FACULTAD DE INGENIERÍAS Y CIENCIAS AGROPECUARIAS

DESARROLLO DE UNA APLICACIÓN MÓVIL PARA SISTEMA OPERATIVO ANDROID QUE REALICE MEDICIONES Y MAPEO DE RUIDO UTILIZANDO GEOLOCALIZACIÓN.

Trabajo de Titulación presentado en conformidad con los requisitos establecidos para optar por el título de Ingeniero en Electrónica y Redes de Información

Profesor Guía

Mg. Ángel Gabriel Jaramillo Alcázar

Autor

Miguel Antonio Amores Bassante

Año

2016

DECLARACIÓN DEL PROFESOR GUÍA

“Declaro haber dirigido este trabajo a través de reuniones periódicas con el estudiante, orientando sus conocimientos y competencias para un eficiente desarrollo del tema escogido y dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación”

_____

Ángel Gabriel Jaramillo Alcázar.
Magister. Gerencia de Sistemas y Tecnologías de Información
C.I 1715891964

DECLARACIÓN DE AUTORÍA DEL ESTUDIANTE

“Declaro que este trabajo es original, de mi autoría, que se han citado las fuentes correspondientes y que en su ejecución se respetaron las disposiciones legales que protegen los derechos de autor vigentes”

_____

Miguel Antonio Amores Bassante

C.I 0502896467

AGRADECIMIENTOS

“Agradezco a mi familia y a todas las personas que estuvieron pendientes”.

DEDICATORIA

“A mis padres, a mis hermanos
y a mis amigos”.

RESUMEN

Este trabajo de titulación trata de el desarrollo de una aplicación móvil en SO Android siguiendo una metodología de desarrollo ágil SCRUM que permita medir el nivel de ruido en cualquier lugar, y ubicar la posición en un mapa utilizando el API de Google Maps para así poder generar un mapa de calor de todos los lugares en donde se han realizado las mediciones. Se utilizará el micrófono del dispositivo para hacer la medición. A la aplicación se hará un registro y *login* para el acceso, se mostrarán notificaciones cuando se mida el ruido y los datos de mediciones serán enviados a una base de datos para después mostrarlos en una página web mediante un mapa de calor. El sitio web permitirá mostrar reportes de las mediciones realizadas en un mapa de calor. Se usará lenguaje HTML y JavaScript para el desarrollo. Se usa una base de datos para almacenar los datos de registros y mediciones realizadas en la aplicación móvil. La infraestructura estará basada en la aplicación móvil, la cual permita realizar al usuario las mediciones de ruido. Se usará lenguaje de programación Java y Android Studio como IDE. La infraestructura tecnológica para el desarrollo del prototipo será dimensionada en un entorno de acceso público a través de Internet.

ABSTRACT

This work is a mobile application develop on Android OS following SCRUM, an agile development methodology, to measure the noise level anywhere, and locate the position on a map using the Google Maps API in order to generate a heat map of all the places where measurements have been made. The microphone of the device will be used to make the measurement. The application will have register and login to access, notifications will be displayed when the noise and the measured measurement data will be sent to a database and then display them on a web page via a heat map. The web site will show reports of measurements made in a heat map. HTML and JavaScript language will be used for development. A database is used to store data records and measurements in the mobile application. The infrastructure will be based on the mobile application, which allows the user to perform noise measurements. Java and Android Studio IDE will be used. The technological infrastructure for the development of the prototype will be sized in a public access environment via the Internet.

INDICE

INDICE DE FIGURAS

INDICE DE TABLAS

Introducción

Alcance

El alcance de este trabajo de titulación es desarrollar una aplicación móvil en SO Android siguiendo una metodología de desarrollo ágil SCRUM que permita medir el nivel de ruido en cualquier lugar, y ubicar la posición en un mapa utilizando el API de Google Maps para así poder generar un mapa de calor de todos los lugares en donde se han realizado las mediciones. Se utilizará el micrófono del dispositivo para hacer la medición. A la aplicación se hará un registro y l*ogin* para el acceso, se mostrarán notificaciones cuando se mida el ruido y los datos de mediciones serán enviados a una base de datos para después mostrarlos en una página web mediante un mapa de calor.

Los componentes de la solución serán los siguientes:

Aplicación Web: Se desarrollará una página web para mostrar reportes de las mediciones realizadas en un mapa de calor. Se usará lenguaje HTML y JavaScript para el desarrollo.

Servicio web Restful: Se desarrollarán servicios web que permitan enviar y consultar datos de una base de datos SQL.

Base de datos: Se usará una base de datos para almacenar los datos de registros y mediciones realizadas en la aplicación móvil.

Aplicación Móvil: La infraestructura estará basada en la aplicación móvil, la cual permita realizar al usuario las mediciones de ruido. Se usará lenguaje de programación Java y Android Studio como IDE.

La infraestructura tecnológica para el desarrollo del prototipo será dimensionada en un entorno de acceso público a través de Internet.

Para alcanzar el cumplimiento de este proyecto se va a utilizar lo aprendido en las materias de: programación orientada a objetos, programación en aplicaciones móviles y temas aprendidos de forma particular.

Justificación

La razón principal de realizar este proyecto es debido a que no existe ninguna aplicación en el mercado Ecuatoriano que permita a una persona conocer los niveles de ruido al que está expuesto en un determinado lugar o donde realiza sus actividades diarias y mostrar los resultados en un mapa de calor. Considerando las mediciones se pueden tomar acciones al respecto de los niveles de ruido. Esta será una aplicación bastante práctica y de fácil manejo para cualquier persona.

Adicionalmente el proyecto propuesto puede ser de gran ayuda siendo utilizada como una herramienta de alerta para instituciones que regulan el medio ambiente. Como se sabe el impacto de los niveles de ruido es un gran problema en los sectores urbanos, por lo tanto es necesario saber en qué lugares existen niveles de ruido más alto del tolerable. De esta manera al hacer una medición de ruido en un determinado lugar y ubicarlo en el mapa, otras personas pueden estar alertadas que en ese lugar existe un nivel de ruido muy alto y tomar las debidas precauciones.

Objetivo General

Desarrollar una aplicación móvil para sistema operativo Android que realice mediciones de nivel de ruido y mapeo utilizando geolocalización.

Objetivos Específicos

- Analizar los parámetros que determinan los niveles de ruido a partir de la normativa de la OMS.
- Analizar las metodologías y herramientas a utilizarse para el desarrollo de la aplicación.

- Diseñar la solución en base a los requerimientos levantados para el proyecto.
- Realizar pruebas funcionales de la aplicación implementada en diferentes ambientes donde los niveles de ruido sean altos, medios y bajos.

1. Capítulo I. Marco Teórico

1.1. El ruido

El ruido es un sonido no deseable el cual se mide en decibelios (dB) (OMS, 2015). Es un factor muy molesto que ha ido creciendo en los últimos años, superando los niveles tolerables y siendo este originado por fuentes fijas y móviles. El sector urbano es el más afectado debido a las industrias, el aeropuerto y la gran cantidad de vehículos. La situación empeora cuando aumenta el tráfico, los conductores no usan de manera adecuada las bocinas de sus autos.

Según la Organización mundial de la Salud, el 76% de la población que vive en zonas urbanas sufre un gran impacto de ruido. Algunos de los efectos que causa el ruido son:

- Enfermedades fisiológicas: Se pueden producir en ambientes en torno a 100 decibelios, algunas tan importantes como la pérdida parcial o total de la audición.
- Enfermedades psíquicas: Se producen por el exceso de ruido, tales como el estrés, pérdidas de sueño, disminución de la atención, falta de rendimiento, etc.
- Enfermedades sociológicas: Alteraciones en la comunicación, el rendimiento.
- Las zonas con más de 65 decibelios son consideradas inaceptables. Cuando el ruido está por debajo de los 80 dB el oído humano no representa una alteración definitiva, solamente generan molestias pasajeras. Cuando el ruido sobrepasa los 90 dB pueden aparecer lesiones irreversibles cuanto mayor sea la exposición a este. A continuación se presenta una tabla con los niveles de ruido máximos permisibles:

TIPO DE ZONA SEGÚN USO DE SUELO	NIVEL DE PRESIÓN SONORA EQUIVALENTE NPS eq [dB (A)]	
	DE 06H00 A 20H00	DE 20H00 06H00
Zona hospitalaria y educativa	45	35
Zona Residencial	50	40
Zona Residencial mixta	55	45
Zona Comercial	60	50
Zona Comercial mixta	65	55
Zona Industrial	70	65

Figura 1. Niveles máximo de ruido permisible

Tomado del Anexo 5 del Libro VI del Texto Unificado de Legislación Secundaria del Ministerio del Ambiente.

Nivel de intensidad del sonido	
200 dB	Bomba atómica
180 dB	Explosión de volcán / Cohete en despegue
140 dB	Umbral del dolor
137.5 dB	Récord Guiness de ruido en un estadio
130 dB	Avión en despegue
120 dB	Motor de avión en marcha
110 dB	Concierto / acto cívico
100 dB	Perforadora eléctrica
90 dB	Tráfico / Pelea de dos personas
80 dB	Tren
70 dB	Aspiradora
50/60 dB	Aglomeración de gente / Lavaplatos
40 dB	Conversación
20 dB	Biblioteca
10 dB	Respiración tranquila
0 dB	Umbral de audición

Figura 2. Nivel de ruido que generan algunas fuentes fijas y móviles.

Tomado de (MetrosCubicos, s.f)

En el Ecuador, el Ministerio de Ambiente (MAE), es la entidad encargada de realizar controles y seguimientos del ruido, verificando el cumplimiento del Plan de Manejo Ambiental de proyectos y actividades. La forma de realizarlo es con auditorías ambientales y reportes de monitoreo ambiental.

Según un artículo del comercio, “las multas por ruido varían entre 0.2 y 4 salarios básicos unificados dependiendo de la gravedad de la infracción.” (El Comercio, 2010)

1.1.1 Decibel

Es una medida que se usa en acústica para comparar una cantidad con otra conocida como referencia, por lo tanto el decibelio se usa para comparar la presión sonora con una presión de referencia. La presión de referencia puede variar según el tipo de medida que se va a realizar. En el caso de la presión sonora se usa la siguiente:

Nivel de Referencia para la Presión Sonora (en el aire) = 0.00002 = 2E-5 [Pa]
La fórmula usada para medir la presión Sonora es la siguiente

$$SPL = 20 * \log(P/Pr) \qquad \text{(Ecuación 1)}$$

Donde:
SPL = Nivel de presión sonora
P = Presión medida
Pr = Presión de referencia

Los decibelios son un valor lineal, quiere decir que son valores tomados sin ninguna alteración. Si la presión sonora el medida de esta forma, linealmente, tendrá poco valor en la percepción del oído humano. Por esto en necesario ajustar los valores de dB para que puedan ser percibidos correctamente por el oído humano. La sensibilidad del oído humano ante las variaciones de presión sonora sigue una escala logarítmica. Por ello la corrección de los dB se realiza ponderando los valores medidos. Los decibeles ya ponderados en “A” son representados como dBA y los lineales como dB.

Por ejemplo si se miden 80dB en una frecuencia específica, al ponderar la medida será 60.9 dBA, es decir que el nivel de presión sonoro de 80dB será captado por el oído humano como 60.9 dBA.

1.2. Metodología de desarrollo

“Una metodología de desarrollo de software, consiste en la convención de prácticas, métodos, principios, técnicas y herramientas cuya principal utilidad es la de otorgar un mejor rendimiento del equipo de trabajo y por sobre todo, permitir la obtención de mejores resultados en lo que se produce durante el proyecto.” (Bahit, 2011).

Existen las metodologías tradicionales y metodologías ágiles. Las tradicionales son orientadas por planeación, es decir, primero inician con un riguroso proceso de recolección de requerimientos y luego el análisis y diseño. Las metodologías tradicionales se eligen cuando el proyecto es muy grande, tiene su estructura muy definida, sigue un proceso en una sola dirección sin marcha atrás, el proceso nunca cambia, los requerimientos ya están definidos para todo el proyecto y demanda poca comunicación con el cliente una vez terminado el proyecto. (Permalink, 2015).

Las metodologías ágiles en cambio se pueden modificar para que se ajusten al equipo de trabajo y al proyecto. Los proyectos se dividen en proyectos más pequeños para ser tratados de forma independiente y realizarlos en un período corto. Además un punto fuerte es la constante comunicación con el cliente ya que tienen una retroalimentación y se pueden adaptar a los cambios.
La tabla 3 detalla los aspectos de ambos tipos de metodologías:

Metodologías tradicionales	Metodologías ágiles
Predictivos	Adaptativos
Orientados a procesos	Orientados a personas
Proceso rígido	Proceso flexible
Se concibe como un proyecto	Un proyecto es subdividido en varios proyectos más pequeños
Poca comunicación con el cliente	Comunicación constante con el cliente
Entrega de software al finalizar el desarrollo	Entregas constantes de software
Documentación extensa	Poca documentación

Figura 3. Metodologías tradicionales vs metodologías ágiles.

Tomado de (Cadavid, Fernández y Morales, s.f)

En este proyecto no se usará una metodología tradicional ya que estas necesitan de una documentación muy estricta, además que no son adaptables a los cambios. Por lo contrario, las metodologías ágiles son modernas y tienen otra mentalidad, la cual es adaptarse a los cambios sabiendo lo amplio que es el mundo del software.

Entre las metodologías ágiles más usadas y con mejores resultados se encuentran SCRUM y XP (Bahit, 2011). Escoger entre estas metodologías, depende mucho de la empresa o de los desarrolladores. A continuación se presenta una tabla con ventajas y desventajas de ambas:

Tabla 1. Ventajas y desventajas de SCRUM y XP

	Ventajas	Desventajas
SCRUM	• Se entrega un producto funcional al final de cada sprint. • Se puede ajustar la funcionalidad a la necesidad del cliente. • Visualización del proyecto día a día.	• No posee tanta evidencia o documentación.
XP	• Programación organizada. • Taza de errores menor. • Refactorización. • Programación en parejas.	• Recomendable en proyectos cortos. • Imposible prever todo antes de programar.

En este caso se escoge Scrum al ser más apropiada para el proyecto y no necesitar de una documentación tan estricta, además de poder siempre entregar un producto funcional. Scrum es un proceso donde se aplican un conjunto de buenas prácticas y poder obtener un mejor resultado. Scrum es más propicio para proyectos donde se necesitan obtener rápidos resultados, por esta razón es que se realizan entregas parciales donde los requisitos pueden cambiar.

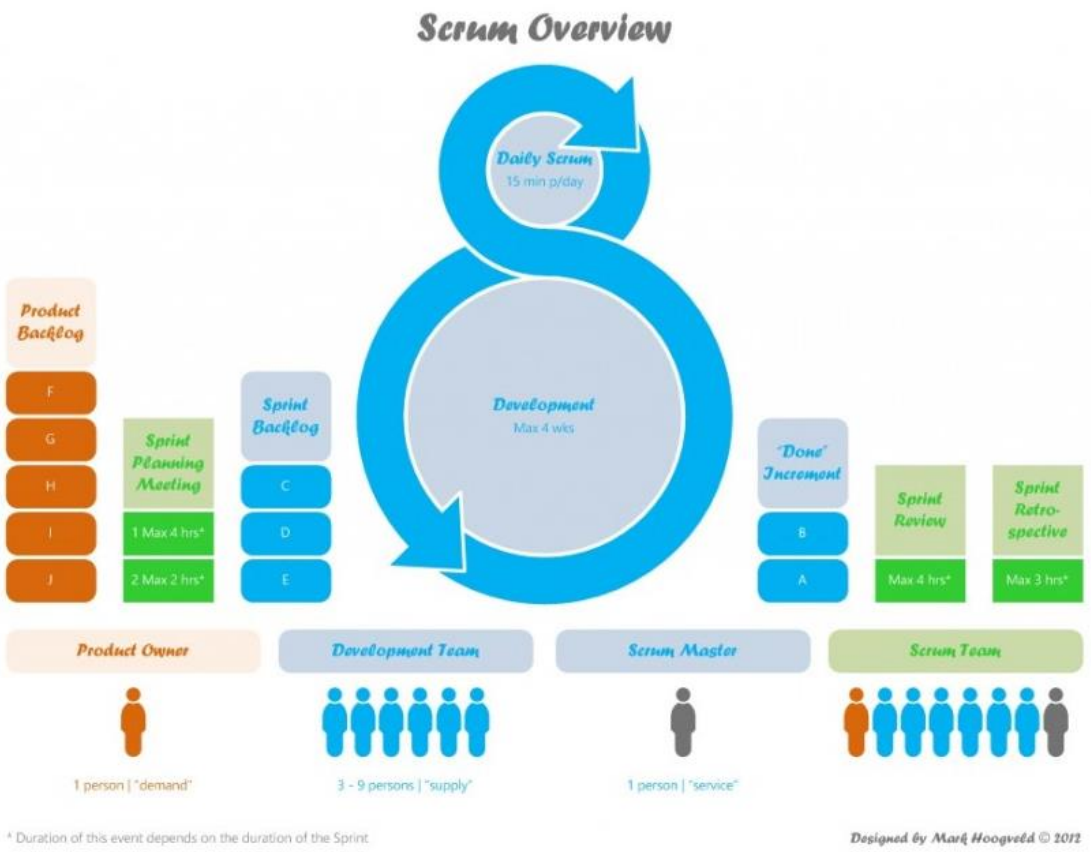

Figura 4. Roles en SCRUM.

Tomado de (Calvinx, s.f)

1.2.1 Roles

Product Owner (Cliente)

Es quien representa a todas las personas interesadas en el proyecto y escribe las historias de usuario, las prioriza y coloca en el Product Backlog. Además se asegura que el equipo trabaje de forma adecuada de acuerdo a la perspectiva del negocio (Bahit, 2011).

Scrum Master (Facilitador)

Es el encargado de que el equipo cumpla con el objetivo del sprint llevando acabo los valores y principios agiles, las reglas y procesos de scrum. Elimina

todos los obstáculos que tenga el equipo para poder finalizar el proyecto con éxito. Protege al equipo de las distracciones que permitan realizar un sprint satisfactoriamente. (Bahit, 2011).

Team (Equipo de desarrollo)

Es el grupo de desarrollo que tiene la responsabilidad de entregar el proyecto. Es un equipo auto organizado donde comparten información y confían entre ellos. Es recomendable que el equipo sea estable durante el proyecto y que cada miembro sea solidario, colaborador y evitar la competencia. (Bahit, 2011).

1.2.2 Actividades

Sprint (Ejecución de la iteración)

Es el periodo en el cual se realiza un trabajo. Generalmente los sprint duran de 2 a 3 semanas y es recomendable que sea constante. Al final de un sprint, el equipo debe mostrar los avances logrados y el resultado obtenido debe ser un producto entregable para el cliente. (ProyectosAgiles, 2014).

Sprint Planning (Planificación de la iteración)

En esta reunión el cliente presenta al equipo los requisitos del proyecto priorizados, donde luego el equipo los examina, pregunta las dudas al cliente y selecciona los requisitos con mayor prioridad que se compromete a completar. (ProyectosAgiles, 2014).

Scrum Daily Meeting

Esta reunión sirve para facilitar el intercambio de información y colaboración entre los miembros del equipo para así poder ser más productivo. Cada miembro realiza una revisión de otro para poder hacer nuevas adaptaciones. Si es necesario. Cada miembro del equipo debe responder a las siguientes preguntas:

- ¿Qué he hecho desde la última reunión?

- ¿Qué voy a hacer a partir de este momento?
- ¿Qué impedimento tengo o voy a tener para cumplir algún objetivo?

(ProyectosAgiles, 2014).

Sprint Review (Revisión del sprint)

- Revisar los trabajos completados y los no completados.
- Presentar el trabajo completado al cliente. Por lo general su alias es *demo*.
- El trabajo que no se completó no puede ser presentado.
- Se realiza en un tiempo máximo de 4 horas.

(ProyectosAgiles, 2014).

Sprint Retrospective (Retrospectiva)

Con el objetivo de mejorar la productividad y calidad en el proyecto que se está desarrollando, el equipo analiza la manera en que trabaja durante un sprint, porque se consiguen o no los objetivos que se comprometieron a un inicio.

- ¿Qué cosas han funcionado bien?
- ¿Qué es lo que hay que mejorar?
- Probar nuevas cosas en el siguiente sprint.

(ProyectosAgiles, 2014).

1.2.3 Herramientas

Product Backlog

Es una lista ordenada de todos los requerimientos del producto, el cual contiene descripciones de las funcionalidades deseadas. El product owner es el responsable del product backlog, de incluir el contenido, viabilidad y orden. Es dinámico, lo que quiere decir que constantemente cambia. El Product Backlog lista todas las características, funciones, requerimientos y mejoras futuras para el producto, representa el “qué” va a ser construido. Contiene estimaciones, tanto del valor para el negocio como el esfuerzo del equipo de desarrollo. Con esto el product owner puede priorizar las tareas y así comenzar

con los requerimientos más importantes y que lleven menor tiempo al equipo de desarrollo. (ProyectosAgiles, 2014).

Sprint Backlog

Es un subconjunto de requisitos que describen el “cómo” el equipo de desarrollo va a implementar los requerimientos. Por lo general los requerimientos se subdividen en tareas a las cuales se les asignan ciertas horas de trabajo, ninguna superior a 16 horas. Las tareas del Sprint Backlog no se asignan, sino cada miembro del equipo las toma de acuerdo según sea más adecuado. (ProyectosAgiles, 2014).

1.3 Herramientas

1.3.1 Android Studio

Android Studio es un entorno de desarrollo integrado (IDE), está diseñado específicamente para desarrollar en Android y fue basado en IntelliJ IDE de JetBrains.

Esta herramienta está disponible para descargar para Windows, Mac OS X y Linux, y se lo distribuye de forma gratuita, actualmente Android Studio está en la versión 1.2 y cuenta con las siguientes características:

- Renderización en tiempo real.
- Consola de desarrollador: esta característica provee de consejos de optimización, ayuda para la traducción, estadísticas de uso.
- Editor de código: esta característica permite escribir código utilizando un editor profesional con un avanzado terminador de código, plegado de código, resaltado de sintaxis, y mensajes de alerta que muestran advertencias, errores, y otra información contextual en la línea donde se generó la equivocación.

- Integración con GitHub: esta característica permite comenzar proyectos usando códigos de ejemplos de Google desde GitHub.
- Dispositivos virtuales: esta característica permite hacer uso de Virtual Device Manager, el cual provee de dispositivos pre-definidos de Android para depurar y ejecutar las aplicaciones.
- Soporte para Gradle: Android Studio cuenta con el soporte de Gradle, una herramienta de automatización de la construcción de nuestro código.

(developer.android, 2015).

La razón de utilizar esta herramienta y no otras de las que se puede encontrar, es porque es la herramienta oficial que provee Google para desarrollar aplicaciones de forma nativa. Anteriormente se usaba Eclipse para desarrollar en Android, pero debido a que Android Studio se lanzó de forma oficial en su versión estable, se empezó a utilizar esta.

1.3.2 Java

Java es la base para prácticamente todas las aplicaciones de red, para desarrollo de aplicaciones móviles y embebidas, juegos y software empresarial. Java está en todas partes, centros de datos, computadoras, teléfonos, consolas de juegos, internet, etc.

- Java en el 97% de equipos empresariales.
- 9 millones de desarrolladores Java en el mundo.
- Java se ejecuta en 3 mil millones de móviles.

Muchos de los desarrolladores usan Java porque ha sido probado, ajustado y ampliado por la comunidad de Java.(Java, 2015).

Orientado a objetos

La programación orientada a objetos se acerca a como expresar las cosas en la vida real. Por ejemplo un automóvil tiene una serie de características que podrían ser el color, marca y modelo. También posee funcionalidades como

pueden ser acelerar, detenerse, estacionarse, etc. Por lo tanto en el esquema POO el automóvil es el objeto, las propiedades serían color, marca y modelo. Los métodos serían las funcionalidades acelerar, detenerse y estacionarse. (Alvarez, 2001).

Independencia de la plataforma

La independencia de plataforma significa que el programa se va a escribir una sola vez y se pueda ejecutar en cualquier entorno. Para esto el código Java se compila y se genera un archivo de extensión .class el cual contiene *bytecodes*. Esto es un código que está a “medio camino” entre el código fuente y el código máquina que entiende el dispositivo destino. Entonces el bytecode es ejecutado en la máquina virtual Java (JVM), que interpreta y ejecuta el código.

> La razón de usar Java es porque es el lenguaje de programación oficial que usa Google para crear aplicaciones nativas.

1.3.3 REST

REST (Representational State Transfer) es un estilo de arquitectura de software para el diseño de aplicaciones de red. Es una arquitectura bastante simple y eficiente, ya que está basada en llamadas http para realizar la transferencia de datos entre cliente y servidor, tal como lo hace la World Wide Web (www).

REST es muy versátil, ya que la información esta publicada o es accedida a través de entidades llamadas recursos. Por recursos se entiende la representación de un concepto de negocio. Cada recurso posee un estado interno, que no puede ser accedido directamente desde el exterior. Lo que sí es accesible desde el exterior es una o varias representaciones de dicho estado. La representación es el formato en que los datos son tranferidos entre el cliente y el servidor (ej. HTML, XML y JSON). Tanto los recursos REST como

sus diferentes representaciones son accedidos mediante un identificador único de recurso (URI).

Las aplicaciones REST o RESTful utilizan solicitudes u operaciones http para inter-operar con los recursos publicados dentro de esta arquitectura y son los siguientes:

- POST: Permite crear nuevos recursos.
- GET: Permite la lectura de recursos o búsqueda.
- PUT: Permite la actualización de recursos.
- DELETE: Permite la eliminación de recursos.

Esta arquitectura de comunicación actualmente es una de las preferidas por sus grandes ventajas como:

- Al seguir los principios de http, se puede aprovechar toda la infraestructura que este ofrece, como: cache, proxies, corta-fuegos, compresión, etc.
- Independiente del sistema operativo y de lenguaje de programación del cliente y servidor.
- Múltiples representaciones de datos o recursos.
- La implementación y mantenimiento del código es bastante simple.
- Recursos y acciones auto-descubribles, no es necesario un WSDL.

(Sindikos, 2011).

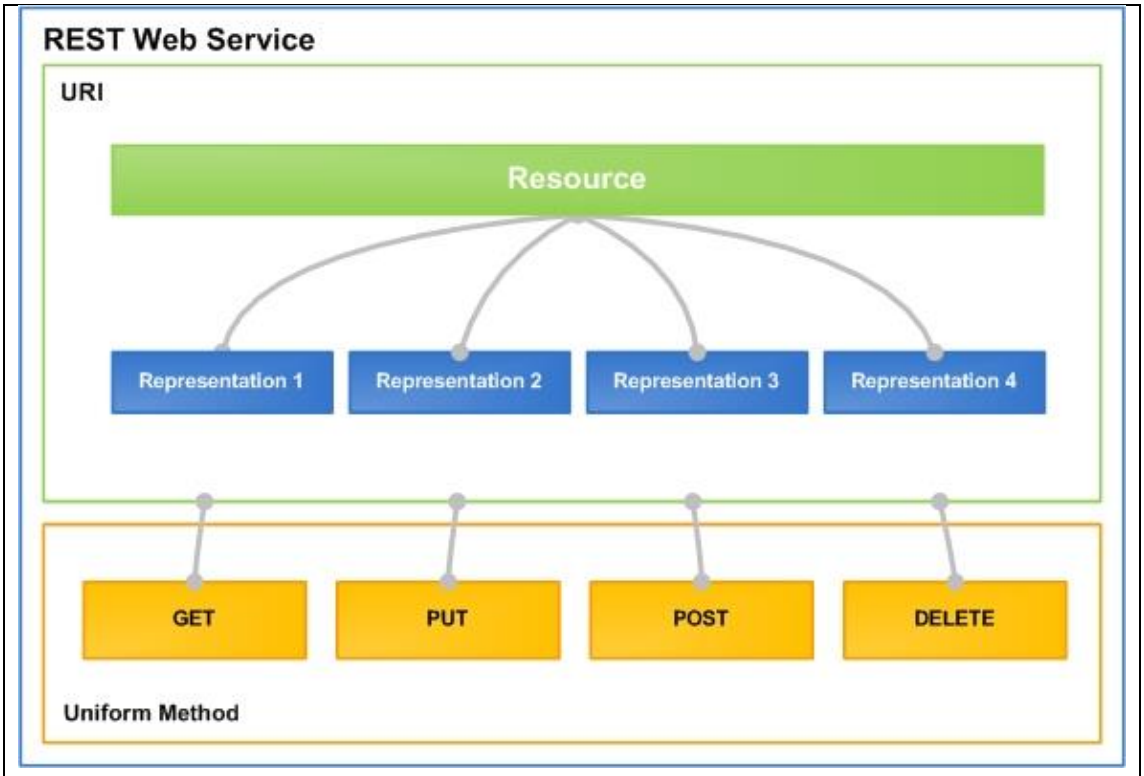

Figura 5. Arquitectura REST
Tomado de (sindikos, s.f)

1.3.4 Ruby

Ruby es un lenguaje de programación dinámico, relativo, orientado a objetos y de propósito general. Suporta múltiples paradigmas de programación como funcional, orientado a objetos e imperativa. En Ruby todo es un objeto, desde los tipos de datos y hasta las clases. (Matsumoto, 2000).

Características

- Orientado a objetos
- Niveles de variable: global, clase, instancia y local
- Manejo de excepciones
- Recolector de basura automático
- Altamente portable

- Hilos de ejecución simultáneos
- Introspección, reflexión y meta programación
- Inyección de dependencias
- Alteración de objetos en tiempo de ejecución
- Amplia librería estándar, incluyendo módulos para YAML, JSON, XML, OpenSSL, http, FTP, etc. Posee licencia de software libre GPL.

Se va a utilizar Ruby en este proyecto para realizar el servicio web. A diferencia de otros lenguajes de programación como Java o PHP, Ruby fue desarrollado para proyectos web y poder realizarlos en un tiempo muy corto, además de la fácil sintaxis que posee.

1.3.5 HTML

Son las siglas de HyperText Markup Language y es un estándar que sirve para la elaboración de páginas web. HTML se escribe en forma de etiquetas (<>) y puede describir en cierto punto la apariencia de la página. También puede hacer una referencia a un tipo de programa llamado script, el cual puede modificar el funcionamiento de la página. (Luján, 2001).

Elementos

Los elementos tienen 2 propiedades, atributo y contenido. Un elemento generalmente tiene una etiqueta de inicio y una de cierre (<html> </html>). Elementos como
 no tienen una etiqueta de cierre.

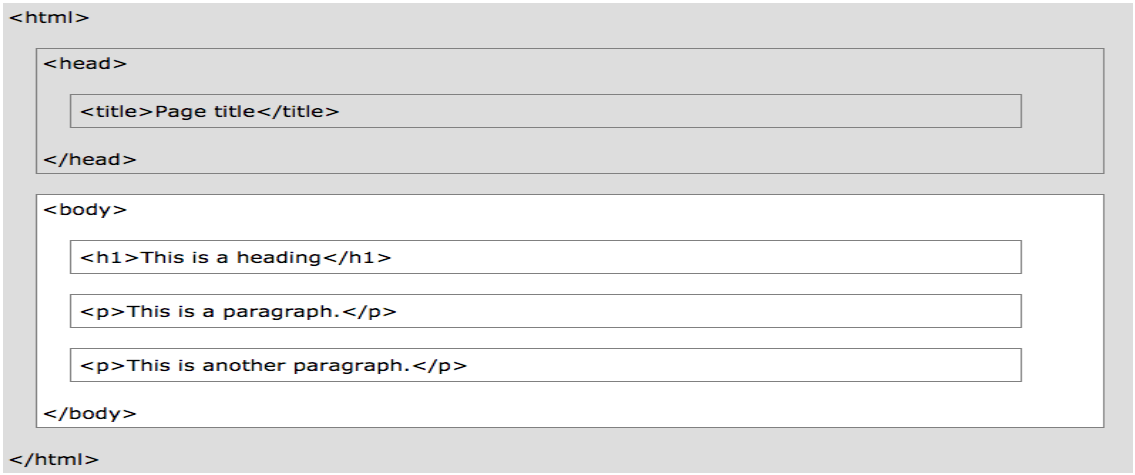

Figura 6. Estructura general de código HTML.

Tomado de (w3Schools, s.f)

1.3.6 JAVASCRIPT

Es un lenguaje de programación interpretado definido como orientado a objetos, usa el estándar ECMAScript. Se usa generalmente en el lado del cliente (FrontEnd) con el objetivo de brindar funcionalidad. Todos los navegadores web actuales incluyen JS.

Características

Imperativo y estructurado: JavaScript es compatible con gran parte de la estructura de programación C. Una diferencia de sintaxis con respecto a C es la inserción de punto y coma, es decir, en JavaScript se puede omitir el punto y como al final de una sentencia.

Funcional: Posee propiedades y métodos, como .call() o .bind(). También posee funciones anidadas, es decir, una función que está definida dentro de otra.

Prototípico: JavaScript usa prototipos para la herencia en lugar de clases.

Los usos más frecuentes de JavaScript son escribir funciones en las páginas HTML que interactúan con el DOM (Document Object Model). Algunos de los ejemplos son:

- Cargar nuevo contenido en la página o enviarla al servidor mediante llamadas AJAX sin necesidad de recargar la página.
- Animación de los elementos de la página.
- Validación de valores de entrada como un formulario .

Seguridad

JavaScript permite que existan programadores que hagan uso inapropiado del lenguaje como inyectar scripts que ejecuten código malicioso, de esta manera la página realizará acciones sin el consentimiento del usuario. Existen proyectos como AdSafe o Secure ECMA Script que brindan mayores niveles de seguridad en especial en código desarrollado por terceros.
(RFC4329, 2014).

1.3.7 SQLite

SQLite es una pequeña librería de código que implementa un motor SQL de base de datos **auto-contenido, sin-servidor, cero-configuración, y transaccional**. El código de esta librería es de dominio público, por lo tanto es libre para cualquier propósito, comercial o privado.

- Auto-contenido: Requiere soporte muy mínimo de bibliotecas externas o del sistema operativo. Esto hace que sea muy adecuado para su uso en dispositivos integrados que carecen de la infraestructura de soporte de un computador. Está escrito en ANSI-C y debe ser fácilmente codificado por cualquier compilador C estándar. Hace un uso mínimo de la biblioteca estándar de C.

- No requiere servidor y es cero-configuración: No requiere de un proceso o de un proceso intermedio de comunicación, como TCP/IP, para enviar y recibir datos, lo hace directamente al archivo de base de datos que se encuentra en el disco duro del dispositivo. La principal ventaja de trabajar bajo esta arquitectura es que no hay un proceso separado para instalar, configurar, inicializar, administrar y solucionar errores. Dado esto SQLite es un motor de base de datos que necesita cero-configuración.
- Transaccional: SQLite implementa transacciones serializables que son atómicas, consistentes, aisladas y durables (ACID, siglas en inglés), incluso si la transacción es interrumpida por un fallo del programa, una caída del sistema operativo. De esta manera SQLite cumple los pilares fundamentales a lo que refiere una transacción. (sqlite, 2015).

1.3.8 PostgreSQL

PostgreSQL Es un sistema de gestión de base de datos relacional orientado a objetos. Como es un proyecto de código abierto, PostgreSQL no es manejado por una empresa o persona. Las características más importantes son:

- Soporta distintos tipos de datos.
- Herencia entre tablas.
- Copias de seguridad en caliente.
- Unicode
- Múltiples métodos de autentificación.
- Acceso encriptado vía SSL.
- Licencia BSD
- Disponible para Linux, Unix, Windows.

Alta concurrencia

Dispone de un sistema denominado MVCC (Acceso concurrente multi versión), el cual mientras un proceso escribe en una tabla, otras puedan acceder a la misma sin bloqueos. (postgresql, 2016).

1.4 Plataformas

A continuación se describen las plataformas utilizadas en el desarrollo del proyecto:

1.4.1 Android 4.1

Android 4.1 es la primera versión de Jelly Bean, sistema operativo desarrollado por Google. Tiene un rendimiento mejorado comparado con versiones previas además de una interfaz más intuitiva y una latencia más baja del touch. Entre las características más relevantes se encuentran las siguientes:

- Accesibilidad mejorada: mejor administración de gestos y movimientos del usuario en el *touch screen*.
- Notificaciones expansibles: las aplicaciones ahora pueden mostrar más largas, las cuales pueden ser expandidas o borradas con un toque o un desplazamiento.
- Tamaño variable en widgets de aplicaciones: se introduce una mejora en el cambio de tamaño automático.
- Android Beam: tecnología basada en NFC para compartir cualquier tipo de archivo.
- Administración de ancho de banda: ayuda a las aplicaciones a manejar apropiadamente el uso de datos.
- Android browser y Webview: introduce mejoras en videos HTML5, CSS3 y CANVAS. (developer.android, 2016).

La razón de seleccionar esta versión de sistema operativo es porque no existe una pérdida significativa en cuanto a cobertura de dispositivos, ya que más del 80% de dispositivos Android utilizan esta versión. En el caso de escoger la

versión 5.0 se tendría un cobertura del 20% de dispositivos en todo el mundo. (developer.android, 2016).

1.4.2 Servidor Web JBoss WildFly

WildFly es un servidor web comúnmente conocido como JBoss AS o simplemente JBoss y es desarrollado por Red Hat. Está escrito en Java e implementa la plataforma Java, Java EE y corre en múltiples plataformas. Además es software libre y open source.

Características principales

- API de desarrollo
- Cuenta con JavaBeans versión 3 y 2.1
- Integra Hibernate para realizar la persistencia a la base de datos.
- Incluye autenticación Java y servicio de autorización.
- Integración con Java Message Service (JMS)
- Incluye Java Naming and directory interface (JNDI)
- JSP
- JAX-WS para web services tipo SOAP
- Conectores JDBC
- Soporta el ultimo estándar JAX-RS 2 para Web services REST
- *Framework* Arquillan para realizar testing unitarios
  (wildfly, 2016).

2. Capítulo II. Análisis y Diseño

En este capítulo se va a realizar un análisis de los requerimientos para el desarrollo de la aplicación usando la metodología Scrum y el diseño de las interfaces.

2.1. Product Backlog

Esta herramienta o artefacto de la metodología SCRUM, se utiliza para documentar las funcionalidades que se espera que otorguen a los usuarios finales, es decir las historias de usuario, basadas en las ideas del propio cliente. Una historia de usuario es un requerimiento del cliente.

Cada historia de usuario presenta los siguientes campos:

- ID: Identificador único de cada historia de usuario.
- Historia: Descripción de la funcionalidad en palabras del cliente.
- Criterio de aceptación: Lo que de cumplir la historia para que sea aceptada por el usuario.
- Prioridad: El nivel de importancia que el cliente desea frente a esta historia.
- Estado: El estado que el equipo de desarrollo define.
- Puntos: Se mide en puntos el esfuerzo que toma realizar la historia de usuario.

Historias de Usuario

A continuación se presentan las historias de usuario obtenidas del alcance del proyecto.

Tabla 2. Historia de Usuario 1

ID: APP1	
HISTORIA: Como usuario, se requiere una interfaz para creación de una cuenta.	
CRITERIO DE ACEPTACION La interfaz debe contar con un campo para username, email y password	
Prioridad:	alta / **media** / baja
Esfuerzo:	5
Estado:	Not started / Doing / Done

Tabla 3. Historia de Usuario 2

ID: APP2	
HISTORIA: Como usuario se requiere una interfaz para inicio de sesión en la aplicación	
CRITERIO DE ACEPTACION La interfaz debe contar con un campo para email y password	
Prioridad:	alta / **media** / baja
Esfuerzo:	4
Estado:	Not started / Doing / Done

Tabla 4. Historia de Usuario 3

ID: APP3	
HISTORIA: Como usuario se requiere una interfaz para poder medir el ruido de la ubicación actual	
CRITERIO DE ACEPTACION Se debe presentar el nivel de ruido en decibeles en tiempo real	
Prioridad:	**alta** / media / baja
Esfuerzo:	8
Estado:	Not started / Doing / Done

Tabla 5. Historia de Usuario 4

ID: APP4	
HISTORIA: Como usuario se requiere una opción para guardar las mediciones realizadas	
CRITERIO DE ACEPTACION Se guardarán externa y localmente	
Prioridad:	**alta** / media / baja
Esfuerzo:	5
Estado:	Not started / Doing / Done

Tabla 6. Historia de Usuario 5

ID: APP5	
HISTORIA: Como usuario se requiere una interfaz con un mapa de calor donde se pueda visualizar las mediciones realizadas por el usuario	
CRITERIO DE ACEPTACION Mostrar el valor de la medición en cada punto	
Prioridad:	**alta** / media / baja
Esfuerzo:	6
Estado:	Not started / Doing / Done

Tabla 7. Historia de Usuario 6

ID: MDC1	
HISTORIA: Como usuario se requiere una interfaz de inicio de sesión para el sistema web.	
CRITERIO DE ACEPTACION La interfaz debe contar con un campo para usuario y otra para contraseña.	
Prioridad:	alta / **media** / baja
Esfuerzo:	4
Estado:	Not started / Doing / Done

Tabla 8. Historia de Usuario 7

ID: MDC2	
HISTORIA: Como usuario se requiere una interfaz web donde se pueda consultar las mediciones realizadas por un rango de fechas.	
CRITERIO DE ACEPTACION Visualizar mediciones en una tabla	
Prioridad:	alta / **media** / baja
Esfuerzo:	5
Estado:	Not started / Doing / Done

Tabla 9. Historia de Usuario 8

ID: MDC3	
HISTORIA: Como usuario se requiere una interfaz web donde muestre un mapa de calor para visualizar todas las mediciones realizadas	
CRITERIO DE ACEPTACION Visualizar mediciones de todos los usuarios	
Prioridad:	**alta** / media / baja
Esfuerzo:	6
Estado:	Not started / Doing / Done

2.2. Sprint Backlog

Esta herramienta de la metodología SCRUM se usa para dividir las historias de usuario en tareas a las cuales se les asignan ciertas horas de trabajo. Cada miembro del equipo escoge los sprints más adecuados para cada uno.

Contiene los siguientes campos:

- Id de historia de usuario.
- Actividades a realizar durante la historia de usuario.
- Persona responsable.

Sprint 1

Este sprint contendrá la interfaz móvil de registro de usuario y el servicio web.

Tabla 10. Sprint 1

APP1	
ACTIVIDADES	**RESPONSABLE**
Crear interfaz de registro con los campos correspondientes	Miguel Amores
Crear servicio REST tipo POST para enviar datos al servidor	Miguel Amores

Sprint 2

Este sprint contendrá la interfaz móvil de *login* y el servicio web.

Tabla 11. Sprint 2

APP2	
ACTIVIDADES	**RESPONSABLE**
Crear interfaz de *login* con campos	Miguel Amores
Crear servicio REST tipo GET de usuario	Miguel Amores
Consumir servicio REST y validar usuario	Miguel Amores

Sprint 3

Este sprint contendrá la interfaz de medición de ruido con su funcionalidad, uso del gps y registro en la base de datos.

Tabla 12. Sprint 3

APP3	
ACTIVIDADES	**RESPONSABLE**
Crear interfaz y funcionalidad para medir el ruido mediante el micrófono del celular	Miguel Amores
Crear funcionalidad de los botones *play*, *stop* y *save*	Miguel Amores
Crear funcionalidad para obtener las coordenadas geográficas	Miguel Amores

Sprint 4

Este sprint contendrá la interfaz móvil con el mapa de calor.

Tabla 13. Sprint 4

APP4	
ACTIVIDADES	**RESPONSABLE**
Crear una interfaz con un mapa de calor para visualizar las mediciones realizadas en cada ubicación	Miguel Amores
Consultar las mediciones realizadas por usuario logueado y visualizarlas en el mapa	Miguel Amores

Sprint 5

Este sprint contendrá todo el sistema web desde la interfaz de *login*, la interfaz de consulta de mediciones y el mapa de calor.

Tabla 14. Sprint 5

MDC	
ACTIVIDADES	**RESPONSABLE**
Crear interfaz de inicio de sesión con su funcionalidad.	Miguel Amores
Crear interfaz de reporte de mediciones.	Miguel Amores
Modificar servicio web para búsqueda de mediciones con rango de fechas.	Miguel Amores
Crear una interfaz web con un mapa de calor donde pueda visualizar las mediciones realizadas por todos los usuarios	Miguel Amores

2.3. Diagrama de Secuencia

A continuación se presenta el diagram de secuencia para visualizer el comportamiento de las aplicaciones web y móvil ante las entradas generadas por el usuario.

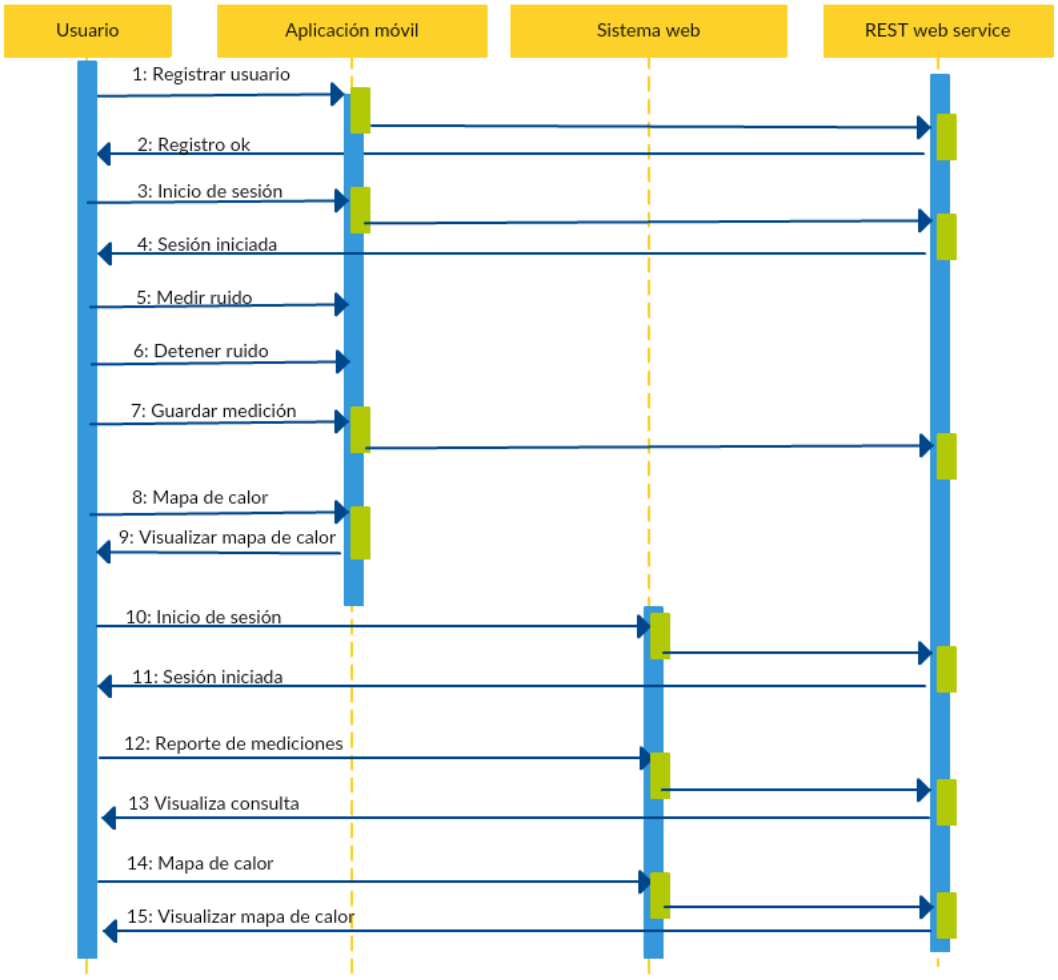

Figura 7. Diagrama de secuencia

2.4. Diagramas de Clases UML

A continuación se presenta el diagrama de clases en el cual se muestra todos los objetos relacionados en el proyecto, así como sus atributos y métodos

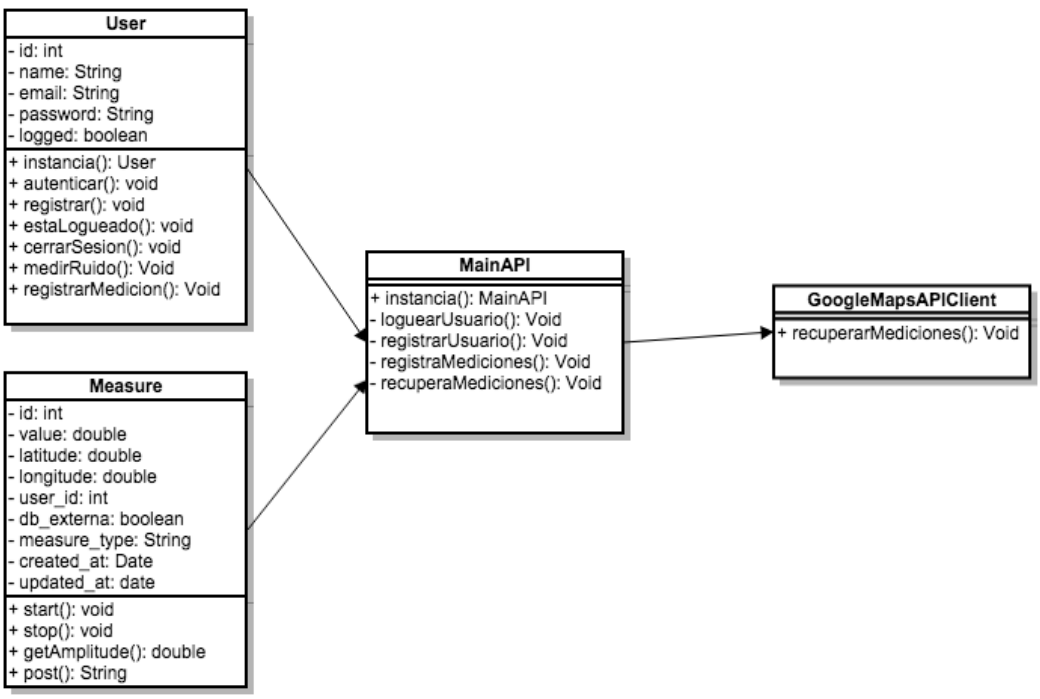

Figura 8. Diagrama de Clases

2.5. Diagrama de Base de Datos

A continuación se presenta el diagrama que contiene la base de datos usada en el proyecto.

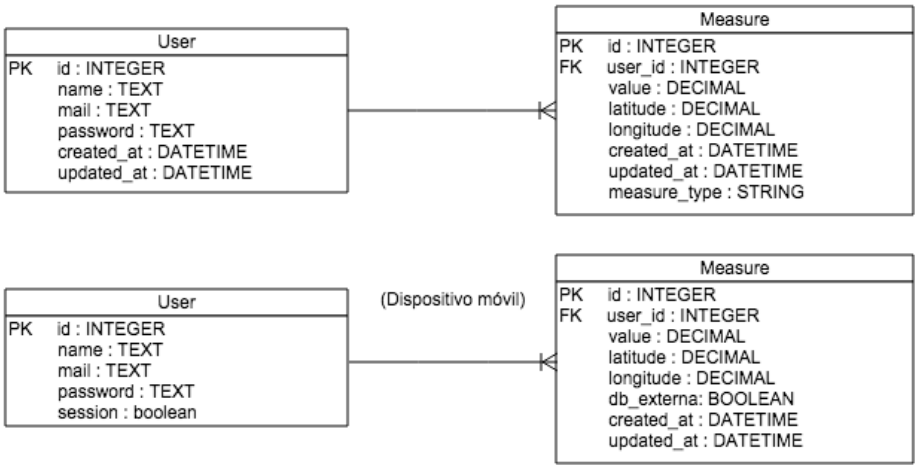

Figura 9. Diagrama de Base de Datos

2.6. Descripción de Servicio Web RESTful

El servicio web en general lo que hace es actuar como intermediario entre la base de datos y la aplicación móvil/sistema web. Atiende a la petición de un recurso y responde a la misma.

Figura 10. Arquitectura de Servicio Web REST

En la siguiente tabla se especifica el servicio web para la autenticación del usuario, tanto para la aplicación móvil y el sistema web.

Tabla 15. Servicio REST GET user

GET user	
URL	https://polar-fjord-2695.herokuapp.com/user
Formato de respuesta	JSON
Objeto de respuesta	User
Método HTTP	GET
PARAMETROS	
mail	Email del usuario (requerido)
password	Password del usuario (requerido)
CODIGOS DE RESPUESTA	
200 (OK)	Usuario encontrado y devuelve un objeto User
404 (Not Found)	Usuario no encontrado y devuelve status 404

Ejemplo de Petición:

GET:
http://192.168.1.22:3000/user?mail=miguelamores@gmail.com&password=12345

Respuesta:


```
{
    "id":1,
    "name":"miguelamores",
    "mail":"miguelamores@gmail.com",
    "password":"12345",
    "created_at":"20150814T02:33:08.051Z",
    "updated_at":"20150814T02:33:08.051Z"
}
```

En la siguiente tabla se especifica el servicio web para el registro del usuario en la aplicación móvil.

Tabla 16. Servicio REST POST user

POST user	
URL	https://polar-fjord-2695.herokuapp.com/user
Formato de respuesta	JSON
Objeto de respuesta	User
Método HTTP	POST
PARAMETROS	
username	Alias de usuario
mail	Email del usuario (requerido)
password	Password del usuario (requerido)
CODIGOS DE RESPUESTA	
200 (OK)	Usuario encontrado y devuelve un objeto User
404 (Not Found)	Usuario no encontrado y devuelve status 404

Ejemplo de Petición:

POST: http://192.168.1.22:3000/user/

POST DATA:

username=miguel&
mail: miguelamores@gmail.com&
password=12345

{
"id":1,
"name":"miguelamores",
"mail":"miguelamores@gmail.com",
"password":"12345"
}

En la siguiente tabla se especifica el servicio web para el registro de una nueva medición realizada desde la aplicación móvil.

Tabla 17. Servicio REST POST measure

POST measure	
URL	https://polar-fjord-2695.herokuapp.com/measure
Formato de respuesta	JSON
Objeto de respuesta	Código HTTP
Método HTTP	POST
PARAMETROS	
user_id	ID de usuario
value	Medición en decibeles
latitude	Coordenada geográfica
longitude	Coordenada geográfica
created_at	Fecha de creación
CODIGOS DE RESPUESTA	
201 (CREATED)	Medición registrada y devuelve código de respuesta

Ejemplo de Petición:

POST: http://192.168.1.22:3000/measure/

POST DATA:

value=75.5&
latitude=-0.180653&
longitude=-78.467834&
user_id=1

```
{
        "id":1,
        " value ":"75.5",
        "latitude":"-0.180653",
        "longitude":"-78.467834",
        " user_id ":"1"
}
```

En la siguiente tabla se especifica el servicio web para consultar todas las mediciones realizadas por los usuarios y ser usadas en el mapa de calor web. Los parámetros son usados para la consulta con rango de fechas en el reporte del sistema web.

Tabla 18. Servicio REST GET measure

GET measure	
URL	https://polar-fjord-2695.herokuapp.com/measure
Formato de respuesta	JSON
Objeto de respuesta	Código HHTP
Método HTTP	GET
PARAMETROS (Reporte sistema web)	
created_at	Fecha de inicio
created_at	Fecha de fin
CODIGOS DE RESPUESTA	
200 (OK)	Encontradas todas las mediciones

Ejemplo de Petición:

GET: http://192.168.1.22:3000/measure/


```
[
        {
                "id":1,
                " value ":"75.5",
                "latitude":"-0.180653",
                "longitude":"-78.467834",
                " user_id ":"1"
        },
        {
                "id":2,
                " value ":"60.0",
                "latitude":"-0.180674",
                "longitude":"-78.467898",
                " user_id ":"2"
        }
]
```


2.7. Interfaces Gráficas

A continuación se presentan las interfaces gráficas tanto de la aplicación móvil como del sistema web.

2.6.1 Aplicación Móvil

Registro de usuario

Se utiliza para crear una nueva cuenta de usuario

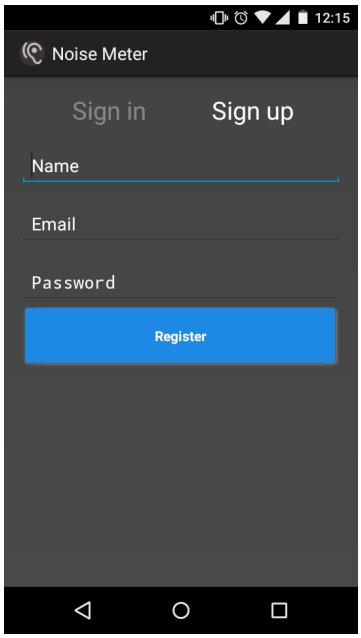

Figura 11. Interfaz de registro de usuario.

Inicio de sesión

Se utiliza para ingresar a la aplicación con una cuenta creada previamente.

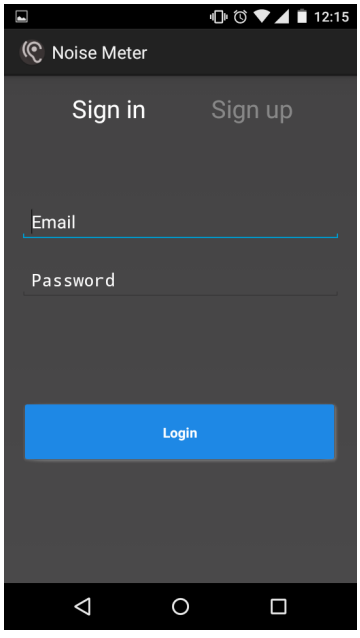

Figura 12. Interfaz de inicio de sesión.

Medición de ruido

Se utiliza para visualizar la medición de ruido.

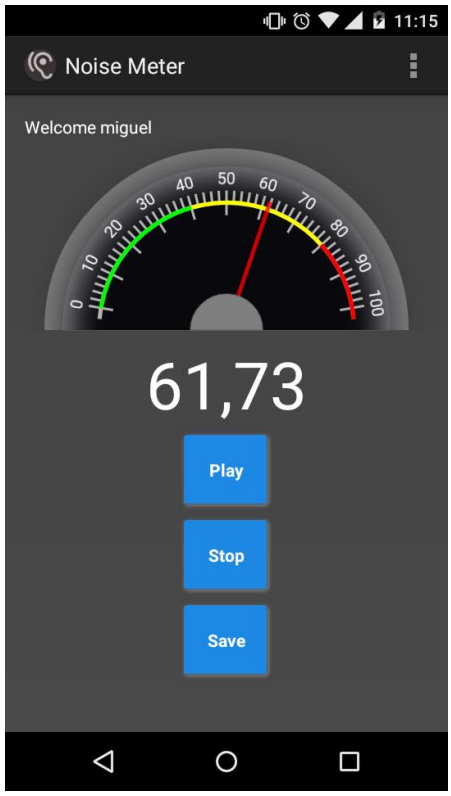

Figura 13. Interfaz de medición de ruido.

Submenú

El submenú sirve para mostrar el mapa de calor y para cerrar sesión.

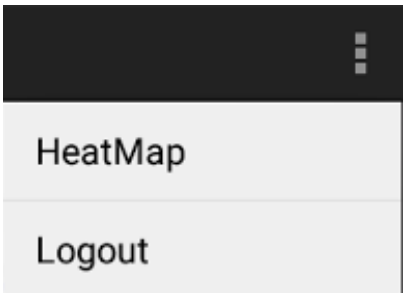

Figura 14. Submenú

Notificaciones

Las notificaciones serán mostradas al momento de realizar cierta acción.

Se muestra cuando los datos de registro no son correctos o están incompletos.

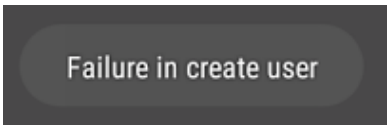

Figura 15. Falla de registro de usuario

Se muestra cuando las credenciales del usuario no son correctas o están incompletas.

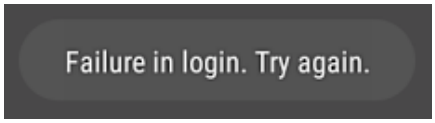

Figura 16. Falla de inicio de sesión de usuario

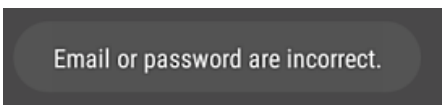

Figura 17. Falla de inicio de sesión de usuario

Se muestra cuando la medición fue registrada en la base de datos del dispositivo móvil.

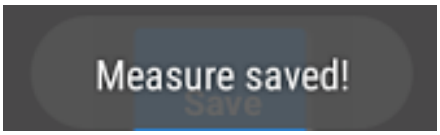

Figura 18. Medición registrada localmente

Se muestra cuando la medición fue registrada en la base de datos externa.

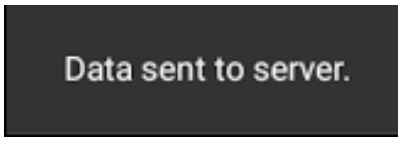

Figura 19. Medición registrada externamente

Se muestra cuando se están registrando mediciones locales en la base de datos global.

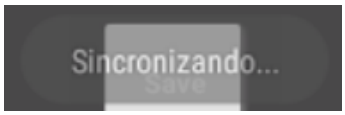

Figura 20. Medición registrada externamente al iniciar aplicación.

Speedometer

Se usa para mostrar el cambio en un valor en una gráfica similar a un velocímetro.

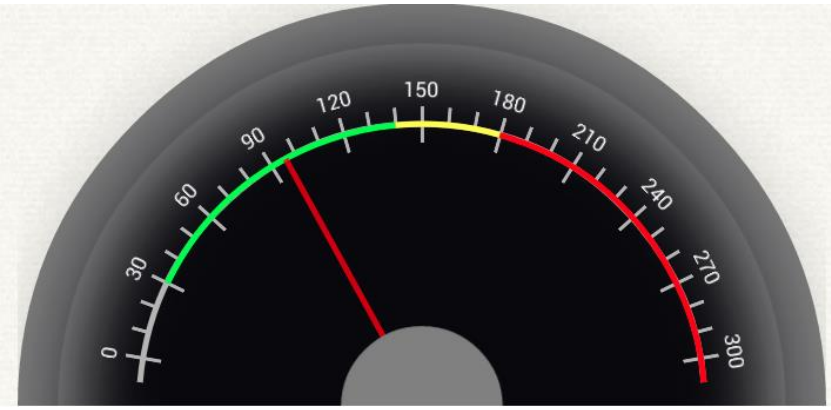

Figura 21. Speedometer

Mapa de calor móvil

Un mapa de calor muestra colores en el mapa para representar la densidad de puntos. En este caso se usa para mostrar las mediciones guardadas localmente en el dispositivo.

Figura 22. Mapa de calor de aplicación móvil

Adaptado de (googlemaps, s.f)

2.6.2 Sistema Web

Interfaz de autenticación

La siguiente interfaz se usa para acceder al sistema web, usando las credenciales registradas en la aplicación móvil.

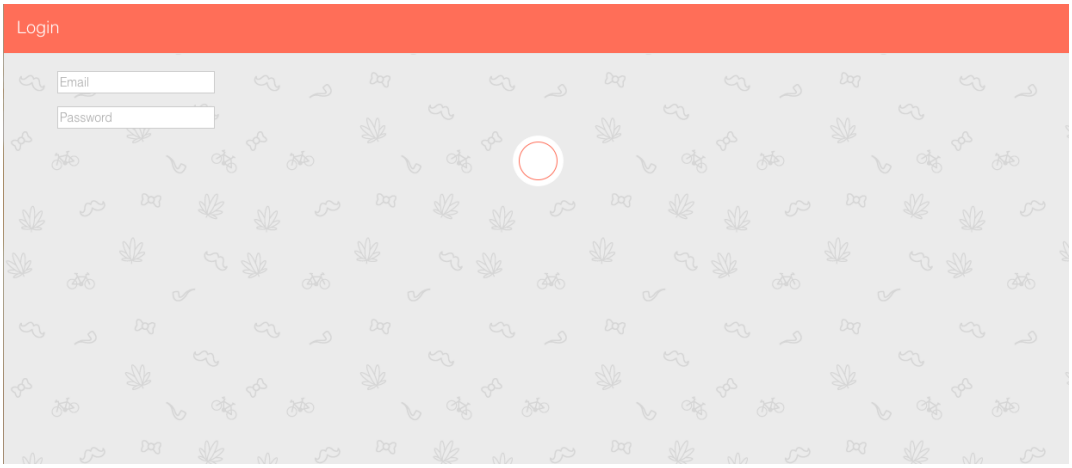

Figura 23. Interfaz de inicio de sesión del sistema web.

Interfaz de consulta de mediciones

La siguiente interfaz se usa para realizar consultas de las mediciones realizadas por un rango de fechas. Desde esta página se accede al mapa de calor mediante el botón “Go to map”.

Figura 24. Interfaz de reporte de mediciones.

Mapa de calor web

Se usa para mostrar las mediciones realizadas por todos los usuarios.

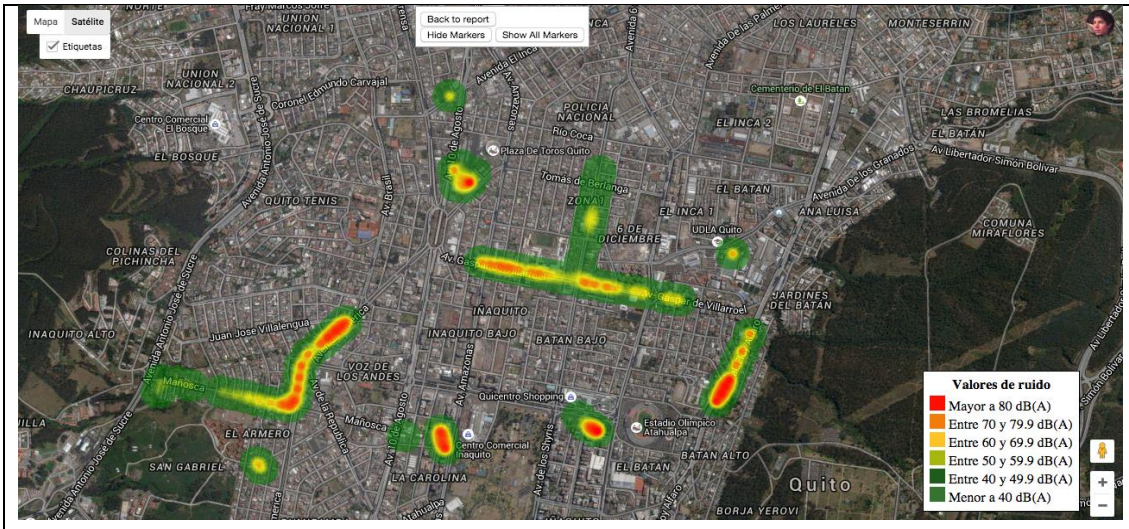

Figura 25. Mapa de calor de aplicación web

Adaptado de (googlemaps, s.f)

3 Capitulo III. Implementación y Pruebas

En este capítulo se explicará el proceso de implementación del proyecto, cómo se aplicó SCRUM para el mismo y las pruebas realizadas.

3.1 Sprint (Ejecución de la iteración)

Esta técnica fue aplicada en el Product Backlog, especificando cada tarea que se debía realizar en las iteraciones. En la figura se puede apreciar que la historia de usuario se dividió en 2 actividades, las cuales van a formar parte del primer *sprint* o iteración. Cabe resaltar que un *sprint* puede contener actividades de diferentes historias de usuario.

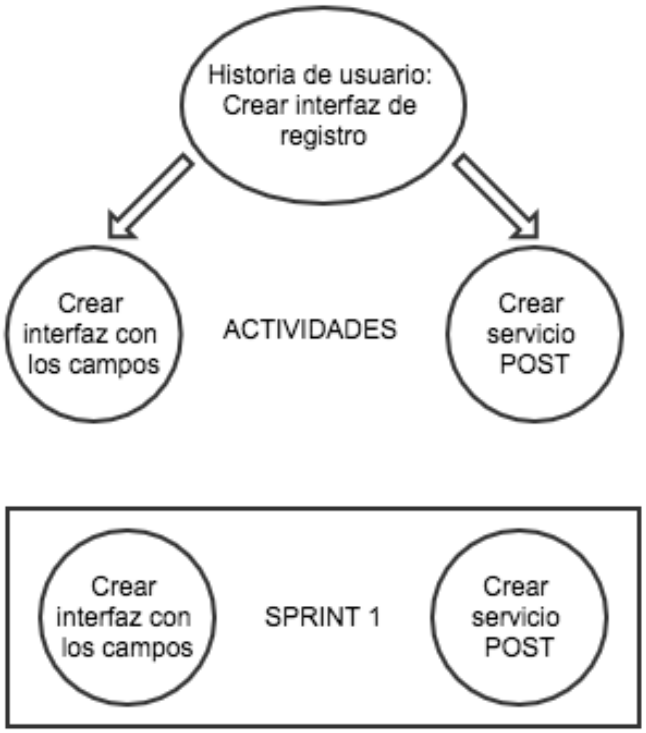

Figura 26. Sprint.

3.2 Sprint Planning (Planificación de la iteración)

En esta actividad, el dueño del producto le presenta al equipo de trabajo las historias de usuario prioritarias. Entonces es cuando ambos definen cuales van a ser desarrolladas en el sprint a planificar.

Tabla 19. Priorización de Historias de usuario

HISTORIA DE USUARIO	PRIORIDAD
APP1	1
APP2	2
APP3	3
APP4	4
APP5	5
MDC3	6
MDC2	7
MDC1	8

3.3 Scrum Daily Meeting

Esta técnica que introduce la metodología SCRUM no se aplicó, debido a que el desarrollo de la aplicación fue elaborado por una sola persona.

3.4 Sprint Review (Revisión del sprint)

Esta técnica fue aplicada en cada reunión de tutoría, realizando una revisión de las actividades completadas y no completadas con el tutor asignado.

Tabla 20. Sprint review

SPRINT	FECHA	OBSERVACIÓN TUTOR
Sprint 1	16-06-2015	OK
Sprint 2	03-07-2015	El usuario debe realizar una sola vez la etapa de autenticación
Sprint 3	22-09-2015	OK
Sprint 4	06-11-2015	OK

Sprint 5	29-02-2016	Se debe acceder al sistema web mediante una autenticación

3.5 Sprint Retrospective (Retrospectiva)

Con el objetivo de mejorar su productividad y calidad en el proyecto que se está desarrollando, el equipo analiza la manera en que trabaja durante un sprint, porque se consiguen o no los objetivos que se comprometieron a un inicio.

Tabla 21. Sprint retrospective

SPRINT	¿Qué cosas han funcionado bien?	¿Qué es lo que hay que mejorar?
1	El procedimiento de crear primero el servicio REST y probarlo antes de implementar en la aplicación.	Las interfaces gráficas se pueden mejorar.
2	El procedimiento de crear primero el servicio REST y probarlo antes de implementar en la aplicación.	Las interfaces gráficas se pueden mejorar.
3	El procedimiento de crear primero el servicio REST y probarlo antes de implementar en la aplicación.	Se puede mejorar la administración del GPS cuando está desactivado.
4	Consultar las mediciones locales. Mostrar cada medición representada en el mapa de calor.	Se puede mejorar la forma de representar los puntos de medición representados en el mapa, ya que no se puede manejar una escala del radio de los puntos.
5	Inicio de sesión y consulta de mediciones por fechas. Representar las mediciones en el mapa de calor.	Se puede mejorar el tiempo de respuesta del login de un usuario.

3.6 Implementación

Aquí se va a explicar los pasos que se siguieron para implementar cada proyecto, la estructura de clases y el despliegue de las aplicaciones.

3.6.1 Desarrollo de app móvil

Lo primero que se implementó en el proyecto fue la aplicación móvil. Por lo tanto está estructurado de la siguiente manera:

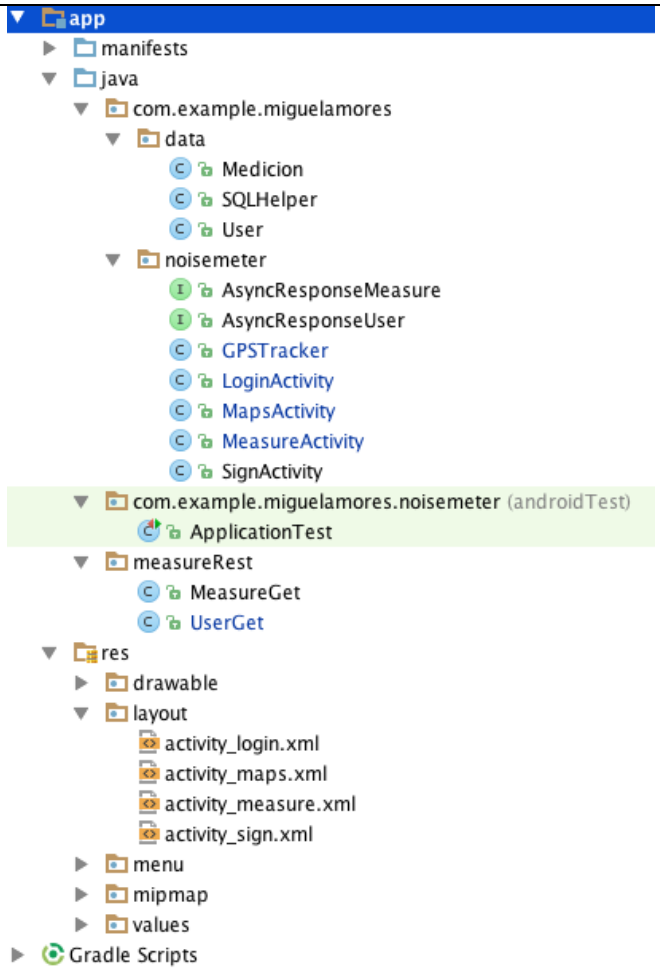

Figura 27. Estructura MVC del proyecto móvil

- com.example.miguelamores.data: En este paquete se encuentran los archivos java de modelo de las entidades y la gestión de base de datos del dispositivo móvil.

    - Medicion: Aquí se define los atributos y métodos de una medición.

    - User: Aquí se define los atributos y métodos de un usuario.
    - SQLHelper: Este es el archivo de base de datos. Aquí se define la tabla con los datos.

- com.example.miguelamores.noisemeter: En este paquete se encuentran los archivos java para registro de usuario, inicio de sesión, uso de gps, medición de ruido, dibujar mapa y mostrar datos sobre él.

    - GPSTracker: Aquí se define el funcionamiento del gps para obtener las coordenadas geográficas.
    - SignActivity: Este archivo tiene la función para realizar el registro del usuario en la base de datos.
    - LoginActivity: Este archivo tiene la función para realizar el inicio de sesión del usuario.
    - MeasureActivity: Este archivo tiene las funciones para usar el micrófono del dispositivo, obtener los valores medidos y convertirlo en ruido medido en decibelios. Parte del código se tomó del foro http://stackoverflow.com/questions/14181449/android-detect-sound-level#95. Además aquí se realiza el registro de las mediciones.
    - MapsActivity: Este archivo tiene la función de inicializar el mapa, obtener las mediciones y mostrarlas al usuario.

- com.example.miguelamores.measureRest: En este paquete se encuentran los archivos java para invocar el los servicios Rest del usuario y la medición.

    - MeasureGet: Este archivo se usa para llamar al servicio Rest para obtener las mediciones realizadas.
    - UserGet: Este archivo se usa para llamar al servicio Rest de usuario y poder encontrarlo.

- res: En esta carpeta se encuentran los archivos xml para graficar las interfaces de la aplicación móvil.

    - activity_login: Este archivo se usa para graficar la interfaz de registro e inicio de sesión del usuario.
    - activity_measure: Este archivo se usa para graficar la interfaz de medición de ruido.
    - activity_maps: Este archivo se usa para mostrar el mapa de calor.

3.6.2 Servicio web y Base de Datos

El siguiente paso fue crear el servicio web y la base de datos utilizando Ruby on Rails. La ventaja de este *framework* es que nos ayuda a crear la base de datos por línea de comandos.

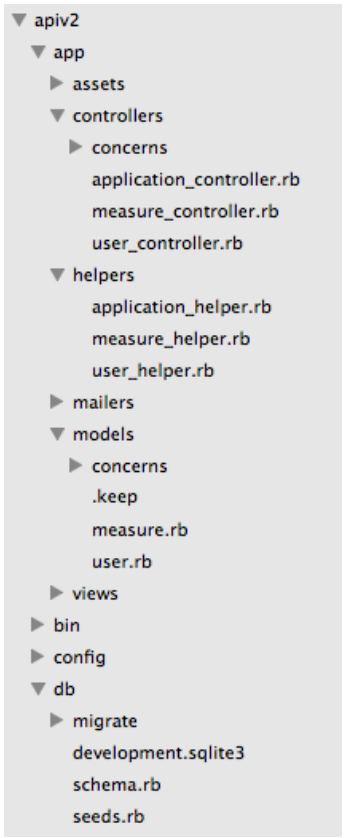

Figura 28. Estructura de clases del servicio REST

Para crear el proyecto se usan los siguientes comandos en una terminal:

- rails new api: Este comando se usa para crear la arquitectura del proyecto con el nombre “api”.
- rails generate controller user index: Este comando se usa para crear el archivo donde se realizarán los métodos de crear y leer usuario.
- rails generate model User name:text mail:text password:text: Este comando se usa para crear el archivo para generar la base de datos.
- rake db:créate y rake db:migrate: Estos comandoS se usan para generar la base de datos.

El proyecto quedó estructurado de la siguiente manera:

- apiv2/app/controllers: En esta carpeta se crean los archivos para realizar los métodos de crear, leer y actualizar de usuarios y mediciones.

    - measure_controller: Archivo que contiene operaciones CRUD de las mediciones.
    - user_controller: Archivo que contiene operaciones CRUD de los usuarios.

- apiv2/app/models: En esta carpeta se crean los archivos de cada entidad y donde se realizan las relaciones entre ellos.
- apiv2/app/config/database.yml: Este archivo es donde se encuentrar las configuraciones de base de datos como el host, usuario, contraseña, etc.
- Apiv2/app/db/schema.rb: Este archivo contiene el esquema de la base de datos.

3.6.3 Sistema Web

Después de haber finalizado tanto la aplicación móvil, el servicio web y la base de datos, se comenzó con el desarrollo del sistema web, el cual cuenta con la siguiente arquitectura:

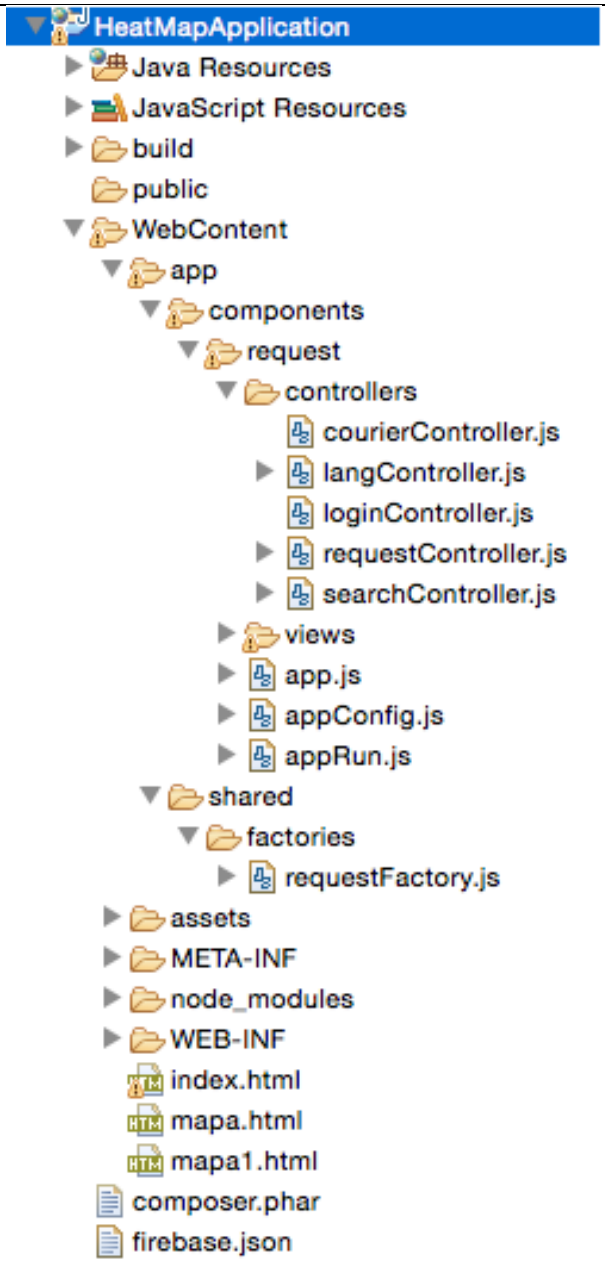

Figura 29. Estructura de clases y vistas del proyecto web.

- WebContent/app/components/request: Esta esta ruta se tienen archivos de configuración de la aplicación, la carpeta de controladores y la de vistas.

    - app.js: Es el módulo de la aplicación, donde se referencia los componentes que se van a utilizar.
    - appConfig.js: Este archivo contiene la configuración de rutas e idioma de la aplicación.

- WebContent/app/components/request/controllers: Son los archivos que contienen la lógica de negocio de cada página del sistema web.
- WebContent/app/components/request/views: Se encuentran los archivos HTML.
- WebContent/app/shared: Contiene los archivos con lógica de negocio para compartir a toda la aplicación web.

- factories/requestFactory: Archivo donde se realiza las llamadas a los servicios web.

3.6.4 Despliegue de aplicaciones

Una vez finalizadas todas las historias de usuario, se procedió a desplegar los servicios web y el sistema web en producción. En otras palabras, subir a la web para que sea accesible desde cualquier lugar.
Primero se despliega el servicio web, para eso en el terminal se usan los siguientes comandos:

- cd apiv2: El comando permite cambiarnos a la ruta del proyecto.
- git init: El comando inicializa un repositorio git.
- heroku git:remote –a miguelamoresheatmap
- git add .
- git commit –am “Comentario”

- git push heroku master: Estos 3 últimos comandos son los que realizan el despliegue del servicio web.

Seguidamente se realiza el despliegue del sistema web, para eso en el terminal se usan los siguientes comandos:

- npm install -g firebase-tools: Este commando se utiliza para instalar herramientas para poder usar el *hosting.*
- cd HeatMapApplication: El comando permite cambiarnos a la ruta del proyecto.
- firebase deploy: Este comando realiza el despliegue de la aplicación web

3.7 Pruebas de Aceptación

Las pruebas de aceptación se realizan al final de cada sprint para dar por finalizado el mismo. Se muestran a continuación:

En la siguiente tabla tenemos la prueba de creación de usuario desde la aplicación móvil.

Tabla 22. Prueba de aceptación 1

CREACION DE UNA CUENTA DE USUARIO		
ID Historia de usuario	APP1	
SECUENCIA DE LA PRUEBA		
Procedimientos	**Resultados esperados**	**Resultado**
Ingresar los datos para crear una nueva cuenta	Validar los datos ingresados y cambiar a interfaz de *login*	OK

En la siguiente tabla tenemos la prueba de inicio de sesión a la aplicación móvil.

Tabla 23. Prueba de aceptación 2

INICIO DE SESION A LA APLICACIÓN MOVIL		
ID Historia de usuario	APP2	
SECUENCIA DE LA PRUEBA		
Procedimientos	**Resultados esperados**	**Resultado**
Ingresar las credenciales de usuario previamente registrado	Validación de credenciales e ingreso a la aplicación	OK

En la siguiente tabla tenemos la prueba medición de ruido de la ubicación actual del usuario.

Tabla 24. Prueba de aceptación 3

MEDICION DE RUIDO		
ID Historia de usuario	APP3	
SECUENCIA DE LA PRUEBA		
Procedimientos	**Resultados esperados**	**Resultado**
Presionar el botón *play* de la aplicación	Visualizar los valores de ruido en pantalla. Mostrar dialogo de activación de gps en caso de estar desactivado	OK

A continuación se presentan las pruebas de funcionalidad de la aplicación que fueron realizadas en el laboratorio de acústica de la Universidad de Las Américas, con la finalidad de verificar que las medidas realizadas con el dispositivo móvil fueran similares a mediciones realizadas con un sonómetro. Las herramientas usadas en el laboratorio son las siguientes:

Sonómetro analizador de espectros serie SC310 con número de serie T238856, fue usado para comparar los ruidos medidos con el dispositivo móvil.

Figura 30. Sonómetro analizador de espectros.

La fuente generadora de ruido, como su nombre lo indica, fue utilizada para generar diversos ruidos los cuales fueron medidos con el sonómetro y el dispositivo móvil.

Figura 31. Generador de Ruido.

El parlante fue usado para emitir el ruido proveniente del generador de ruido, en un cuarto especial con paredes usadas para aislamiento acústico**,** es decir para evitar que el ruido se escape o ingrese a dicho cuarto.

Figura 32. Parlante y paredes.

En la primera prueba se emitió un ruido de aproximadamente 64 dB(A), donde se puede ver que el sonómetro marca 64.2 dB(A) y el dispositivo móvil 64.99 dB(A), una diferencia de medición relativamente baja.

Figura 33. Prueba de medición 1.

En la segunda prueba se emitió un ruido de aproximadamente 74 dB(A), donde se puede ver que el sonómetro marca 73.7 dB(A) y el dispositivo móvil 74.74 dB(A), un margen de error muy pequeño.

Figura 34. Prueba de medición 2.

En la tercera prueba se emitió un ruido de 87 dB(A), donde se puede ver que el sonómetro marca 87 dB(A) y el dispositivo móvil 86.77 dB(A), un margen de error aún más pequeño.

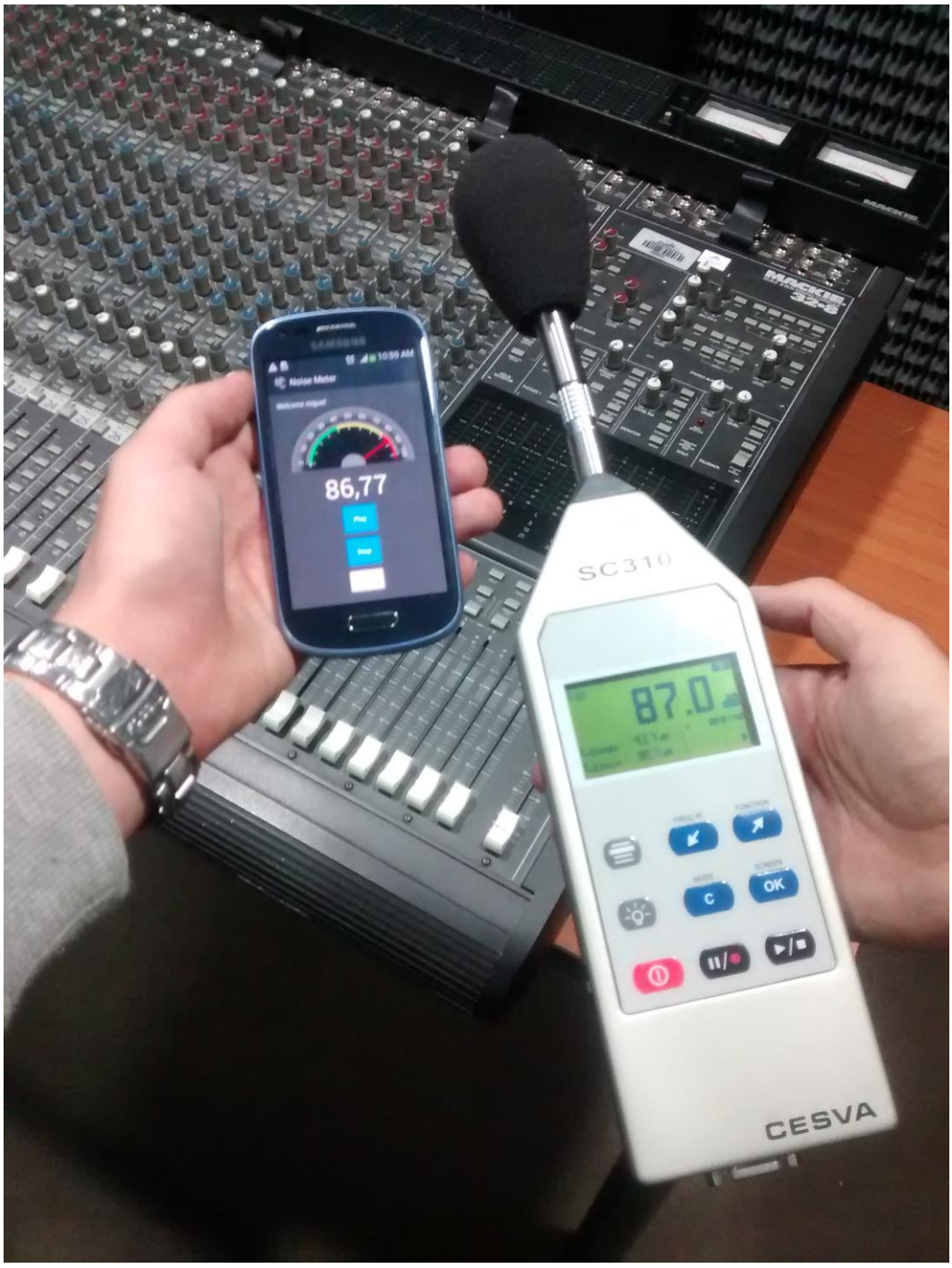

Figura 35. Prueba de medición 3.

En la cuarta prueba se emitió un ruido de aproximadamente 93 dB(A), donde se puede ver que el sonómetro marca 92.7 dB(A) y el dispositivo móvil 89.11 dB(A), un más grande que las anteriores.

Figura 36. Prueba de medición 4.

Todas estas pruebas fueron realizadas con ruidos entre las frecuencias de 300 a 1000 Hz, donde marcan aproximadamente la misma medición. Con frecuencias fuera de dicho rango, existió un margen de error demasiado grande.

Las siguientes pruebas fueron realizadas en ambientes reales donde se puede captar desde ruidos bajos hasta altos.

Prueba realizada en zona de trabajo, donde el sonómetro marca 51 dB(A) y el dispositivo móvil 49.1 dB(A).

Figura 37. Prueba de medición 5, zona laboral.

Prueba realizada en zona residencial, donde el sonómetro marca 60.8 dB(A) y el dispositivo móvil 60.72 dB(A).

Figura 38. Prueba de medición 6, zona residencial.

Prueba realizada en avenida transitada, donde el sonómetro marca 69.3 dB(A) y el dispositivo móvil 67.15 dB(A).

Figura 39. Prueba de medición 7, avenida transitada.

Prueba realizada en zona industrial, donde el sonómetro marca 91 dB(A) y el dispositivo móvil 89.02 dB(A).

Figura 40. Prueba de medición 8, zona industrial.

Pruebas realizadas con celular Sony xperia, donde se puede observar la gran diferencia con un celular de marca Motorola. Como se puede ver la diferencia es considerable.

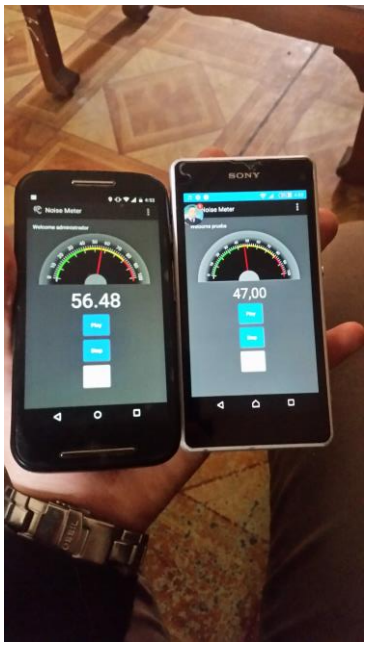

Figura 41. Prueba con Sony xperia

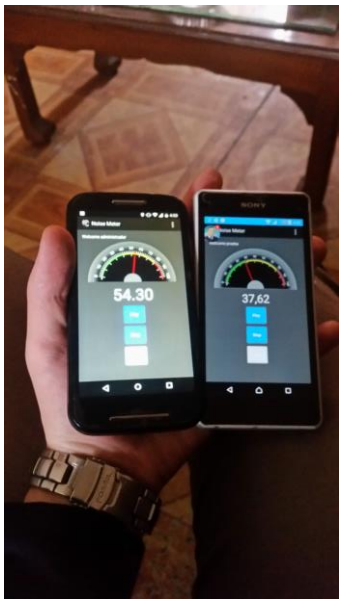

Figura 42. Prueba con Sony xperia

Tabla 25. Mediciones en Zona de Trabajo.

ZONA	RESULTADO APLICACIÓN (dBA)	RESULTADO SONÓMETRO (dBA)
Zona de trabajo	49.1	51.0
	46.36	48.2
	48.3	51.8
	49.51	52.5
	48.46	51.5
	49.86	52.8
	50.08	51.0
	48.43	51.5
	49.31	53.0
	49.31	51.3
Resultado	48.97950443	51.62718083
Margen de error	5.62%	

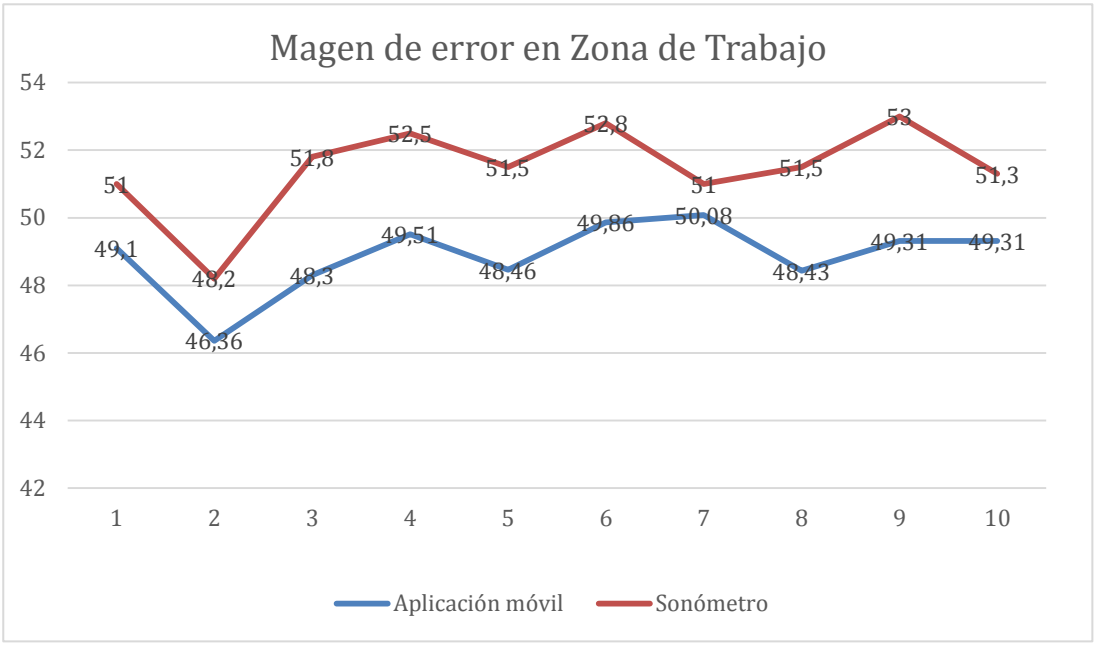

Figura 43. Margen de error de la aplicación móvil en zona de trabajo.

Tabla 26. Mediciones en Zona Residencial.

ZONA	RESULTADO APLICACIÓN (dBA)	RESULTADO SONÓMETRO (dBA)
Zona residencial	60.72	60.8
	61.05	58.2
	54.95	53.6
	54.1	53.3
	52.83	52.1
	53.91	54.5
	59.69	54.5
	53.33	54.5
	64.86	63.8
	63.61	64.0
Resultado	59.98515348	59.15096625
Margen de error	1.39%	

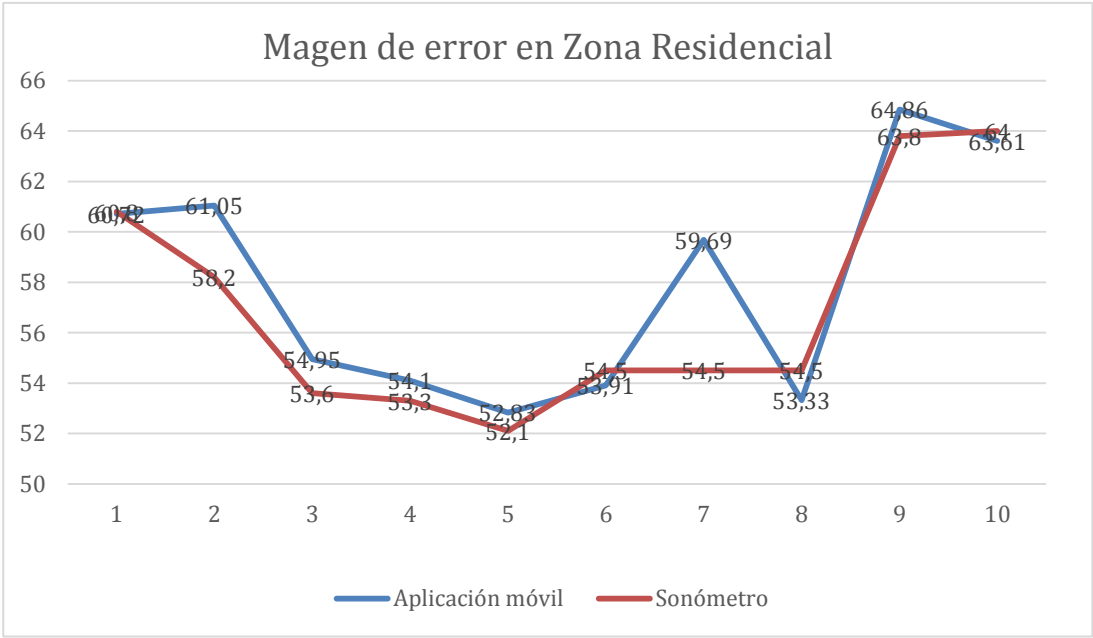

Figura 44. Margen de error de la aplicación móvil en zona residencial.

Tabla 27. Mediciones en Avenida.

ZONA	RESULTADO APLICACIÓN (dBA)	RESULTADO SONÓMETRO (dBA)
Avenida	67.15	69.3
	62.98	61.0
	63.2	62.8
	64.52	62.8
	64.54	67.9
	64.26	66.9
	68.16	67.1
	63.33	67.7
	67.74	70.3
	70.8	73.3
Resultado	66.46117494	68.30953078
Margen de error	2.73%	

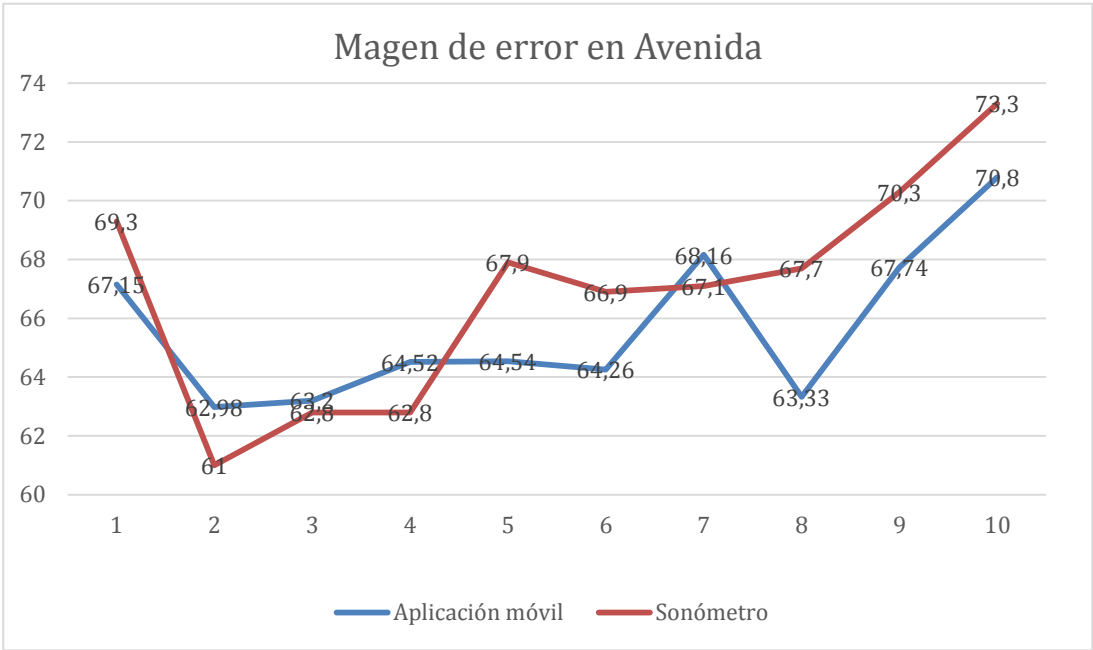

Figura *45.* Margen de error de la aplicación móvil en avenida.

Tabla 28. Mediciones en Zona Industrial.

ZONA	RESULTADO APLICACIÓN (dBA)	RESULTADO SONÓMETRO (dBA)
Zona industrial	89.02	91.0
	89.47	89.1
	88.84	89.4
	89.07	88.4
	89.44	86.3
	88.74	91.4
	89.38	90.7
Resultado	89.14577042	89.75524489
Margen de error	0.68%	

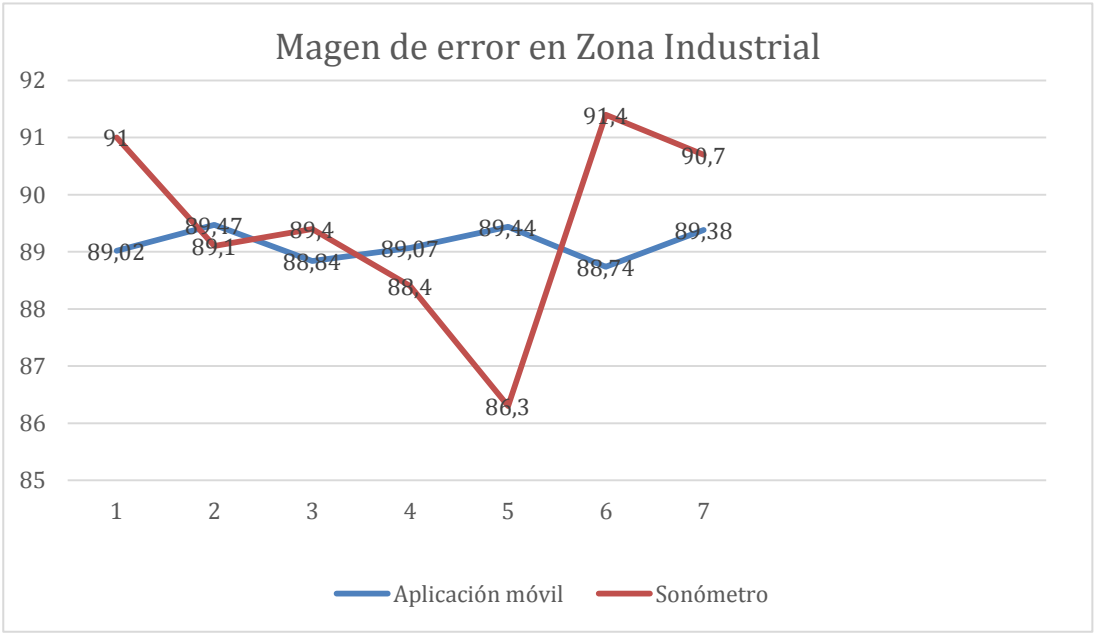

Figura 46. Margen de error de la aplicación móvil en zona industrial.

Tabla 29. Mediciones en Zona Industrial.

ZONA	RESULTADO APLICACIÓN (dBA)	RESULTADO SONÓMETRO (dBA)
Zona de Trabajo	48.97950443	51.62718083
Zona Residencial	59.98515348	59.15096625
Avenida	66.46117494	68.30953078
Zona industrial	89.14577042	89.75524489

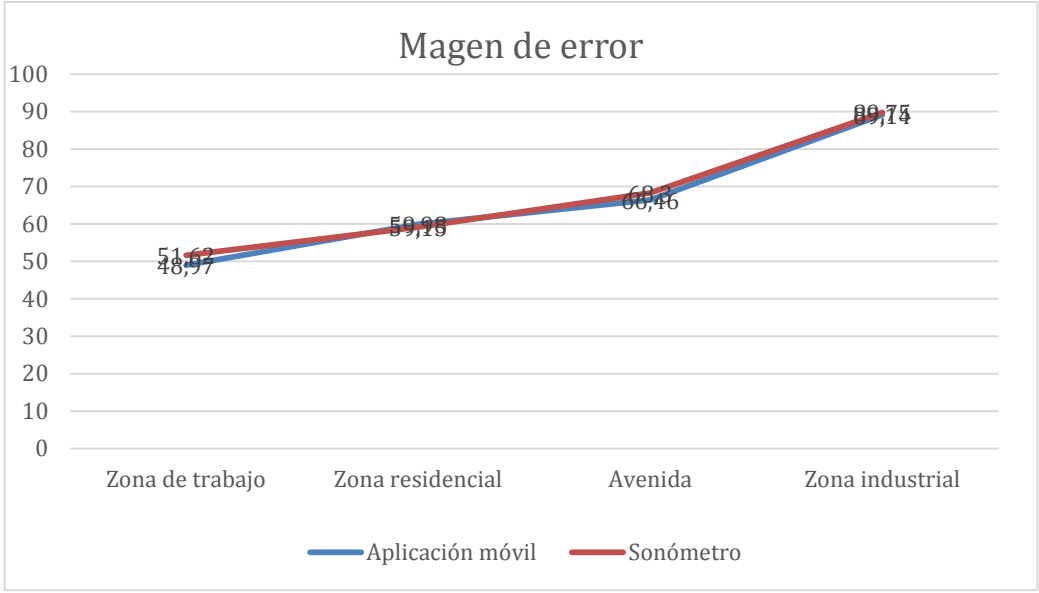

Figura 47. Margen de error total de la aplicación móvil.

En la siguiente tabla tenemos la prueba relacionada con el registro de las mediciones realizadas.

Tabla 30. Prueba de aceptación 4

REGISTRO DE MEDICIONES		
ID Historia de usuario	APP4	
SECUENCIA DE LA PRUEBA		
Procedimientos	**Resultados esperados**	**Resultado**
Detener el proceso de medición con el botón *stop* y presionar el botón *save*	Registrar la medición realizada en la base del móvil y la externa	OK

En la siguiente tabla tenemos la prueba de almacenamiento de las mediciones realizadas.

Tabla 31. Prueba de aceptación 5

MOSTRAR MEDICIONES EN EL MAPA DE CALOR MOVIL		
ID Historia de usuario	APP5	
SECUENCIA DE LA PRUEBA		
Procedimientos	**Resultados esperados**	**Resultado**
Seleccionar en el submenú la opción *Heatmap*	Mostrar el mapa de calor con los puntos guardados en la base del dispositivo móvil.	OK

En la siguiente tabla tenemos la prueba de inicio de sesión al sistema web.

Tabla 32. Prueba de aceptación 6

INICIO DE SESION AL SISTEMA WEB		
ID Historia de usuario	MDC1	
SECUENCIA DE LA PRUEBA		
Procedimientos	**Resultados esperados**	**Resultado**
Ingresar las credenciales del usuario creado en la aplicación móvil	Validación de credenciales e ingreso al sistema web	OK

En la siguiente tabla tenemos la prueba de consulta de mediciones por rango de fechas.

Tabla 33. Prueba de aceptación 7

CONSULTA DE MEDICIONES		
ID Historia de usuario	MDC2	
SECUENCIA DE LA PRUEBA		
Procedimientos	**Resultados esperados**	**Resultado**
Seleccionar el rango de fechas en las que se registraron las mediciones y buscar	Mostrar las mediciones en el rango de fecha establecido	OK

En la siguiente tabla tenemos la prueba de consulta de mediciones por rango de fechas.

Tabla 34. Prueba de aceptación 8

MOSTRAR MEDICIONES EN EL MAPA DE CALOR WEB		
ID Historia de usuario	MDC3	
SECUENCIA DE LA PRUEBA		
Procedimientos	**Resultados esperados**	**Resultado**
Presionar el botón *Go to map* en la interfaz de consulta de mediciones	Mostrar todas las mediciones realizadas	OK

4. Conclusiones y Recomendaciones

Conclusiones

Gracias al análisis realizado entre metodologías ágiles y tradicionales, se pudo escoger la metodología mas adecuada para el proyecto, siendo esta SCRUM, ya que tiene muchas ventajas a la hora de realizar cambios en la aplicación y no tener un gran impacto, en este caso al haber aumentado funcionalidades en el sistema web, como la consulta de mediciones y la interfaz de inicio de sesión. Otra de las ventajas de usar esta metodología fue la revisión al final de cada Sprint con el tutor y darla por satisfecha. Resultó de gran ayuda al establecer un proceso lógico y funcional de cómo manejar un proyecto de desarrollo de software.

En el proyecto fue muy importante saber seleccionar las herramientas, ya que estas facilitaron el desarrollo de las aplicaciones. Android Studio fue de gran ayuda ya que posee un patrón de diseño muy moderno y provee componentes específicos, facilitando el desarrollo para Android. Ruby on Rails fue otro de los mas importantes ya que al ser moderno y muy productivo para entornos web, facilitaron el desarrollo de los servicios web y la creación de la base de datos.

Al aplicar las pruebas de aceptación se pudo verificar que cada una de las historias de usuario fueron satisfechas.

Con las pruebas de medición realizadas, se concluyó que todos los celulares no van a marcar el nivel de ruido correcto. Esto ya que los dispositivos móviles no poseen un micrófono del mismo fabricante, y por lo tanto con diferentes características. En las pruebas, los celulares de marca Motorola son los que marcaron de manera casi precisa con el sonómetro en todos los niveles de ruido, mientras que con Samsung solamente se aproximaba con mediciones de valores bajos y muy altos.

La aplicación resulta más precisa en ambientes industrial y residencial según las pruebas realizadas y el margen de error calculado.

El proyecto en general es aplicable, siempre y cuando se utilicen dispositivos de marca Motorola o Samsung en caso de medir ruido bajos o altos. Además se debe tomar en cuenta que la base de datos permite un máximo de 10 000 registros. Es decir, el proyecto es aplicable si es para cierta entidad como una institución educativa.

Recomendaciones

Es recomendable seguir realizando pruebas con diferentes marcas de celulares e ir probando el margen de error que poseen en comparación con un sonómetro.

Se recomienda realizar la aplicación para otras plataformas como por ejemplo IOS y probar su margen de error.

Se recomienda hacer uso de herramientas de diseño gráfico para mejorar las interfaz de las aplicaciones.

Se recomienda estar pendiente de las nuevas herramientas, métodos y estándares de diseño que estipula Google cada cierto tiempo, con el fin de mantener un código actualizado y atractivo al usuario final.

REFERENCIAS

Android. (s.f). *Jelly Bean*, Recuperado el 10 de febrero de 2016 de : http://developer.android.com/about/versions/jelly-bean.html#android-41 http://wildfly.org/about/.

Aulet, J. (s.f). *Introducción a REST*. Recuperado el 17 de noviembre de 2015 de: http://www.sindikos.com/2011/05/breve-introduccion-a-rest-1/

DesarrolloWeb.(s.f). *Qué es la programación orientada a objetos*. Recuperado el 01 de marzo de 2016 de: http://www.desarrolloweb.com/articulos/499.php

Google (s.f). *Mapa de calor: Conceptos*. Recuperado el 10 de febrero de 2016 de: https://support.google.com/fusiontables/answer/1152262?hl=en.

Lindholm, T., Yellin, F. (1999). *The Java Virtual Machine specification*. (2ª. ed). Addison-Wesley.

Luján. M. (s.f). *Programación en Internet*. Recuperado el 10 de febrero de 2016 de: http://hdl.handle.net/10045/16994.

MetrosCubicos. (s.f). *Decibeles permitidos en zonas residenciales.* Recuperado el 17 de noviembre de 2015 de: http://www.metroscubicos.com/articulo/consejos/2013/12/03/modifican-limites-permisibles-de-emision-de-ruido .

Ministerio de Ambiente del Ecuador.(s.f). *Ecuador de le dice ¡NO AL RUIDO!*. Recuperado el 17 de noviembre de 2015 de: http://www.ambiente.gob.ec/hoy-ecuador-le-dice-no-al-ruido/

Ministerio de Ambiente del Ecuador.(2015). *LIMITES PERMISIBLES DE NIVELES DE RUIDO AMBIENTE PARA FUENTES FIJAS Y FUENTES*

MÓVILES, Y PARA VIBRACIONES. Quito, Ecuador: Ministerio del Ambiente.

Navarro, A., Fernández, J y Morales, J. (2013). Revisión de metodologías ágiles para el desarrollo de software. (1ª. ed.). Cali, Colombia: Universidad Icesi.

Matsumoto, Y. (2000). *The Ruby Programming Language: Concept*

Permalink. (s.f). *Metodologías de Desarrollo de Software*. Recuperado el 10 de febrero de 2016 de : http://latecladeescape.com/h/2015/07/metodologias-de-desarrollo-del-software.

Postgresql. (s.f). *Postgresq: Introdución*. Recuperado el 10 de febrero de 2016 de: http://www.postgresql.org/about/.

RFC4329 (s.f). *Javascript: Concepto*. Recuperado el 10 de febrero de 2016 de: http://web.archive.org/web/20140316122853/http://www.apps.ietf.org/rfc/rfc4329.html#sec-5.

Sqlite, (s.f). *SQLite: Concepto*. Recuperado el 10 de febrero de 2016 de: https://www.sqlite.org/features.html.

ProyectosAguiles. (s.f). *Qué es scrum.* Recuperado el 17 de noviembre de 2015 de : http://www.proyectosagiles.org/que-es-scrum .

ScrumGuides. (s.f). *Scrum.* Recuperado el 17 de noviembre de 2015 de*:* http://www.scrumguides.org/scrum-guide.html#artifacts-productbacklog .

ANEXOS

ANEXO 1: MANUAL DE USUARIO

Aplicación Móvil

1. Registro de usuario

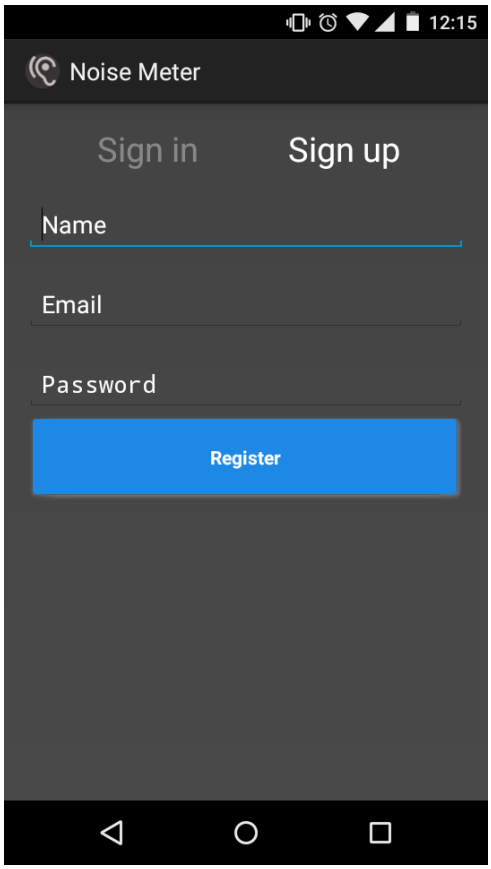

Figura 48. Interface de registro de usuario.

Para crear una cuenta, el usuario de llenar todos los campos de texto, una vez completados dichos campos debe presionar el botón *Register* para finalizar.

Si el registro es satisfactorio, la aplicación re direcciona hacia la pantalla de inicio de sesión para ingresar.

1.1 Mensaje de error

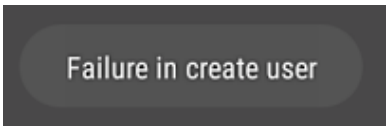

Figura 49. Falla de registro de usuario

Este mensaje es mostrado cuando el usuario no ingresó datos válidos o bien no los ingresó.

2. Inicio de Sesión

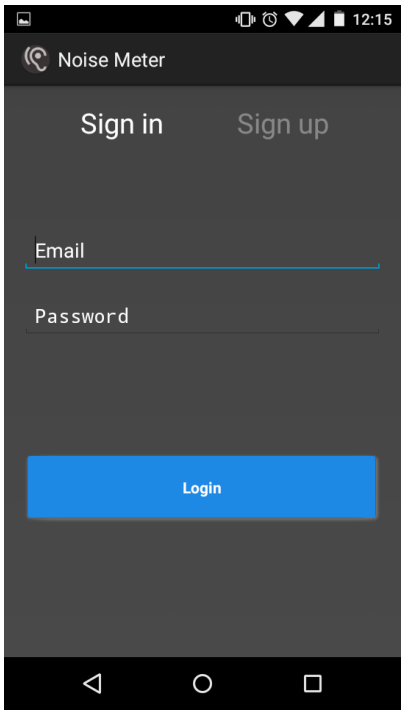

Figura 50. Interface de inicio de sesión.

Para ingresar a la aplicación el usuario necesita ingresar su correo electrónico, su contraseña y presionar el botón *Login.*

2.1 Mensajes de error

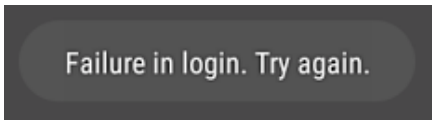

Figura 51. Falla de inicio de sesión de usuario

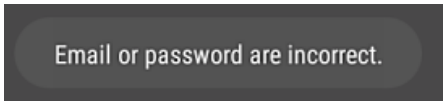

Figura 52. Falla de inicio de sesión de usuario

Estos mensajes son mostrados cuando los datos del usuario son incorrectos o no los ingresó.

3. Medición de ruido

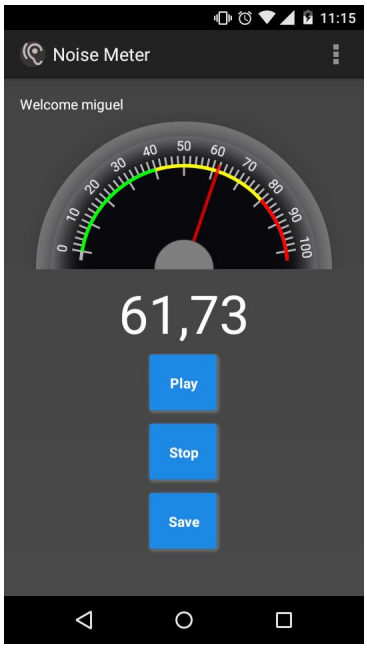

Figura 53. Interface de medición de ruido.

Se debe presionar el botón *Play* para comenzar a medir ruido, en la pantalla aparece el valor captado, luego presionar el botón *Stop* y seguidamente el botón *Save* para guardar la medición.

3.1 Mensajes satisfactorios

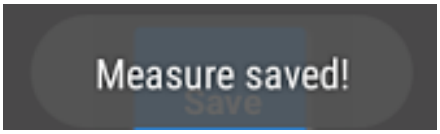

Figura 54. Medición registrada localmente

El mensaje se muestra cuando la medición se guardó en el dispositivo móvil.

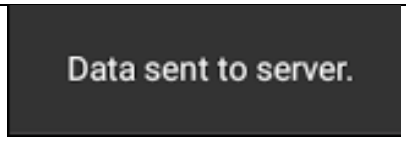

Figura 55. Medición registrada externamente

El mensaje se muestra cuando la medición se guardó en la base de datos externa.

3.2 Submenú

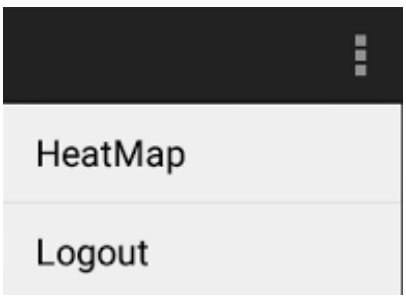

Figura 56. Submenú

En la esquina superior derecha está el submenú, el cual tiene las opciones para ir a la pantalla con el mapa de calor en el dispositivo móvil y una para cerrar sesión.

4. Mapa de calor móvil

Figura 57. Mapa de calor de aplicación móvil

El mapa de calor representa todas las mediciones guardadas en el dispositivo móvil.

Sistema Web

1. Inicio de Sesión

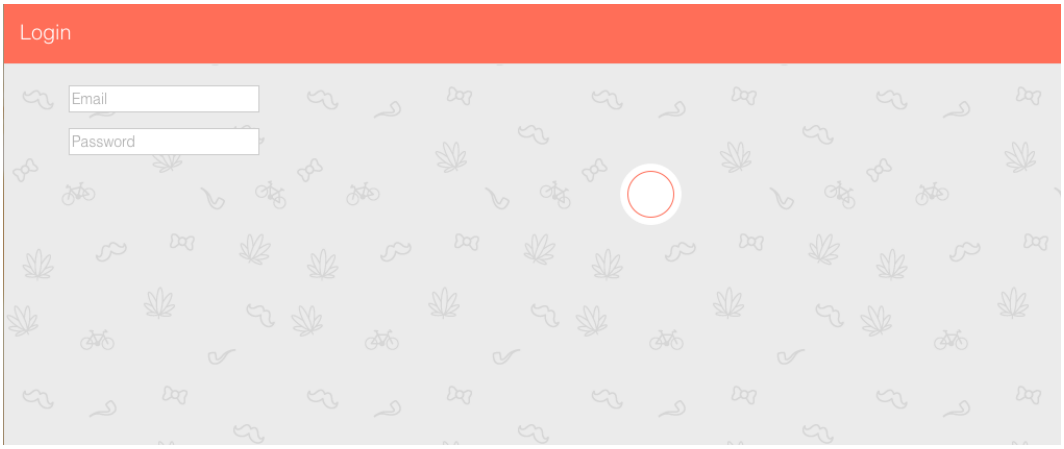

Figura 58. Interfaz de inicio de sesión del sistema web.

Para ingresar al sistema web el usuario necesita digitar los mismos datos de la aplicación móvil.

2. Consulta de mediciones

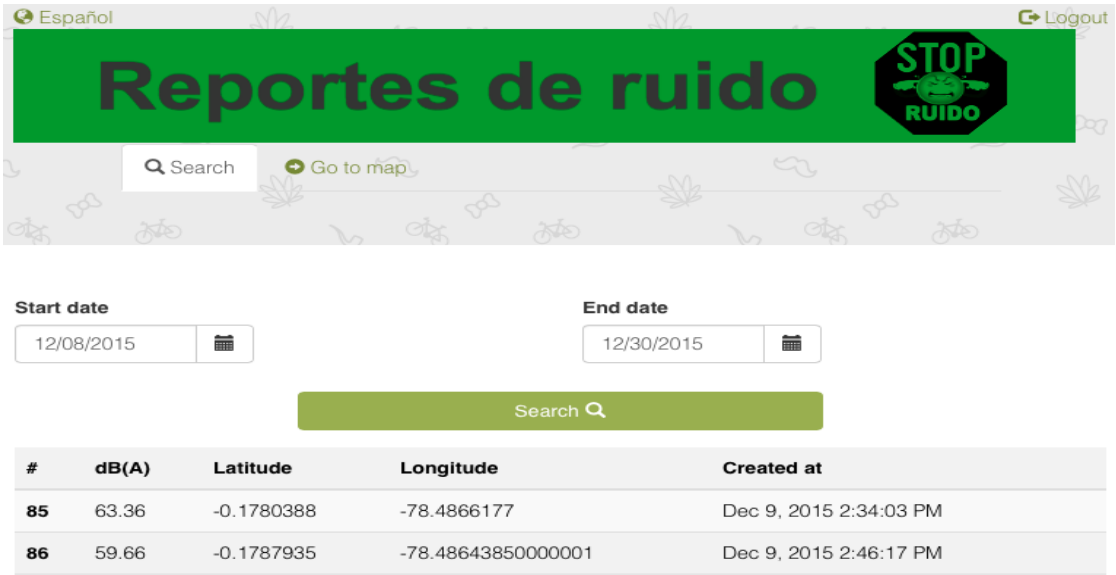

Figura 59. Interfaz de reporte de mediciones.

En esta interfaz el usuario selecciona un rango de fechas en las cuales fueron registradas las mediciones y presiona el botón *Search*. Se muestran en una tabla como se puede apreciar en la imagen.

Como se puede observar abajo del banner existen 2 pestañas, *Search* que es para la consulta de mediciones y *Go to map* que es para mostrar el mapa de calor web

3. Mapa de calor web

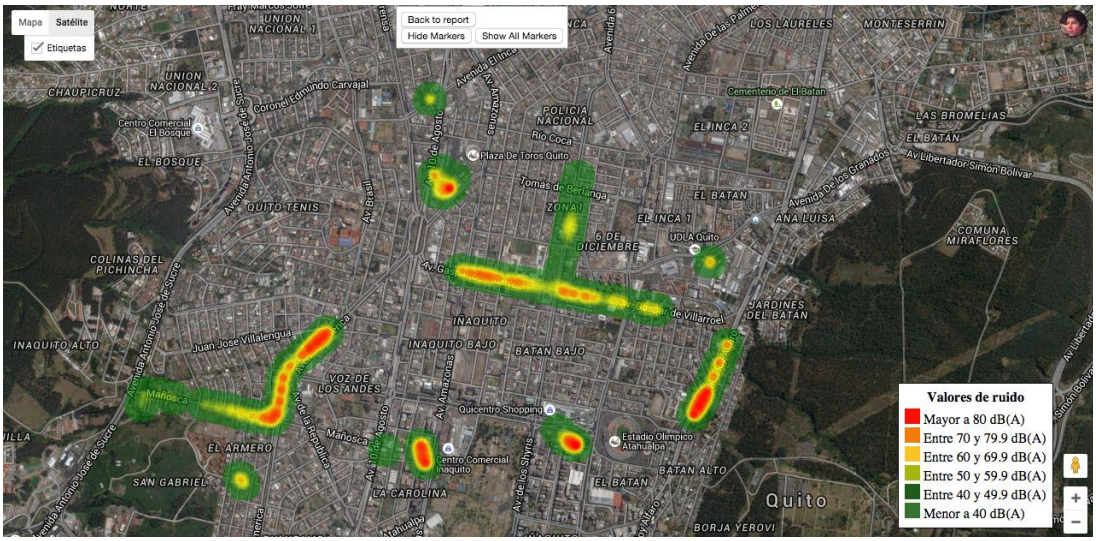

Figura 60. Mapa de calor de aplicación web

Adaptado de (googlemaps, s.f)

En este mapa se presentan todas las mediciones registradas. En la parte superior tiene las opciones *Show All Markers* para insertar marcadores en cada medición y mostrar su valor. La opción *Hide markers* sirve para ocular los marcadores.

ANEXO 2: CODIFICACIÓN

1. Registro de Usuario

```
register.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        try {
            new HttpAsyncTask().execute("https://"+url+"/user");
            signIn.setTextColor(Color.WHITE);
            signUp.setTextColor(Color.GRAY);
            name.setVisibility(View.INVISIBLE);
            register.setVisibility(View.INVISIBLE);
            btnIngresar.setVisibility(View.VISIBLE);
        } catch (Exception e) {
            e.printStackTrace();

        }
    }
});

private class HttpAsyncTask extends AsyncTask<String, Void, String> {

    @Override
    protected String doInBackground(String... urls) {

        User user = new User();

        user.setName(name.getText().toString());
        user.setEmail(email.getText().toString());
        user.setPassword(password.getText().toString());
        return POST(urls[0],user);
    }
    // onPostExecute displays the results of the AsyncTask.
    @Override
    protected void onPostExecute(String result) {
        if(result.equals("201")){
            Toast.makeText(getBaseContext(), "Create success", Toast.LENGTH_LONG).show();
        }else {
            Toast.makeText(getBaseContext(), "Failure in create user", Toast.LENGTH_LONG).show();
            progress.setVisibility(View.INVISIBLE);
        }

    }
}
```

Figura 61. Método de la aplicación que permite registrar un usuario.

Este método realiza los siguientes procedimientos:

1. Al presionar el botón de registro se realiza una llamada a una tarea asíncrona, la cual se encarga de llamar al servicio REST.
2. Se envían los datos ingresados por el usuario mediante el método POST.

3. El método *onPostExecute* se encarga de recibir la respuesta de la creación.

1.1 Método para crear usuario

```
def create
    user = User.new(user_params)
    if user.save
        render json: user, status: 201, location: user
    else
      render nothing: true, status: 404
    end
  end


private
    def user_params
        params.require(:user).permit(:id, :name, :mail, :password)
    end
```

Figura 62. Método REST para crear una cuenta de usuario.

Este método REST permite crear una nueva cuenta de usuario con los respectivos parámetros.

1.2 Clase User en Ruby

```
class User < ActiveRecord::Base
    has_many :measure
    validates :name, :mail, :password, presence: true
    validates :mail, uniqueness: true
    validates :mail, :email_format => {:message => 'Invalid email_format'}
end
```

Figura 63. Clase User en Ruby para validaciones.

Esta clase *User* permite validar que la cuenta posea todos los campos, el email sea único y tenga el formato correcto.

2. Inicio de Sesión en dispositivo móvil

```
userGet = new UserGet(new AsyncResponseUser() {
    @Override
    public void getUserRest(String response) {

        statusCode = response;

        try {
            btnIngresar.setEnabled(false);
            JSONObject jsonObject = new JSONObject(statusCode);
            id = jsonObject.get("id").toString();
            userName = jsonObject.get("name").toString();
            sqliteEmail = jsonObject.getString("mail").toString();
            sqlitePassword = jsonObject.getString("password").toString();
            progress.setVisibility(View.VISIBLE);
        } catch (JSONException e) {
            e.printStackTrace();
            Toast.makeText(getBaseContext(), "Failure in login. Try again.", Toast.LENGTH_SHORT).show()
            btnIngresar.setEnabled(true);
            progress.setEnabled(false);
        }

    }
});
userGet.execute("https://" + url + "/user" + "?mail=" + email.getText().toString() +
                    "&password=" + password.getText().toString());
```

Figura 64. Método de la aplicación para autenticar al usuario.

Este método es responsable de llamar al servicio REST mediante la tarea asíncrona *UserGet* para obtener un usuario. Define la URL, estructurada de la siguiente manera:

- <nombre_página>/user: Ruta base para obtener usuarios.
- ?fields=mail,password: Se aplican filtros para obtener usuario deseado.

En caso de haber encontrado un usuario, retornará un objeto con dicho usuario, de lo contrario un código de error y mostrando al usuario el mensaje *Failure in login. Try again*.

2.1 Clase asíncrona

```
public class UserGet extends AsyncTask<String, String, String> {

    public AsyncResponseUser asyncResponseGet=null;

    public UserGet(AsyncResponseUser asyncResponseGet){
        this.asyncResponseGet = asyncResponseGet;
    }

    @Override
    protected String doInBackground(String... strings) {
        String urlString = strings[0]; // URL to call
        String result = "";
        int statusCode = 0;

        try {
            URL url = new URL(urlString);
            HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
            statusCode = urlConnection.getResponseCode();
            InputStream inputStream = urlConnection.getInputStream();
            urlConnection.setConnectTimeout(5000);
            urlConnection.setReadTimeout(5000);

            if (statusCode != 200){
                throw new Exception();
            }
            if (null != inputStream)
                result = IOUtils.toString(inputStream);
        } catch (SocketTimeoutException s) {
            return "Server is down!";
        } catch (Exception e) {
            System.out.println("Status----------------> " + statusCode);
            System.out.println(e.getMessage());
            return String.valueOf(statusCode);
        }
        return result;
    }

    @Override
    protected void onPostExecute(String result) {
        asyncResponseGet.getUserRest(result);
        Log.i("FromOnPostExecute", result);
        System.out.println(result);
    }
```

Figura 65. Tarea asíncrona para conexión con Servicio REST.

Esta clase es llamada desde el inicio de sesión para establecer la conexión con el servicio REST de usuario y obtener los datos. Realiza el siguiente procedimiento:

- Obtiene la URL a llamar en el método *doInBackground.*
- Se abre una conexión y retorna el código de respuesta con los datos en caso de que existan.
- Verifica si el código de respuesta es diferente a 200 retorna el código de error, caso contrario los datos de usuario.

2.2 Registro local de usuario

```
Cursor cursor;
cursor = sqLiteDatabase.query("usuario", new String[]{"_id"}, "_id=?",
        new String[]{id.toString()}, null, null, null);

if (cursor.moveToFirst()) {
    do {

        try {
            ContentValues contentValues = new ContentValues();
            contentValues.put("session", true);
            sqLiteDatabase.update("usuario", contentValues, "_id=" + Integer.valueOf(id), null);
            Log.i("Login", "User updated to true");
            break;
        } catch (NumberFormatException e) {
            e.printStackTrace();
            Log.e("Login", "User updated to true FAILED");
        }

    }
    while (cursor.moveToNext());

} else {
    try {
        ContentValues contentValues = new ContentValues();
        contentValues.put("_id", Integer.valueOf(id));
        contentValues.put("nombre", userName);
        contentValues.put("email", sqliteEmail);
        contentValues.put("contrasena", sqlitePassword);
        contentValues.put("session", true);
        sqLiteDatabase.insert("usuario", null, contentValues);
        Log.i("Login", "User inserted");
    } catch (NumberFormatException e) {
        e.printStackTrace();
        Log.e("Login", "User inserted FAILED");
    }
}
cursor.close();
```

Figura 66. Método de la aplicación que permite registrar al usuario localmente.

El método realiza los siguientes procedimientos:

1. Consulta en la base del móvil si existe un usuario con el id especificado.
2. Si encuentra un resultado, se actualizará el campo *session* a TRUE.
3. Si no encuentra ningún resultado, se registrará el usuario con todos sus datos.

2.3 Inicio de sesión automático

```
try {
    cursor = sqLiteDatabase.query("usuario", new String[]{"session", "_id", "nombre", "email"},
            null, null, null, null, null);

    if (cursor.moveToFirst()) {
        do {
            boolean sessionBool = cursor.getInt(0) > 0;
            if (sessionBool) {

                Intent intent = new Intent(LoginActivity.this, MeasureActivity.class);
                intent.putExtra("id", cursor.getInt(1));
                intent.putExtra("name", cursor.getString(2));
                intent.putExtra("mail", cursor.getString(3));
                startActivity(intent);
                finish();
                System.out.println("Encontrado BOOLEAN " + cursor.getInt(1));
                break;
            }

        }
        while (cursor.moveToNext());
    }
    cursor.close();
} catch (Exception e) {
    e.printStackTrace();
}
```

Figura 67. Método de la aplicación que permite realizar autenticación automática.

Si un usuario se autenticó anteriormente, no es necesario que lo realice nuevamente. El método realiza el siguiente procedimiento:

1. Consulta todos los usuarios de la base local.
2. Si existe un usuario con el campo *session* en TRUE, realiza una autenticación de ese usuario.

```ruby
def index

    users = User.find_by(mail: params[:mail], password: params[:password])
    if users
      render json: users, status: 200
    else
      render nothing: true, status: 404
    end

end
```

Figura 68. Método REST para buscar un usuario.

Este método REST permite realizar la búsqueda del usuario a autenticar.

3. Medición de Ruido

```java
public double getAmplitude() {
    if (mRecorder != null)
        return (mRecorder.getMaxAmplitude());
    else
        return 0;
}

public void start() throws IOException {
    if (mRecorder == null) {
        handler.removeCallbacks(runnable);
        mRecorder = new MediaRecorder();
        mRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
        mRecorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
        mRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
        mRecorder.setOutputFile("/dev/null");
        mRecorder.prepare();
        mRecorder.start();
    }
}

runnable = new Runnable() {
    @Override
    public void run() {
        powerDb = 0;
        tex.setText(String.valueOf(powerDb));
        powerDb = (20 * Math.log10(getAmplitude()));
        speedometer.setSpeed(powerDb);
        tex.setText(String.valueOf(String.format("%.2f",powerDb)));

        handler.postDelayed(this, 500);
    }
};
```

Figura 69. Método para usar el micrófono y medir el ruido.

Estos métodos realizan los siguientes procedimientos:

1. El método *start* inicializa el micrófono del dispositivo móvil.
2. El método *getAmplitude* permite obtener todo el ruido captado desde el micrófono del dispositivo móvil.
3. Finalmente, se usa la fórmula para obtener el ruido y se muestra en un widget.

3.1 Registro de Medición

```
btnSave.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        if(latitude != 0 && longitude != 0){

            new HttpAsyncTask().execute("https://"+url+"/measure");

            ContentValues contentValues = new ContentValues();
            contentValues.put("valor_db", powerDb);
            contentValues.put("latitud", latitude);
            contentValues.put("longitud", longitude);
            contentValues.put("hora", String.valueOf(new Date()));
            contentValues.put("usuario_id", idUser);
            sqLiteDatabase.insert("medicion", null, contentValues);
            Toast.makeText(MeasureActivity.this, "Measure saved!", Toast.LENGTH_LONG).show();
        }else {
            SnackBar snackBar = new SnackBar(MeasureActivity.this, "GPS is disabled!");
            snackBar.show();
        }


    }
});
```

Figura 70. Método de la aplicación para registrar una medición localmente.

Este método se encarga de registrar la medición realizada por el usuario en la base del dispositivo móvil. Después realiza la llamada de una tarea asíncrona para registrar la medición mediante el servicio REST.

```
private class HttpAsyncTask extends AsyncTask<String, Void, String> {
    @Override
    protected String doInBackground(String... urls) {
        Medicion medicion = new Medicion();

        medicion.setValor_db(powerDb);
        medicion.setLongitud(longitude);
        medicion.setLatitud(latitude);
        medicion.setUsuario_id(idUser);

        return POST(urls[0],medicion);
    }

    @Override
    protected void onPostExecute(String result) {
        if (result.equals("201")){
            SnackBar snackBar = new SnackBar(MeasureActivity.this, "Data sent to server.");
            snackBar.show();
        } else {
            SnackBar snackBar = new SnackBar(MeasureActivity.this, "Error sending data to server.");
            snackBar.show();
        }
    }
}
```

Figura 71. Tarea asíncrona para envío de datos al servicio REST.

Esta clase es llamada desde el botón de registro de medición para establecer la conexión con el servicio REST y enviar los parámetros. Realiza el siguiente procedimiento:

- Se llena el objeto con todos los parámetros de la medición y retorna el mismo.
- Después del registro, verifica el código de respuesta y muestra los respectivos mensajes al usuario.

```
def create
    measure = Measure.new(measure_params)
    if measure.save
        render json: measure, status: 201, location: measure
    else
      render nothing: true, status: 500
    end
end

private
    def measure_params
        params.require(:measure).permit(:id, :value, :latitude, :longitude, :user_id)
    end
```

Figura 72. Método REST para registro de medición.

Este método del servicio REST permite registrar la medición con todos los parámetros y retorna los códigos de respuesta.

4. Mapa de Calor en el Móvil

```
private void setUpMap() {

        ArrayList<WeightedLatLng> list = new ArrayList<>();
        LatLng latLng;
        WeightedLatLng weightedLatLng;
        final SQLHelper sqlHelper = new SQLHelper(this);
        sqLiteDatabase = sqlHelper.getWritableDatabase();
        Cursor cursor;

        // Create the gradient.
        int[] colors = {
                Color.rgb(102, 225, 0), // green
                Color.rgb(255, 0, 0)    // red
        };

        float[] startPoints = {
                0.2f, 1f
        };

        Gradient gradient = new Gradient(colors, startPoints);

        cursor = sqLiteDatabase.query("medicion", new String[]{"valor_db", "latitud",
            "longitud", "hora"}, null, null, null, null, null);

        if (cursor.moveToFirst()) {
            do {
                latLng = new LatLng(cursor.getDouble(1), cursor.getDouble(2));
                weightedLatLng = new WeightedLatLng(latLng, cursor.getDouble(0));
                list.add(weightedLatLng);
                mMap.addMarker(new MarkerOptions().position(latLng).
                    title(String.valueOf(String.format("%.2f", cursor.getDouble(0))) + " dB"));
            }
            while (cursor.moveToNext());
        }
        cursor.close();

        heatMap = new HeatmapTileProvider.Builder().weightedData(list).gradient(gradient).build();
        TileOverlay mOveray = mMap.addTileOverlay(new TileOverlayOptions().tileProvider(heatMap));
    }
```

Figura 73. Método para presentar el mapa de calor en móvil y datos de mediciones.

Este método es responsable de iniciar el modulo del mapa, obtener y presentar el valor de cada medición guardada localmente usando los puntos de ubicación.

5. Inicio de sesión en sistema web

```
$scope.loginFunction = function(){
    //console.log($scope.mail);
    $scope.requestsSearch = requestFactory.login({
        mail : $scope.mail,
        password : $scope.password
    });

    $timeout(function(){
        if($scope.requestsSearch.name !== undefined){
            $rootScope.user = 1;
        }
        console.log($scope.requestsSearch.name);

        if($rootScope.user == 0){
            alert("Error. Intente de nuevo!");
        } else{
            $state.go('request.recent.search');
        }


    }, 1800);


}
```

Figura 74. Método donde recibe los datos del usuario para iniciar sesión.

Este método es responsable de enviar el usuario y contraseña de la persona, y además recibe la respuesta con la cual determina si los datos son correctos para poder iniciar sesión.

6. Reporte de mediciones

```
$scope.requestsSearchFunction = function(){
    var incMax = new Date($scope.dtMax);
    incMax.setDate($scope.dtMax.getDate() + 1);
    var incMin = new Date($scope.dtMin);
    incMin.setDate($scope.dtMin.getDate());

    $scope.requestsSearch = requestFactory.byAdvancedSearch({
        startDate : $filter('date')(incMin, 'yyyy/M/d'),//'M/d/yyyy'
        endDate : $filter('date')(incMax, 'yyyy/M/d'),
        });


    $timeout(function(){
        console.log($scope.requestsSearch);
    }, 800);
}
```

Figura 75. Método donde llama al web service para consulta de datos.

Este método es responsable de obtener el rango de fechas, llamar al servicio web para la búsqueda y obtener la respuesta para mostrarla en pantalla.

6.1 Método de consulta de mediciones

```
def index
  if params[:startDate]
    measures = Measure.joins(:user).where("measures.created_at >= :startDate AND
      measures.created_at <= :endDate",
    {startDate: params[:startDate], endDate: params[:endDate]})
  else
     measures = Measure.all
  end
  render json: measures, status: 200
end
```

Figura 76. Servicio web restful para realizar consulta de mediciones.

Este servicio web es responsable de obtener todas las mediciones realizadas en un rango de fechas, o bien todas las mediciones.

7. Mapa de Calor Web

```
var myLatlng = new google.maps.LatLng(25.6586, -80.3568);
    var myOptions = {
      zoom: 14,
      maxZoom: 16,
      minZoom: 10,
      center: new google.maps.LatLng(-0.180653, -78.467834),
      mapTypeId: google.maps.MapTypeId.HYBRID
    };
    map = new google.maps.Map(document.getElementById("map-canvas"), myOptions);
    // heatmap layer
    heatmap = new HeatmapOverlay(map,
      {
        "radius": 0.0006,
        "maxOpacity": 1,
        "minOpacity":0.5,
        "scaleRadius": true,
        "useLocalExtrema": true,
        latField: 'lat',
        lngField: 'lng',
        valueField: 'count',
        gradient: {
              '.5': 'green',
              '.8': 'yellow',
              '1': 'red'
            }
      }
    );
      var testData2 = {
          max: 500,
          data: pointData
        };
    heatmap.setData(testData2);
    }
```

Figura 77. Método para presentar el mapa de calor web y datos de mediciones.

Archivo JavaScript donde se carga y configura el mapa de calor web.