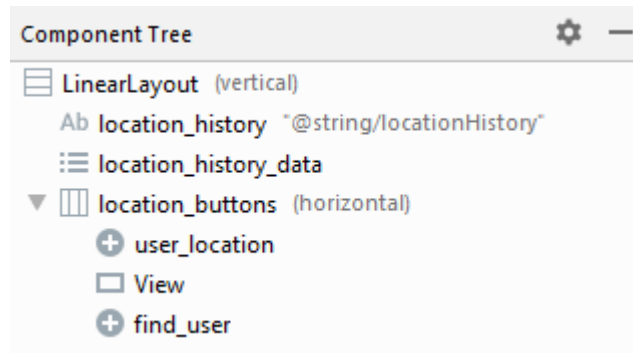


Figura 20: Árbol de components para fichero indoor_location.xml. Recuperada de indoor_location.xml

Como se puede apreciar en la imagen, existen dos botones. Estos dos botones serán los actuadores para lanzar los dos métodos requeridos: Compartir la ubicación del usuario y mostrar la ubicación de otro usuario.

Primeramente, compartir la ubicación del usuario. Esta funcionalidad implica que un usuario que lo desee decida compartir su ubicación para que sea consultada por otros usuarios de SWADroid. Este requerimiento obliga entonces a que se indique en el *manifest* de la aplicación que esta solicitará **permisos de ubicación**. Más concretamente se añadirán dos permisos para que el sistema funcione correctamente.

```
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />

<uses-permission
android:name="android.permission.ACCESS_BACKGROUND_LOCATION" />
```

ACCESS_FINE_LOCATION: Permiso para establecer la ubicación de forma precisa. Hay otra alternativa que sería ACCESS_COARSE_LOCATION que se usa cuando sólo se quiere establecer la ubicación usando wifi y datos móviles. Como en el futuro hay previsión de mejorar el sistema se ha elegido FINE_LOCATION.

ACCESS_BACKGROUND_LOCATION: Este permiso sirve para que aplicaciones en segundo plano hagan uso de la localización. En este módulo se enviará la ubicación siempre que el usuario lo permita así que también habrá situaciones en las que se deberá enviar, aunque la aplicación esté en segundo plano.

Además de estos permisos que van implícitos con la instalación de la aplicación se añade una opción al sistema de preferencias. De este modo el usuario podrá elegir si desea compartir su ubicación o no. Para implementar esto me he basado en el resto de preferencias que se han empleado para el desarrollo de SWADroid.

Esto consiste en definir una variable dentro de la clase *Preferences*. El valor de esta variable se encuentra definido en un documento *xml* en el que se almacena el texto que aparecerá cuando se muestren el conjunto de preferencias en la vista correspondiente. Este valor se debe incluir en dos ficheros, la razón de esto es que SWADroid incluye traducciones que permiten que la aplicación se pueda usar tanto en inglés como en español. Estos ficheros de descripción se denominan *strings.xml* y se encuentran dentro de */res/values/strings*.

```
<string name="locationHistory">Historial de localizaciones</string>
<string name="update">Actualizar</string>
<string name="prefSyncLocationTimeTitle">Ubicación</string>
<string name="prefShareLocation">Habilitar ubicación</string>
<string name="prefSyncLocationTimeSummary">Frecuencia de sincronización para localización</string>
<string name="prefSyncLocationTitle">Ubicación</string>
<string name="locationDisabled">Necesitas activar la ubicación</string>
<string name="userLocation">Ubicación de</string>
</resources>
```

Figura 21: Traducciones a español del fichero *strings.xml*. Recuperada del fichero */res/values/string*.

```
/**
 * Synchronization time preference for location
 */
public static final String SYNCLOCATIONTIMEPREF = "prefSyncLocationTime";
/**
 * Sharing location preference
 */
public static String SHARELOCATION = "prefShareLocation";
```

Figura 22: Declaración de preferencias. Recuperada del fichero *Preferences.java*.

Otra preferencia que incluyo es la frecuencia de refresco de la ubicación, es decir, enviar la ubicación del usuario a SWAD para que esta sea consultada por otros usuarios. En consecuencia, con una frecuencia alta podemos detectar si una persona se está moviendo o se encuentra en un aula, por ejemplo. Si elegimos una baja frecuencia de refresco se optimizará el uso de recursos del sistema Android.

Para definir esta preferencia se hace uso del fichero *list.xml* que define al igual que *strings.xml* una serie de variables traducidas.

```
<string-array name="prefSyncLocationTimeEntries">
    <item>1 minute</item>
    <item>2 minutes</item>
    <item>5 minutes</item>
    <item>15 minutes</item>
    <item>30 minutes</item>
    <item>1 hour</item>
</string-array>
<string-array name="prefSyncLocationTimeValues">
    <item>1</item>
    <item>2</item>
    <item>5</item>
    <item>15</item>
    <item>30</item>
    <item>60</item>
</string-array>
```

Figura 23: Preferencias de refresco de ubicación. Recuperada del fichero *strings.xml*

Se prosigue ahora con el desarrollo de la actividad principal que contiene la lógica del módulo. Para simplificar la explicación se hará por partes, cada una contendrá la información relativa a cada botón de los descritos anteriormente.

3.5.1 Compartir ubicación

Corresponde al botón de la izquierda y al pulsarlo se realizarán llamadas periódicas al servidor de SWAD con la actualización de la ubicación del usuario. Si el usuario deshabilita la preferencia de la ubicación, estas llamadas se detendrán.

Para manejar el evento del click se usa un **Listener**. Para ello es necesario declarar una variable del tipo del botón que para este diseño será un *FloatingActionButton* concretamente y se llamará a la función *setOnClickListener* que establece qué se debe hacer cuando se pulse el botón.

Para implementar todo esto existen dos opciones: usar una **expresión lambda** o no, personalmente creo que el código es más legible con una expresión lambda cuando se está trabajando con listeners, por lo tanto, se realizará la codificación de esa manera.

```

111 FloatingActionButton updateLocation = findViewById(R.id.user_location);
112 updateLocation.setOnClickListener(v -> {
113     if(Preferences.getShareLocation()) {
114         if (userSearched)
115             locationHistory.clear();
116
117         textView.setText(getString(R.string.locationHistory));
118         int syncTime = Integer.parseInt(Preferences.getSyncLocationTime());
119
120         Runnable wifiScanner = this::scanWifi;
121         try {
122             scheduler.scheduleAtFixedRate(wifiScanner, 0, syncTime, TimeUnit.MINUTES);
123         } catch (IllegalArgumentException e){
124             e.printStackTrace();
125         }
126     } else {
127         scheduler.shutdown();
128         Toast.makeText(getApplicationContext(), getResources().getString(R.string.locationDisabled), Toast.LENGTH_LONG).show();
129     }
130 });
131 }

```

Figura 24: Recuperada de SetOnClickListener updateLocation en la clase IndoorLocation

En la línea 118 se obtiene el tiempo de actualización que se empleará.

En la línea 120 se crea un objeto Runnable con scanWifi. Esto se explicará más adelante porque es una de las partes más complejas de analizar.

En la línea 122 se establece la frecuencia de actualización.

En el else se realiza la detención de las tareas programadas de actualización cuando el usuario deshabilita la preferencia.

En la línea 122 se emplea una referencia al método que será Runnable. Este método se encargará de **escanear las redes wifi** cercanas, filtrarlas por SSID, calcular la distancia aproximada a cada una de ellas y ordenarlas por este último parámetro. Además, realizará dos llamadas a la API de SWAD. La primera será para obtener la información necesaria sobre la ubicación, para ello el conjunto de localizaciones de la ETSIT identificadas por su dirección física se encuentran en una base de datos de SWAD. Esta petición no es más que una consulta por clave primaria a esta base de datos. Más en concreto al método GetLocation, para ello primero hay que observar la descripción del método en la API.

getLocation

Returns a location (institution, center, building and room) for Wi-Fi-based positioning system given a MAC address.

- Call parameters:
 - **wsKey**: string, identifier returned by `loginByUserPasswordKey` or `loginBySessionKey`.
 - **MAC**: string in XXXXXXXXXXXX format (12 hexadecimal digits)
- Returns a data structure with the following fields:
 - **location**: a data structure, corresponding to a location, with the following fields:
 - **institutionCode**: integer, unique identifier of the institution.
 - **institutionShortName**: string.
 - **institutionFullName**: string.
 - **centerCode**: integer, unique identifier of the center.
 - **centerShortName**: string.
 - **centerFullName**: string.
 - **buildingCode**: integer, unique identifier of the building.
 - **buildingShortName**: string.
 - **buildingFullName**: string.
 - **floor**: signed integer.
 - **roomCode**: integer, unique identifier of the room ($\leq 0 \Rightarrow$ no room found for this MAC; $> 0 \Rightarrow$ room found).
 - **roomShortName**: string.
 - **roomFullName**: string.

Figura 25: Método GetLocation de la API de SWAD

Se puede comprobar entonces que este método devuelve una estructura del tipo Location. Esta estructura será necesario definirla como un modelo en la aplicación. Para comprobar el correcto funcionamiento de este endpoint se realizará una llamada desde Postman con la que poder comprobar que el resultado es el esperado.

Como parámetros de llamada se deben especificar el *wsKey* y una *MAC*. Para obtener el *wsKey* de forma rápida se puede depurar la aplicación con Android Studio y añadir un breakpoint dentro de una clase destinada a realizar una llamada a un endpoint de SWAD; de esta manera será posible copiar el valor de la *wsKey* sin mucha complejidad. Como *MAC* usaré una aleatoria de las que existen en la base de datos que se mostró anteriormente.

```

<?xml version="1.0" encoding="UTF-8"?>

<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:swad="urn:swad">

  <SOAP-ENV:Header></SOAP-ENV:Header>

  <SOAP-ENV:Body>

    <swad:getLocationOutput>

      <location>

        <institutionCode>1</institutionCode>

        <institutionShortName>ugr.es</institutionShortName>

        <institutionFullName>Universidad de
Granada</institutionFullName>

        <centerCode>2</centerCode>

        <centerShortName>ETSIIT</centerShortName>

        <centerFullName>E. T. S. de Ingenierías Informática y
de Telecomunicación</centerFullName>

        <buildingCode>2</buildingCode>

        <buildingShortName>Aulas</buildingShortName>

        <buildingFullName>Edificio de aulas</buildingFullName>

        <floor>1</floor>

        <roomCode>1</roomCode>

        <roomShortName>1.1</roomShortName>

        <roomFullName>Aula 1.1</roomFullName>

      </location>

    </swad:getLocationOutput>

  </SOAP-ENV:Body>

</SOAP-ENV:Envelope>

```

Como conclusión, se puede verificar que los datos son los esperados y que la función actúa correctamente. Ahora se pasará a su implementación.

Para realizar peticiones a SWAD desde Android será necesario basarse en la estructura que se emplea en el resto de módulos para conservar la estructura de creación. Lo cual implica, crear una clase con el mismo nombre de la función de la API que se implementa y que a su vez herede de la clase abstracta **Module**. Siendo de esta manera necesario implementar los métodos que se declaran en la clase padre sin ser implementados. A continuación, se describirán estos métodos, así como la finalidad de cada uno de ellos dentro del proceso de comunicación con la API.

- **connect**: Este método lanza la ejecución de la tarea implementada por el módulo en una hebra en segundo plano.
- **postConnect**: Este método ejecuta todas aquellas acciones que deben realizarse cuando acabe la ejecución de la tarea principal del módulo como el refresco de la pantalla, por ejemplo.
- **requestService**: Este método es el más complejo ya que implementa la tarea principal que se llevará a cabo en el módulo. Normalmente esta tarea consistirá en construir y enviar una petición a un servicio web de SWAD y procesar la información recibida.

```
@Override
protected void requestService() throws Exception {
    createRequest(SOAPClient.CLIENT_TYPE);
    addParam( param: "wsKey", Login.getLoggedUser().getWsKey());
    addParam( param: "MAC", mac);
    sendRequest(Location.class, simple: true);

    if (result!=null) {...}
    Intent intent = new Intent();
    intent.putExtra( name: "location", location);
    intent.putExtra( name: "distance", distance);
    setResult(RESULT_OK, intent);
}
```

Figura 26: Función request service de la clase GetLocation. Recuperada de IndoorLocation.java

Como se puede observar las primeras líneas van destinadas a realizar la petición. Después de enviar la petición se obtiene el resultado *result* en formato **SOAP**. Para poder mapear los campos del objeto en formato SOAP a una clase definida previamente en el modelo es necesario usar las **properties**. Finalmente, como forma de comunicación entre Actividades se pueden usar los Intents como ya se ha explicado anteriormente. En este caso se pasa un objeto Location con todos los campos actualizado una vez se han ido obteniendo del SoapObject de la respuesta.

```

SoapObject soap = (SoapObject) result;
soap = (SoapObject)soap.getProperty(0);

int institutionCode = Integer.parseInt(soap.getProperty("institutionCode").toString());
String institutionShortName = soap.getProperty("institutionShortName").toString();
String institutionFullName = soap.getProperty("institutionShortName").toString();
int centerCode = Integer.parseInt(soap.getProperty("centerCode").toString());
String centerShortName = soap.getProperty("centerShortName").toString();
String centerFullName = soap.getProperty("centerFullName").toString();
int buildingCode = Integer.parseInt(soap.getProperty("buildingCode").toString());
String buildingShortName = soap.getProperty("buildingShortName").toString();
String buildingFullName = soap.getProperty("buildingFullName").toString();
int floor = Integer.parseInt(soap.getProperty("floor").toString());
int roomCode = Integer.parseInt(soap.getProperty("roomCode").toString());
String roomShortName = soap.getProperty("roomShortName").toString();
String roomFullName = soap.getProperty("roomFullName").toString();

```

Figura 27: Obtención de campos de un SoapObject en la clase GetLocation. Recuperada de GetLocation.java

3.5.2 Buscar usuario

Corresponde al botón de la derecha y al pulsarlo se realizarán llamadas periódicas al servidor de SWAD para la obtención de la ubicación del usuario elegido.

Para manejar el evento del click hago uso de un Listener como he explicado anteriormente. Quedaría de la siguiente manera:

```

FloatingActionButton findUser = findViewById(R.id.find_user);
findUser.setOnClickListener(view -> {
    Log.d(TAG, msg: "onClick: Find user location");
    Intent intent = new Intent(getApplicationContext(), SearchUsers.class);
    intent.putExtra( name: "receivers", arrayReceivers);
    startActivityForResult(intent, Constants.SEARCH_USERS_REQUEST_CODE);
});

```

Figura 28: Listener Find user location

Como se puede observar el procedimiento es muy sencillo. La clase SearchUsers que se usaba anteriormente para enviar mensajes de texto por SWAD ahora se usará para buscar usuarios que compartan su ubicación. A la hora de seleccionar el destinatario incluía una búsqueda que permitía seleccionar incluso el ámbito en el que se debería de buscar. Esto es justamente lo que se necesita para realizar la búsqueda de usuarios dentro del módulo IndoorLocation.



Figura 29: Buscar usuario usando SearchClass.

Recuperada de SWADroid.

Cuando finaliza la tarea de compartir usuario o la de buscar usuario se produce una llamada de Android a la función *onActivityResult*. De este modo se llama a otra actividad y se espera el resultado desde la clase llamadora. Este se convertirá en el procedimiento siempre que se necesite llamar a otra actividad y haya que esperar algún resultado.

```

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    switch (requestCode) {
        case Constants.SEARCH_USERS_REQUEST_CODE:
            if (data != null) {...}
            break;
        case Constants.GET_LOCATION:
            if (data != null) {...}
            break;
        case Constants.GET_LAST_LOCATION:
            if (data != null){...}
            break;
        case Constants.SEND_CURRENT_LOCATION:
            if (data != null){...}
            break;
    }
    super.onActivityResult(requestCode, resultCode, data);
}

```

Figura 30: Función onActivityResult. Recuperada de IndoorLocation.java.

Para identificar la procedencia de cada finalización se emplean constantes que quedan definidas en un archivo llamado *Constants.java*. El motivo de identificar con valores constantes la finalización de una tarea es para ser capaces desde la clase que se llama a la otra actividad de controlar los datos que se obtienen de la finalización de esa actividad.

```

/**
 * Request code for search users
 */
public static final int SEARCH_USERS_REQUEST_CODE = 40;
/**
 * Request code for Manage Location
 */
public static final int MANAGE_LOCATION = 41;
/**
 * Request code for Find User
 */
public static final int FIND_USER = 42;
/**
 * Request code for Get Location
 */
public static final int GET_LOCATION = 43;
/**
 * Request code for Get Last Location
 */
public static final int GET_LAST_LOCATION = 44;
/**
 * Request code for Send Current Location
 */
public static final int SEND_CURRENT_LOCATION = 45;

```

Figura 31: Fragmento de Constants.java. Elaboración propia.

Anteriormente, se comentó que era necesario realizar una deserialización del objeto que devuelve la API de SWAD, pero no es suficiente con esto. Estos objetos deserializados deben usarse para la creación de un objeto definido en el modelo de datos como *Location*, por ejemplo. Siendo así, será posible realizar una llamada a otra actividad con este nuevo objeto que ha sido creado. Para comunicarse con la actividad principal se hará uso de Intents del mismo modo que se ha hecho hasta ahora. Todo esto da explicación entonces a la siguiente figura en la que se obtienen los objetos que devuelve la clase de búsqueda de usuarios.

```
case Constants.SEARCH_USERS_REQUEST_CODE:
    if (data != null) {
        arrayReceivers = data.getParcelableArrayListExtra( name: "receivers");
        if(arrayReceivers != null && arrayReceivers.size() > 0){
            UserFilter user = ((UserFilter)arrayReceivers.get(0));
            String userText = "Location" + " " + user.getUserFirstname() + " "
                + user.getUserSurname1();
            textView.setText(userText);
            Intent getLastLocation = new Intent(getApplicationContext(), GetLastLocation.class);
            getLastLocation.putExtra( name: "userCode", user.getUserCode());
            startActivityForResult(getLastLocation, Constants.GET_LAST_LOCATION);
        }
    }
    break;
```

Figura 32: Search users result. Captura tomada de la clase *IndoorLocation* dentro del método *onActivityResult*. Elaboración propia.

Para obtener el objeto que se pasa desde la clase que realiza la petición se usa *getParcelableArrayListExtra*. Un objeto **Parcelable** no es más que una forma de envolver a otro objeto dentro de una estructura para que posteriormente pueda ser tratado de forma correcta.

Por otro lado, además de obtener los datos del usuario que se ha buscado, se realiza una nueva llamada a otra Actividad. La actividad a la que se está llamando es *GetLastLocation*, esto se debe a que para llamar al método *GetLastLocation* es necesario tener previamente el **userCode** que es una clave primaria empleada para realizar búsquedas de obtención de la ubicación.

getLastLocation

Returns the more recent location (institution, center, building and room) of a given user. A user can only consult the location of another user if the intersection of the centers of their courses is not empty. Both users do not have to share any course, but at least some course of each one has to share center.

- Call parameters:
 - **wsKey**: string, identifier returned by `loginByUserPasswordKey` or `loginBySessionKey`.
 - **userCode**: unique identifier of the user returned, for example, by `findUsers`
- Returns a data structure with the following fields:
 - **location**: a data structure, corresponding to a location, with the following fields:
 - **institutionCode**: integer, unique identifier of the institution.
 - **institutionShortName**: string.
 - **institutionFullName**: string.
 - **centerCode**: integer, unique identifier of the center.
 - **centerShortName**: string.
 - **centerFullName**: string.
 - **buildingCode**: integer, unique identifier of the building.
 - **buildingShortName**: string.
 - **buildingFullName**: string.
 - **floor**: signed integer.
 - **roomCode**: integer, unique identifier of the room ($\leq 0 \Rightarrow$ the user has no registered her/his last location, or you don't have permission to see it; $> 0 \Rightarrow$ room found).
 - **roomShortName**: string.
 - **roomFullName**: string.
 - **checkinTime**: integer of 32 bits with sign, `Unix time`.

Figura 33: `getLastLocation` endpoint. Recuperada de la API de SWAD [50]

Cuando finalice `getLastLocation` se volverá a entrar en la sentencia `switch case` y se ejecutará la parte correspondiente a `getLastLocation`.

```
case Constants.GET_LAST_LOCATION:
    if (data != null){
        LocationTimeStamp locationTimeStamp = (LocationTimeStamp) data.getSerializableExtra( name: "locationTimeStamp");
        assert locationTimeStamp != null;
        locationHistory.add(locationTimeStamp.getRoomFullName() + " ( " +
            locationTimeStamp.getCenterShortName() + " " +
            locationTimeStamp.getInstitutionShortName() + " )");
        locationHistoryAdapter.notifyDataSetChanged();
    }
    break;
```

Figura 34: `getLastLocation` result. Captura tomada de la clase `IndoorLocation` dentro del método `onActivityResult`. Elaboración propia.

En este fragmento lo que se realiza es la obtención de la localización y se añade a un array que es el que se muestra en la vista (**locationHistory**).

Si se navega por el código se encontrará que dentro de este `switch` existen otros dos casos. Ambos son para la primera funcionalidad explicada, el primero actúa de forma similar a `GET_LAST_LOCATION`. El segundo simplemente comprueba que la petición ha tenido éxito.

3.5.3 Base de datos

En este módulo se pueden diferenciar dos tablas dentro de una misma base de datos. La primera sería la encargada de almacenar todos aquellos puntos de acceso de los centros y escuelas que deseen incorporar esta funcionalidad. La segunda es una tabla con la relación ubicación-usuario.

La primera tabla tendrá almacenadas las direcciones físicas de los rúters y los códigos de cada localización. Se ha tomado esta decisión ya que en SWAD existía previamente una tabla que agrupaba a todas las localizaciones, de este modo, se hace uso del **roomCode** como clave externa a esta otra tabla.

DESCRIPCIÓN		
PRIMARY KEY	MAC	Dirección física de un router
FOREIGN KEY	RooCod	Identificador único de una sala

Tabla 6: Tabla room_MAC en SWAD. Recuperada de SWAD.

DESCRIPCIÓN		
PRIMARY KEY	RooCod	Identificador único de una sala
	CtrCod	Identificador único de un centro
	BldCod	Identificador único de un edificio
	Floor	Número de la planta
	Type	Tipo (pasillo, aula, comedor...)
	ShortName	Nombre abreviado de la sala
	FullName	Nombre completo de la sala
	Capacity	Aforo de la sala

Tabla 7: Tabla rooms en SWAD. Recuperada de SWAD.

La segunda tabla será la encargada de establecer la relación entre un usuario y una ubicación. Para ello hará uso de claves foráneas que establezcan una relación con las tablas anteriormente descritas.

DESCRIPCIÓN		
PRIMARY KEY	ChkCod	Identificador único de la tabla check_in
FOREIGN KEY	UsrCod	Identificador único de usuario
FOREIGN KEY	RooCod	Identificador único de una sala
	CheckInTime	Hora de envío de ubicación

Tabla 8: Tabla room_check_in en SWAD. Recuperada de SWAD.

3.5.4 Algoritmo

Anteriormente se ha descrito el funcionamiento del algoritmo a nivel teórico. A continuación, se explicará la implementación que se ha dado, así como las posibles mejoras.

Cuando se comienza a compartir ubicación se establece la distancia a cada punto de acceso para así tener un conjunto que poder ordenar en función a la distancia. Este cálculo se lleva a cabo haciendo una llamada a una función que implementa la lógica de cálculo de distancia.

```
public double calculateDistance(double levelInDb, double freqInMHz) {
    double FREE_SPACE_PATH_LOSS_CONSTANT = 27.55;
    double exp = (FREE_SPACE_PATH_LOSS_CONSTANT - (20 * Math.Log10(freqInMHz)) + Math.abs(levelInDb)) / 20.0;
    return Math.pow(10.0, exp);
}
```

Figura 35: Implementación de cálculo de distancia. Recuperada de IndoorLocation.java.

El algoritmo es bastante sencillo, recibe como parámetros el nivel en decibelios y la frecuencia en mega Hercios. Estos dos valores se obtienen de los objetos *ScanResult* que son los obtenidos al usar un **BroadcastReceiver**. Su implementación es llevada a cabo haciendo uso de una clase anónima en Java. Las clases anónimas en Java son útiles sobre todo cuando se quiere sobrecargar un método de dicha clase como en este caso es el método onReceive.

```

BroadcastReceiver wifiReceiver = (context, intent) -> {
    List<ScanResult> results = wifiManager.getScanResults();
    unregisterReceiver(this);
    if (results.size() > 0) {
        for (ScanResult scanResult : results) {
            int distance = (int) calculateDistance(scanResult.level, scanResult.frequency);
            availableNetworks.add(new Pair<>(scanResult, distance));
        }
        Collections.sort(availableNetworks, (n1,n2) -> n2.second - n1.second);
        Intent getLocation = new Intent(context, GetLocation.class);
        getLocation.putExtra( name: "mac", availableNetworks.get(0).first.BSSID.replace( target: ":", replacement: ""));
        startActivityForResult(getLocation, Constants.GET_LOCATION);
    }
};

```

Figura 36: Implementación del escanner de Wifi. Recuperada de Indoorlocation.java

Resumiendo, se sobrescribe el método **onReceive** que es llamado cuando se finaliza el escaneo de las redes Wifi. Se obtiene un conjunto de redes que se almacenan en una lista de *ScanResult*. Posteriormente se recorre esta lista calculando la distancia a cada punto de acceso y la red junto a la distancia se va almacenando en otra lista. Se almacena en otra lista porque cuando acaben todas las iteraciones se procede a realizar un proceso de ordenación. Con el conjunto ordenado, se van realizando peticiones a la API de SWAD para comprobar que alguno de esos puntos existe y obtener así la ubicación. De esta forma se garantiza que se tomará siempre la ubicación más cercana en caso de haber más de un punto de acceso cercano. Dentro del método **onReceive** se hace una llama a *GetLocation*, el resto de llamadas se hacen desde el método **onActivityResult** que ya se explicó en secciones anteriores.

4 MANUAL DE USUARIO

4.1 INTRODUCCIÓN

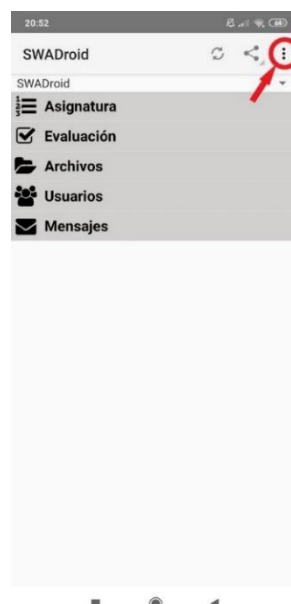
Este manual se ha creado para facilitar el uso del módulo de localización en interiores dentro de la aplicación de SWADroid. A continuación, se detallarán las nuevas funciones añadidas, así como el sentido y utilidad de cada uno de los elementos de diseño que se han incorporado tras la actualización de la aplicación con este módulo.

4.2 FUNCIONES

4.2.1 Habilitar ubicación

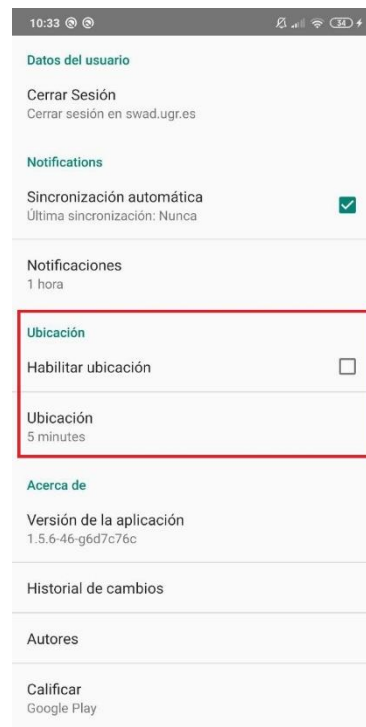
En primer lugar, hay que tener la ubicación activada en el dispositivo Android. En segundo, dar permisos de ubicación a la aplicación. Estos permisos se solicitarán a la hora de instalar la aplicación desde PlayStore. En algunos dispositivos será necesario realizar esta asignación de permisos de forma manual. Para ello ir a “ajustes” del dispositivo Android, ir a “administración de aplicaciones”, seleccionar SWADroid y seleccionar en “gestión de permisos”. Finalmente marcar el permiso correspondiente a la ubicación.

Teniendo estos permisos activos hay que habilitar la ubicación en la aplicación. Para ello, una vez ingrese en el cliente SWADroid ir a ajustes.



*Figura 37: Ajustes de SWADroid.
Recuperada de SWADroid.*

Una vez en estos ajustes aparecerá una ventana con toda la configuración de la aplicación. Ir a la parte que pone ubicación y activar la opción, adicionalmente se podrá establecer la frecuencia con la que SWADroid actualiza la ubicación con el servidor.



*Figura 38: Configuración SWADroid.
Recuperada de SWADroid.*

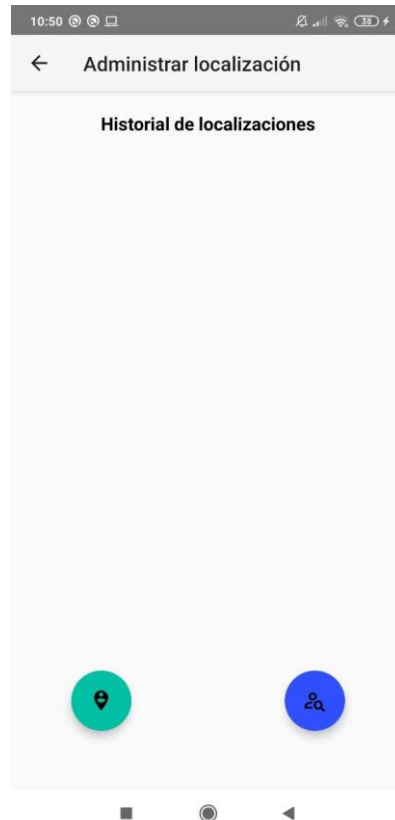
4.2.2 Compartir ubicación

Para empezar a enviar la ubicación ir al menú principal -> Usuarios -> Ubicación. Se abrirá un menú como el de la figura 40.



*Figura 39: Opción de localización dentro del menú Usuarios.
Recuperada de SWADroid.*

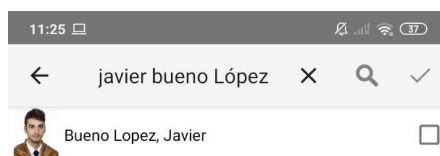
Si se pulsa el botón verde de la izquierda se empezará a compartir la ubicación. Si no se ha realizado alguno de los pasos anteriores la aplicación le notificará con la acción que debe realizar para el correcto funcionamiento. Si todo está funcionando la ubicación en tiempo real aparecerá justo debajo del *Historial de localizaciones*.



*Figura 40: Menú de localización.
Recuperada de SWADroid.*

4.2.3 Buscar la ubicación de un usuario

Si se desea buscar la ubicación de un usuario se pulsará el botón azul y se abrirá una pantalla similar a la de la figura 41.



*Figura 41: Buscar ubicación de usuario.
Recuperada de SWADroid*

En el buscador superior es posible escribir el nombre de la persona que queremos buscar. Además, se pueden realizar búsquedas por asignatura ya que se habilita un cuadro de elección a la hora de clicar la lupa de búsqueda. Una vez realizado este procedimiento se volverá al menú de localización y aparecerá la última ubicación registrada del usuario seleccionado.

5 CONCLUSIONES Y TRABAJOS FUTUROS

La conclusión tras haber desarrollado este módulo para SWADroid es que se han satisfecho todos los requisitos que se estipularon al comienzo del trabajo.

Se puede obtener la ubicación aproximada de cada usuario de forma sencilla basándose en los parámetros obtenidos de los puntos de acceso distribuidos por las dependencias de una Escuela. Además, se permite buscar a otros usuarios que tengan la opción de compartir la ubicación activada.

A nivel personal he aprendido un nuevo stack de tecnologías con el que no había trabajado previamente como es el desarrollo de aplicaciones móviles nativas. Por otro lado, ha sido la primera vez que colaboro en un proyecto **Open Source** con todo lo que ello conlleva.

En cuanto a los trabajos futuros y mejoras que se pueden implementar el sistema hay algunas que a continuación paso a explicar.

5.1 TRILATERACIÓN

Al principio de este trabajo analizaba el dominio del problema y comentaba que existe una técnica que usan los sistemas de geolocalización actuales, la trilateración. Esta técnica finalmente no fue incorporada a este proyecto ya que incrementaba la complejidad del desarrollo en el servidor de SWAD y no era necesario ya que la solución aportada era válida para los requisitos establecidos.

Si en el futuro se deseara añadir trilateración el algoritmo debería crecer de forma muy simple de forma que se tomasen los n puntos de acceso más cercanos a ese usuario y pasar esa información a SWAD que se encargase de establecer mediante un algoritmo de coincidencias las similitudes con las localizaciones disponibles. Esto implicaría además que fuese necesaria una actualización de todas las localizaciones con los puntos de acceso que se obtienen desde cada punto. Cuantos más puntos de acceso se tomen como valor n más preciso será el sistema, aunque se recomienda usar 3.

5.2 ADICIÓN DE CENTROS

Con este proyecto se ha conseguido realizar un mapeo de todos los puntos de acceso que hay en la Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación. El proceso que se siguió para conseguir esto ya se ha mencionado anteriormente. Si alguna persona de otro centro estuviese interesada en habilitar esta función tendría que contactar con un administrador del centro en SWAD. Es importante que sea un administrador de centro ya que solo estos tipos de perfiles son los que pueden añadir, modificar y borrar tanto lugares como direcciones de la plataforma. Posteriormente, se procedería a realizar la búsqueda de puntos de acceso del centro. Para realizar este proceso se puede utilizar la aplicación del primer prototipo que se pone a disposición en el repositorio de GitHub [45]. En algunos casos pueden existir documentos con datos sobre la instalación de puntos de accesos inalámbricos en cada centro que facilitarían este proceso.

6 BIBLIOGRAFÍA

- [1] GNU, «gnu.org,» 3 June 2007. [En línea]. Available: <https://www.gnu.org/licenses/gpl-3.0.html>.
- [2] S. Todavchich. [En línea]. Available: https://medium.com/@moqod_development/understanding-open-source-and-free-software-licensing-c0fa600106c9.
- [3] grandviewresearch, «www.grandviewresearch.com,» Octubre 2018. [En línea]. Available: <https://www.grandviewresearch.com/industry-analysis/gps-market>.
- [4] grandviewresearch, «grandviewresearch,» Octubre 2018. [En línea]. Available: <https://www.grandviewresearch.com/industry-analysis/gps-market>.
- [5] «snell,» [En línea]. Available: [https://en.wikipedia.org/wiki/Triangulation_\(surveying\)#Willebrord_Snell](https://en.wikipedia.org/wiki/Triangulation_(surveying)#Willebrord_Snell).
- [6] ESA, «<http://www.esa.int/>,» [En línea]. Available: http://www.esa.int/Applications/Navigation/Galileo/Galileo_a_constellation_of_navigation_satellites.
- [7] GLONASS, «<https://www.glonass-iac.ru/>,» [En línea]. Available: <https://www.glonass-iac.ru/en/>.
- [8] U.S. GOV, «<https://www.gps.gov/>,» [En línea]. Available: <https://www.gps.gov/>.
- [9] Wikipedia, «Trilateración Wikipedia».
- [10] IEEE, «standards.ieee.org,» 2016. [En línea]. Available: https://standards.ieee.org/standard/802_11-2016.html.
- [11] IEEE, «IEEE 802.11mc,» [En línea]. Available: https://en.wikipedia.org/wiki/IEEE_802.11mc.
- [12] Android, «developer.android.com,» [En línea]. Available: <https://developer.android.com/guide/topics/connectivity/wifi-rtt#requirements>.
- [13] Android, «developer.android.com,» [En línea]. Available: <https://developer.android.com/reference/android/content/BroadcastReceiver>.
- [14] A. Ruiz Moya y F. J. Medina Jiménez, «Red IRIS,» Marzo 2006. [En línea]. Available: <https://www.rediris.es/cert/doc/reuniones/fs2006/archivo/fs2006-UGR.pdf>.
- [15] CISCO, «gamma-solutions.com,» [En línea]. Available: <https://www.gamma-solutions.com/wp->

content/uploads/pdf/product_data_sheet0900aec801b9068.pdf.

- [16] SSPRL UGR, [En línea]. Available:
[http://ssprl.ugr.es/pages/servicio_salud/promocion_salud/programa_primeros_auxilios_desfibriladores/_doc/planodeaetsdeingenieriasinformaticaydetelecomunicacion/!](http://ssprl.ugr.es/pages/servicio_salud/promocion_salud/programa_primeros_auxilios_desfibriladores/_doc/planodeaetsdeingenieriasinformaticaydetelecomunicacion/).
- [17] elperiódic, «La UJI diseña un asistente virtual con geolocalización avanzada para las visitas guiadas en museos,» *La UJI diseña un asistente virtual con geolocalización avanzada para las visitas guiadas en museos*, 24 09 2019.
- [18] . P. J. González Negro, B. Barragáns Martínez y N. Fernández García , 09 2018. [En línea]. Available: <http://calderon.cud.uvigo.es/handle/123456789/229>.
- [19] C. Günthe, «Tracing Contacts to Control the COVID-19 Pandemic,» Cornell University, 2020.
- [20] Internal Positioning, [En línea]. Available: <https://www.internalpositioning.com/>.
- [21] Situm, [En línea]. Available: <https://situm.com/es/>.
- [22] Java. [En línea]. Available: https://www.java.com/es/about/whatis_java.jsp.
- [23] Kotlin. [En línea]. Available: <https://kotlinlang.org/>.
- [24] Flask. [En línea]. Available: <https://flask.palletsprojects.com/en/1.1.x/>.
- [25] Django, [En línea]. Available: <https://www.djangoproject.com/>.
- [26] Spring Boot, [En línea]. Available: <https://spring.io/projects/spring-boot>.
- [27] Laravel, [En línea]. Available: <https://laravel.com/>.
- [28] Android, [En línea]. Available: <https://developer.android.com/studio>.
- [29] Eclipse, [En línea]. Available: <https://www.eclipse.org/downloads/>.
- [30] GitHub, [En línea]. Available: <https://github.com/>.
- [31] GitLab, [En línea]. Available: <https://about.gitlab.com/>.
- [32] BitBucket, [En línea]. Available: <https://bitbucket.org/product/>.
- [33] Firebase, [En línea]. Available: <https://firebase.google.com/?hl=es>.
- [34] MongoDB, [En línea]. Available: <https://www.mongodb.com/es>.
- [35] MySQL, [En línea]. Available: <https://www.mysql.com/>.
- [36] MariaDB. [En línea]. Available: <https://mariadb.org/>.
- [37] PostgreSQL. [En línea]. Available: <https://www.postgresql.org/>.

- [38] Heroku. [En línea]. Available: [Heroku.com](https://heroku.com).
- [39] A. W. Services. [En línea]. Available: <https://aws.amazon.com/es/>.
- [40] Cloud Google, [En línea]. Available: <https://cloud.google.com/?hl=es>.
- [41] Docker, [En línea]. Available: <https://www.docker.com/>.
- [42] ITU, «itu.int,» 14 08 2019. [En línea]. Available: <https://www.itu.int/rec/R-REC-P.525-4-201908-I/es>.
- [43] F. Vergès. [En línea]. Available: <https://www.clearToSend.net/cts-115-free-space-path-loss/>.
- [44] Wikipedia, «Wikipedia,» [En línea]. Available: https://en.wikipedia.org/wiki/Free-space_path_loss#Free-space_path_loss_in_decibels.
- [45] J. B. López. [En línea]. Available: <https://github.com/JaviBL8/IndoorPositioning>.
- [46] J. Bueno López, «github.com,» [En línea]. Available: <https://github.com/JaviBL8/IndoorPositioning>.
- [47] IETF, «tools.ietf.org,» Mayo 2015. [En línea]. Available: <https://tools.ietf.org/html/rfc7519>.
- [48] J. B. López. [En línea]. Available: <https://github.com/JaviBL8/MacLocation>.
- [49] J. Bueno López, «github.com,» [En línea]. Available: <https://github.com/JaviBL8/MacLocation>.
- [50] Postman, [En línea]. Available: <https://www.postman.com/>.
- [51] OpenSWAD, [En línea]. Available: <https://openswad.org/api/>.
- [52] Wikipedia, «wikipedia.org,» [En línea]. Available: https://es.wikipedia.org/wiki/Baliza_electr%C3%B3nica.
- [53] CISCO, «barcodesinc,» [En línea]. Available: <https://www.barcodesinc.com/pdf/Cisco/1230ag.pdf>.
- [54] SWADroid, «SWAD,» [En línea]. Available: <https://swad.ugr.es/es>.
- [55] DrawIO, [En línea]. Available: <https://app.diagrams.net/>.
- [56] AGILE FOR ALL, [En línea]. Available: <https://agileforall.com/wp-content/uploads/2011/11/basic-task-board.png>.

7 ÍNDICE DE FIGURAS

Figura 1: Licencias de código abierto más populares. Imagen extraída de artículo web [2]	23
Figura 2: Tamaño del mercado del GPS desde 2015 a 2020. Recuperado de GrandViewResearch [4]	26
Figura 3: Triangularización ejemplificada sobre un plano bidimensional. Recuperado de Wikipedia [9].....	28
Figura 4: Organización en directorios de SWADroid. Captura de Android Studio con el proyecto de SWADroid abierto.....	31
Figura 5: Rango de rúters CISCO. Recuperada de Barcodesinc [52].....	33
Figura 6: Ubicación de los puntos de acceso en Planta 0. Imagen reutilizada [16].....	34
Figura 7: Icono FIND. Recuperada de FIND [20]	39
Figura 8: Icono situm. Recupera de Situm [21]	39
Figura 9: Metodología de trabajo seguida en SCRUM. Elaboración propia.	42
Figura 10: Scrum Taskboard. Recuperada de AGILE FOR ALL. [53]	43
Figura 11: Ejemplo de amplitud frente a distancia para FSPL. Recuperada de cleartosend.net [43].....	50
Figura 12: Aplicación para la recolección de datos. Captura de aplicación IndoorPositioning [45].	54
Figura 13: Base de datos con los puntos de acceso en Firebase. Captura tomada desde Firebase. Elaboración propia.....	55
Figura 14: Creación y modificación de aulas en SWAD. Recuperada de SWAD [54].....	59
Figura 15: Ejemplo de llamada a una función de la API de SWAD. Captura de pantalla tomada de Postman. Elaboración propia.	61
Figura 16: Función getLastLocation de la API de SWAD. Recuperada de SWAD API [50]	61
Figura 17: Diagrama de actividad. Realizado con DrawIO [55]	62
Figura 18: Diagrama de secuencia. FindUserLocation.....	63
Figura 19: Diagrama de secuencia. UpdateLocation. Realizado con DrawIO [55]	63
Figura 20: Árbol de components para fichero indoor_location.xml. Captura de pantalla del Component Tree. Elaboración propia.	65

Figura 21: Traducciones a español del fichero strings.xml. Captura del fichero /res/values/string. Elaboración propia.....	66
Figura 22: Declaración de preferencias. Captura del fichero Preferences.java. Elaboración propia.....	66
Figura 23: Preferencias de refresco de ubicación	67
Figura 24: Captura de SetOnClickListener updateLocation en la clase IndoorLocation. Elaboración propia.....	68
Figura 25: Método GetLocation de la API de SWAD.....	69
Figura 26: Función request service de la clase GetLocation. Elaboración propia.	71
Figura 27: Obtención de campos de un SoapObject en la clase GetLocation. Elaboración propia.	72
Figura 28: Listener Find user location.....	72
Figura 29: Buscar usuario usando SearchClass.....	73
Figura 30: Función onActivityResult. Captura de la clase IndoorLocation. Elaboración propia...74	
Figura 31: Fragmento de Constants.java. Elaboración propia.	74
Figura 32: Search users result. Captura tomada de la clase IndoorLocation dentro del método onActivityResult. Elaboración propia.	75
Figura 33: GetLastLocation endpoint. Recuperada de la API de SWAD [50]	76
Figura 34: GetLastLocation result.....	76
Figura 35: Implementación de cálculo de distancia. Elaboración propia.....	78
Figura 36: Implementación del escanner de Wifi. Elaboración propia.	79
Figura 37: Ajustes de SWADroid.....	81
Figura 38: Configuración SWADroid	82
Figura 39: Opción de localización dentro del menú Usuarios	83
Figura 40: Menú de localización	84
Figura 41: Buscar ubicación de usuario	85

8 ÍNDICE DE TABLAS

Tabla 1: Comparación de los distintos sistemas empleados para la localización. Elaboración propia.....	37
Tabla 2: Atenuación en frecuencias de 2,4 GHz a 1 m de distancia	51
Tabla 3: Atenuación en frecuencias de 5 GHz a 1 m de distancia	52
Tabla 4: Ubicación de routers inalámbricos identificados por su MAC. Elaboración propia.	56
Tabla 5: API de MacLocation [48]. Elaboración propia.....	57
Tabla 6: Tabla room_MAC en SWAD. Elaboración propia.....	77
Tabla 7: Tabla rooms en SWAD. Elaboración propia.....	77
Tabla 8: Tabla room_check_in en SWAD.....	78