# Título del Proyecto:

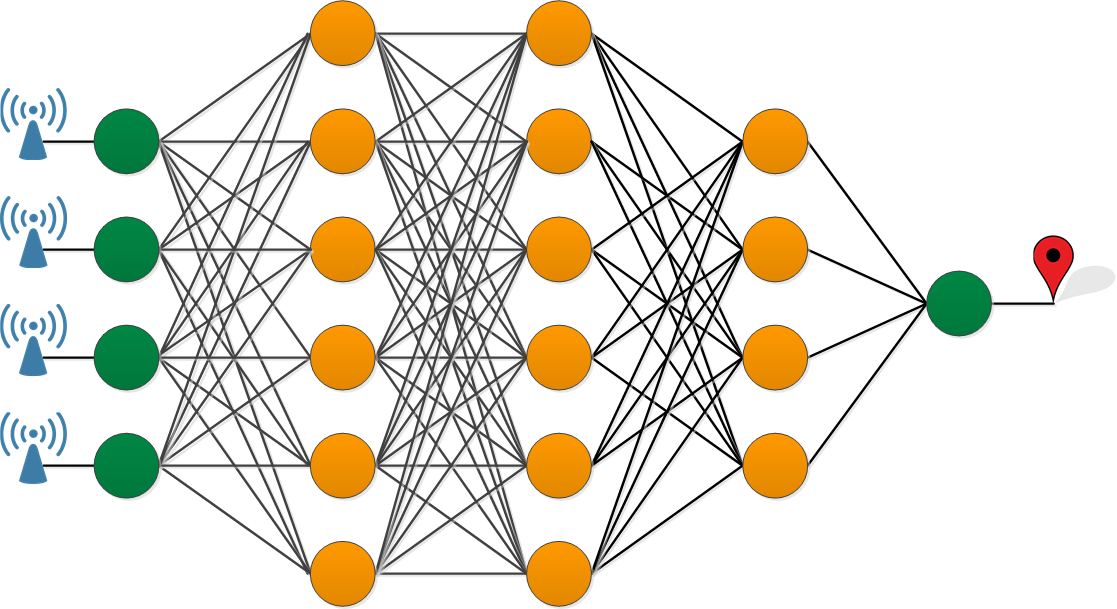
**WiFlow**

# Autor:

Fernández Ippólito, Juan Carlos

# Director:

Dr. Tomás Gironés, Jesús



WiFlow

**TESINA PARA LA**

**OBTENCIÓN DEL TÍTULO DE:**

# Máster en Desarrollo de Aplicaciones Android

**Septiembre de 2018**

Contenido

[Introducción 3](#_Toc421525768)

[Descripción del problema 3](#_Toc421525769)

[Objetivos 3](#_Toc421525770)

[Motivación 3](#_Toc421525771)

[Situación de… / Tecnologías utilizadas 3](#_Toc421525772)

[Arquitectura de la aplicación 3](#_Toc421525773)

[Esquema del diseño 3](#_Toc421525774)

[Modelo de datos 3](#_Toc421525775)

[Vistas 3](#_Toc421525776)

[Conclusiones 3](#_Toc421525777)

[Anexo fuentes 4](#_Toc421525778)

[Listado de fuentes entregadas o enlace a GitHub 4](#_Toc421525779)

[Manual de usuario 4](#_Toc421525780)

# Introducción

## Descripción del problema

Antecedentes y descripción de la problemática a resolver.

La localización derivada de señales satelitales puede volverse muy dificultosa o incluso imposible dentro de una edificación, ya sea un museo, una universidad, etcétera. Esto es debido a la atenuación que supone a esas señales, de por sí ya muy pequeñas, los materiales utilizados en la construcción. Esto es especialmente cierto si se desea discernir pocos metros y la exactitud de la posición empeora debido al efecto de esas atenuaciones.

## Objetivos

El objetivo es encontrar una solución tecnológica que pueda dar respuesta a la pregunta ¿dónde estamos?, en el contexto descripto anteriormente: localización en interiores.

## Motivación

Como proyecto final de la asignatura Android Fundamentos, dictada por la UPV a fin del año 2015, realicé un trabajo denominado WiFi Locus [[1]](#endnote-1). Este proyecto consistía en una aplicación para localización en interiores. Mediante las potencias emitidas por los puntos de acceso WiFi que llegaban al dispositivo móvil, registrados en una en una etapa inicial de entrenamiento haciendo uso de la aplicación en distintos lugares del edificio, intentaba luego calcular la posición en que se encontraba el dispositivo móvil, utilizando fundamentalmente un método de mínima diferencia de cuadrados más algunos otros algoritmos de ajuste, descriptos en la memoria de ese proyecto [[2]](#endnote-2).

¿Sería posible obtener una funcionalidad similar utilizando técnicas de inteligencia artificial, como lo son las redes neuronales con sistemas de aprendizaje automático?

Un desafío interesante, especialmente estando disponible TensorFlow, con la posibilidad entonces de utilizarlo en Android.

# Tecnologías utilizadas

Incluir la información necesaria que facilite la comprensión del proyecto. Solo se incluirá este punto para describir alguna tecnología, paradigma de diseño, plataforma, … no descrita durante el Diploma o Master. Su extensión será siempre reducida.

**Redes neuronales de aprendizaje automático**

Los sistemas de AA (**A**prendizaje **A**utomático) supervisado, aprenden cómo combinar entradas para producir predicciones útiles sobre datos nunca antes vistos, mediante una serie de ejemplos que se les suministra durante la etapa de entrenamiento.

**Tensorflow**

TensorFlow es una biblioteca de código abierto para aprendizaje automático desarrollada por Google, mediante la cual es posible construir y entrenar redes neuronales para detectar y descifrar patrones y correlaciones. Fue liberado como software de código abierto en noviembre del 2015.

El nombre TensorFlow deriva de las operaciones qué tales redes neuronales realizan sobre arrays multidimensionales de datos. Estos arrays multidimensionales son referidos como "tensores"

**T**ensor**F**low proporciona una API de Python, así como APIs de C++ y otros lenguajes. La enorme cantidad de ejemplos y documentación de TensorFlow disponible en Python puede hacer que tal vez el lector comience utilizando este lenguaje para familiarizarse con **TF**, aunque Python parezca inicialmente poco amigable. Al menos ese fue mi caso.

**Keras**

La API de alto nivel de Keras, proporciona bloques de construcción para crear, entrenar y utilizar en predicciones a modelos de AA.

Tiene una interfaz simple optimizada para casos de uso común. Proporciona comentarios bastante claros los errores en la escritura del código y en la ejecución.

Los modelos de las redes en Keras se hacen conectando bloques de construcción configurables, con pocas restricciones.

TensorFlow tiene embebida una implementación de la especificación de la API de Keras, de manera que instalando **TF**, Keras ya está disponible.

Trabajar con Keras, en lugar de hacerlo solo con TensorFlow, hace la curva de aprendizaje del trabajo con redes neuronales menos traumática.

Como pasa en muchos otros paradigmas, al aumentar el nivel y ganar en simplicidad, el costo que hay que pagar es alguna pérdida de versatilidad. En particular en este trabajo, la dificultad más grande a superar fue que el formato modelo de red entrenado usando Keras, no puede ser utilizado directamente en Android y el proceso de obtener uno compatible con ese sistema operativo, fue casi artesanal.

**Python**

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible.

Se trata de un lenguaje de programación multiparadigma, esto significa que más que forzar a los programadores a adoptar un estilo particular de programación, permite varios estilos: programación orientada a objetos, programación imperativa y programación funcional. Otros paradigmas están soportados mediante el uso de extensiones. Es un lenguaje interpretado, usa tipado dinámico (no se comprueban los tipos a medida que se van haciendo los cálculos) y es multiplataforma.

Es muy usado en áreas de ciencia e ingeniería. Tanto en forma nativa como por medio de bibliotecas importadas posee gran potencia para el manejo de datos y cálculos numéricos.

# Arquitectura de la aplicación

NOTA: Los siguientes puntos son orientativos

La idea general de proyecto es colectar cuidadosamente los datos de potencia e identificación (BSSID, es decir el MAC address al que se conecta) de los puntos de acceso que son detectados en un dispositivo móvil, a lo largo y ancho de distintos espacios dentro de una edificación, tomando nota de la ubicación en que se obtuvo cada medición.

A medida que nos desplacemos en cada recinto, el dispositivo móvil registrará distintas potencias e incluso pueden cambiar la constelación de puntos de accesos recibidos: es posible que algunos se pierdan y también que aparezcan otros nuevos.

Estos datos serán utilizados para entrenar y validar el funcionamiento del modelo de la red neuronal.

Las potencias y las BSSIDs constituirán las características utilizadas como entrada en el entrenamiento de la red, mientras que las posiciones correspondientes (una para cada conjunto de potencias y BSSIDs obtenidas en un punto determinado de cada recinto, oficina o espacio) serán las etiquetas que se utilizarán para ajustar los pesos que conectan las neuronas, distribuidas en distintas capas.

## Componentes y herramientas utilizadas

Wifi Locus modificado: https://github.com/jon1721/WiFiLocusModificado.git

JADE Javascript based Database Editor: es un plugin para Chrome, disponible en Chrome Web Store, que permite editar y observar base de datos SQLite: https://chrome.google.com/webstore/detail/javascript-based-database/bponbdjkefbmgkfiiphhabghkkfocook

RebaseData: conversor en línea de SQLite a CSV: https://www.rebasedata.com/convert-sqlite-to-csv-online

Microsoft Excel

PyCharm: IDE para Python desarrollado por JetBrains. Tiene enormes ventajas para quienes han usado Android Studio, pues la GUI y su uso son muy similares. La búsqueda, descarga e instalación de paquetes está muy simplificada. La versión libre Community es totalmente funcional: https://www.jetbrains.com/pycharm/

TensorFlow: https://www.tensorflow.org/

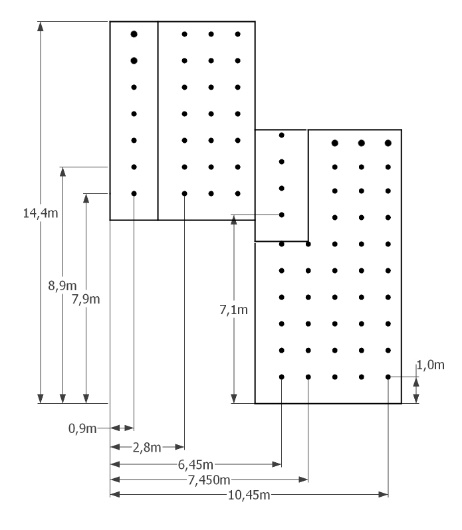
Keras: https://keras.io/

Tensorboard: https://www.tensorflow.org/guide/summaries\_and\_tensorboard

Paso a paso

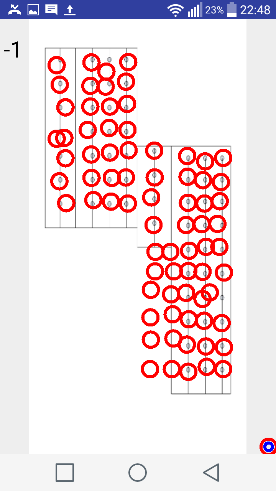
Obtención de los datos

El ámbito de medición fue el que se muestra en el siguiente plano:



En el plano se observan las dimensiones en metros, las paredes y los puntos en que realizaron las mediciones, representados por los pequeños círculos negros.

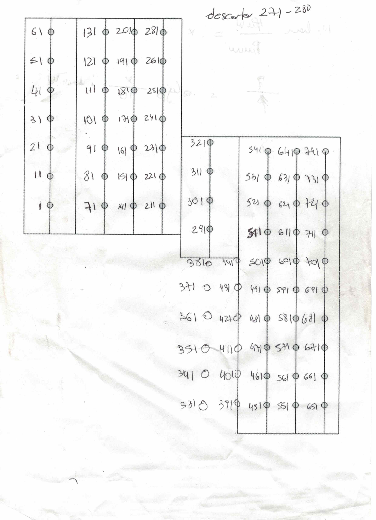
La que sigue es una captura de pantalla de la aplicación WiFi Locus modificada[[3]](#endnote-3) que se utilizó para la captura de los datos:



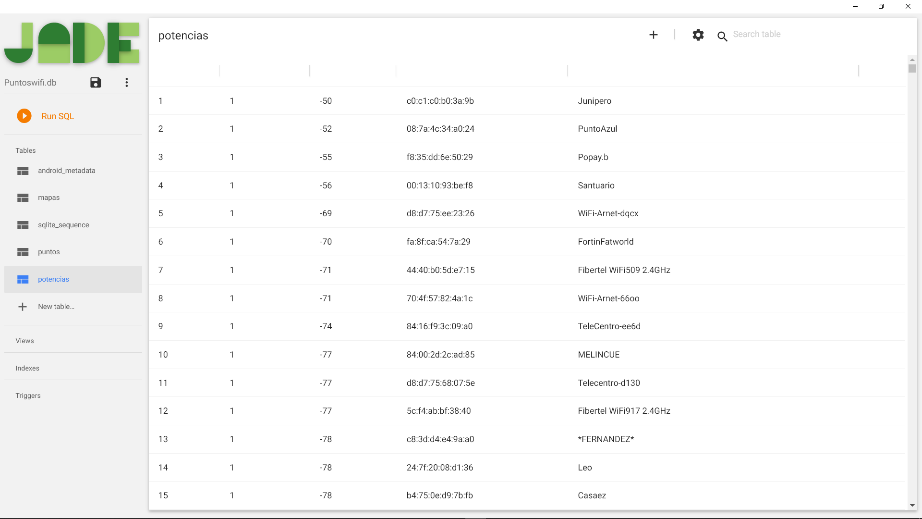
Para realizar cada medición, hay que situarse en la habitación en el punto a medir. En esa condición, pulsar la pantalla. Eso dispara unas mediciones sucesivas, cuya cantidad debe haber sido configurada previamente en los ajustes de la aplicación, almacenándose los datos en una base de datos SQLite. El punto en cuestión queda indicado en la pantalla mediante un círculo rojo.

En total se tomaron mediciones válidas en 74 puntos, 10 mediciones en cada punto, cada una indicada mediante su ID de registro en la base de datos, el cual con cada medición se mostraba en el ángulo superior izquierdo de la pantalla, con el objetivo de llevar un registro externo del proceso.

La siguiente imagen muestra el registro manual del ID de registro inicial de cada punto:



De esta manera, la aplicación generó el archivo Puntoswifi.db[[4]](#endnote-4), en formato SQLite.



Aunque Python puede manejar archivos en formato SQlite, preferí exportarlo a CSV, para poder ver y analizar los resultados en Excel. Utilicé entonces RebaseData [[5]](#endnote-5)para exportar Puntoswifi.db a CSV.

De los archivos obtenidos en este proceso los importante son potencias.csv y puntos.csv, pues contienen la identificación del punto (su posición dentro del edificio) y la identificación de los puntos de acceso WiFi con el nivel de potencia medido para cada uno de ellos en ese lugar.

## Esquema del diseño

Diagrama con los diferentes componentes del diseño y sus interrelaciones.

Justificación de las principales decisiones tomadas en el diseño.

## Modelo de datos

Esquema de la base de datos, incluyendo relaciones entre las tablas.

Listado de servicios webs.

## Vistas

Esquema que muestre las principales pantallas de la aplicación y el diagrama de navegación.

# Capítulos adicionales

Si consideras puedes poner más capítulos adicionales

# Conclusiones

Grado de cumplimiento de los objetivos planteados.

Líneas abiertas.

Consideraciones personales.

# Anexos

## Listado de fuentes entregadas / Código fuente en GitHub

## Manual de usuario

Links

1. http://androidcurso.com/index.php/recursos/proyectos/67-proyectos-android-fundamentos/proyectos-android-fundamentos-6-ed?start=5 [↑](#endnote-ref-1)
2. http://www.dcomg.upv.es/~jtomas/android/ProyectosFundamentos2015/WifiLocus.pdf [↑](#endnote-ref-2)
3. https://github.com/jon1721/WiFiLocusModificado.git [↑](#endnote-ref-3)
4. Archivo disponible en https://github.com/jon1721/wiflow.git [↑](#endnote-ref-4)
5. https://www.rebasedata.com/convert-sqlite-to-csv-online [↑](#endnote-ref-5)