Universidad Autónoma Metropolitana
Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería
Licenciatura en Ingeniería en Computación

Aplicación móvil para medición de ruido ambiental

Proyecto Tecnológico

Presenta:
Martínez Ramírez Giovanni 210200557

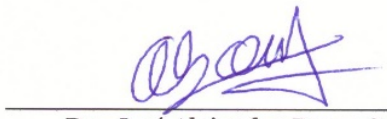
Asesor:
Dr. José Alejandro Reyes Ortiz
Depto. Sistemas, UAM-Azcapotzalco

Co-Asesor:
Dr. Rafael Alberto Méndez Sánchez
ICF-UNAM Cuernavaca, Morelos

Trimestre 2017 Primavera
Julio 2017

Declaratoria

Yo, José Alejandro Reyes Ortiz, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de la UAM Azcapotzalco.



Dr. José Alejandro Reyes Ortiz

Yo, Rafael Alberto Méndez Sánchez, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de la UAM Azcapotzalco.

Dr. Rafael Alberto Méndez Sánchez

Yo, Giovanni Martínez Ramírez, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de la UAM Azcapotzalco.

Martínez Ramírez Giovanni

Agradecimientos

A MI CREADOR Y A MI MAESTRO JESÚS: Por darme la convicción y la fortaleza de seguir con cada alba, además de guiarme por el camino correcto y brindarme infinita ayuda, pero, sobre todo, amor.

A MIS PADRES: No tengo como agradecer su amor y su apoyo todos estos años, sólo me queda compartir con ustedes y los demás, la persona que forjaron y cuidaron, y que con mucho cariño les integra en uno más de muchos logros, recuerden que los amo y que siempre están en mi corazón.

A MI HERMANO: Por darme la oportunidad de crecer contigo y disfrutar buenos y malos momentos, que siempre elijas lo correcto y que jamás desistas de tus sueños y con mucho cariño, te doy las gracias.

A MI FAMILIA: Les doy las gracias a cada uno de ustedes, por haber compartido conmigo una comida, una sonrisa, un viaje, una fiesta y demás, gracias por su cariño y su infinita ayuda.

A MI OTRA FAMILIA:

ANGIE: Por ser más que mi mejor amiga, mi hermana, por brindarme tú luz cuando lo necesitaba y cuando lo necesito, por quererme tal cual soy, por comprenderme en lo que la mayoría no, confiar y creer en mí, por guiarme y recordarme las cosas que conllevan a estar viviendo, por el apoyo incondicional que sin dudar me das, por llegar y compartir conmigo esta vida, te quiero mucho.

CHISPITA: Por abrir el corazón y dejarme entrar, por mostrarme un ejemplo de vida con mucha lucha y perseverancia, por ayudarme y escucharme, por brindarme cariño de tu maravilloso corazón, por compartir y estar conmigo, por eso y más yo soy el que da las gracias.

KARENCITA: Por llegar a mi vida y pasar momentos maravillosos, por escucharme y darme tu cariño, por apoyarme cuando lo necesito, pero sobre todo porque eres cómplice de esta amistad, gracias infinitas.

A MIS AMIGOS: Gerardo, Aarón, Lalo, Miguel, Guadalupe, Mirna, Elizabeth, Guadalupe Mejía, Ysys, Sergio Tamayo, Ayelen, Tania, Hadita, Pau, Andrea, Sophy, Conny, Adrianita, Claudia, Xim y a una amiga muy especial: ARIAT. A mis amigos de Unitec: Miguel, Raúl. A mis amigos de la UAM: Migue, Ive, Adrián Morán, Adrián Alvarado, Javier, Oskar, Irene, Ezra, Nestor, Eduardo, Demart, Areli, Ismael, Fanny, Silvia, Luis Pozos, Jess, Jessica Islas, Lizbeth Monroy, Jessica Galeana, Ivonne, Gice, Lucy, Abril, Ariana y a aquellos que me falto mencionar. A cada uno les doy las gracias por compartir conmigo los mejores momentos, y aunque lejos, perdidos, siempre tienen un lugar en mi corazón, los quiero mucho, y agradezco el haberlos conocido.

A mis amigos del C.E.C.I.Q: Jannette, Juan Venegas, Juan Escobar, Alejandro, Boreal, Pony, Dani, Lalo, Laura, Lalito, Sandy, Gaby, Tania, Rodrigo, Mario, Monse y finalmente a Martha que me invitó a formar parte de este lugar tan padrísimo y maravilloso, en el cual compartí momentos inolvidables y agradezco por llegar a mi vida, los quiero chicos.

A FERNANDO GASCA RUIZ: Por ser el ángel que Dios mando en mi camino para culminar, por apoyarme incondicionalmente cuando más lo necesitaba y por brindarme tu amistad sin conocernos, GRACIAS.

A MI PROFESORES: Por ser guías en mi aprendizaje y mi formación, a los Doctores Rafael Méndez, Gabriela Báez y Alejandro Reyes por darme la oportunidad de trabajar y culminar esta parte de mi desarrollo, finalmente a la doctora Deyanira Ángeles: por brindarme el apoyo cuando realmente no sabía a donde ir

Resumen

A nivel global las personas empiezan a tener enfermedades cuyos síntomas van desde el comportamiento negativo, la baja productividad hasta la pérdida de la capacidad auditiva, pero ¿que lo genera?, algo de lo cual estamos rodeados día con día: altos niveles de contaminación de ruido o también denominado contaminación acústica.

El ruido que se genera a nuestro alrededor se puede interpretar en señales y puede ser medido a través de unidades denominadas Hz y kHz, pero dichas mediciones deben ser transformadas en una escala llamada decibelios (dB) para que pueda ser comparativo con lo que se escucha, así se establece la escala de audición, que va desde los 0 dB hasta los 200 dB.

En el Reino Unido y Alemania se ha empezado a recopilar información en un mapa de ruido y se pretende usar tecnología a través de la red inalámbrica, para actualizarla e implementar la toma de datos con el celular y así reducir el impacto del ruido.

Para este proyecto de integración no se pretende realizar un mapa de ruido, pero si detectarlo con el desarrollo de una aplicación para los dispositivos móviles, implementando el sistema operativo Android y permitiendo medir el ruido ambiental, para que sea posible prevenir las altas señales del mismo, que pueden causar los problemas auditivos mencionados anteriormente.

El proyecto pretende dejar este desarrollo tecnológico en el Laboratorio de Sistemas Dinámicos y Prototipos (LSDP) dentro de la UAM-Azcapotzalco, y así en cada uno de los apartados introducirlos en un ámbito casi nuevo, donde puntualizaremos lo antes establecido de este desarrollo tecnológico.

Índice general

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

El ruido, también llamado contaminación acústica, se entiende como todo sonido no deseado, molesto e incluso dañino para la salud. Es un factor importante que interfiere con la comunicación a través de señales de audio en el ambiente y de los aparatos de comunicación (telecomunicaciones y radio). Así, las señales auditivas pueden perder claridad o perderse si su intensidad es menor que la del ruido.

Las señales que pueden ser escuchadas por el ser humano se encuentran en el llamado rango audible que va desde los 20 Hz hasta los 20 kHz. Para medir estas señales se utiliza una escala logarítmica de presiones llamada decibeles o decibelios (dB). En el apéndice A, se da una definición precisa de esta unidad, que va desde los 0dB en el umbral de audición, pasando por ejemplo por los 70 dB de una aspiradora, los 130 dB de un avión en despegue hasta los 200 dB generados por una bomba atómica.

La exposición permanente o continua al ruido puede provocar daños irreversibles en el oído humano, tales como la pérdida parcial o total de la capacidad auditiva (sordera), cuando la intensidad llega cerca de los 100 dB **[1].** Más aún, estos daños en la salud, también incluyen estrés, alteraciones del sueño, depresión, problemas musculares, agresividad y disminución de la atención **[2]**. Estos daños usualmente se manifestaban en personas de alrededor de 60 años, pero actualmente la población joven (menor a 30 años) también se está viendo afectada, aún sin darse cuenta. Por todo lo anterior y como reglas elementales de convivencia, ya existe una reglamentación para la generación de sonido en zonas habitacionales, conforme a la *Norma Ambiental para el Distrito Federal, NADF-005-AMBT-2013***[3].** Dicha norma establece los límites máximos permitidos a lo largo del día, los cuales se muestran en la Tabla 1.

Horario	Límite máximo permisible
6:00 h. a 20:00 h.	63 dB (A)
20:00 h a 6:00 h.	60 dB (A)

Tabla 1: Límites de ruido permitidos en decibeles **[3]**.

1.1 Sonómetros

Un sonómetro es un instrumento que sirve para medir ruido en la escala de decibeles. Existen sonómetros para mediciones generales y los de alta precisión, cada uno de ellos con diversas funciones tales como la medición manual o automática, con programación previa del tiempo de registro y almacenamientos de datos, entre otras.

Los componentes básicos de un sonómetro son un micrófono, un circuito que procesa la señal detectada y una unidad de lectura (pantalla o leds). Algunos cuentan además con una salida para conexión a un osciloscopio y/o computadora para la visualización de la señal. Actualmente, todos los sonómetros están regidos por la norma IEC 61.672**[4]** y se espera que la señal que registra cada uno de ellos sea similar para un mismo sonido. La calidad de los sonómetros que existen en el mercado y sus diversas funciones determinan su precio, que va desde los $2,500.00 hasta los $25,000.00 aproximadamente. Si bien, casi todos los teléfonos celulares cuentan con sensores incluídos, este tipo de sensor de ruido no está presente por el grado de dificultad, el cual, de estar presente, elevaría el costo del celular.

Dada la importancia de regular el ruido ambiental, en este proyecto proponemos el desarrollo de una aplicación de bajo costo, en una plataforma Android, para uso en un dispositivo móvil (teléfono celular). Esta permitirá la detección y registro de señales auditivas en un intervalo de tiempo, para su posterior análisis, tanto en la ciudad como en zonas remotas. Es importante mencionar que ya existen aplicaciones similares, por lo que la intención principal de este proyecto no es la originalidad, sino la asimilación, desarrollo y generación de tecnología propia dentro de la UAM-Azcapotzalco como parte de los objetivos perseguidos por el Laboratorio de Sistemas Dinámicos y Prototipos (LSDP) de la UAM Azcapotzalco, donde surge este proyecto.

Capítulo 2

Antecedentes

Las aplicaciones para móvil en el análisis de ruido ambiental, son relativamente nuevas y hay mucho por desarrollar para su uso tanto en la industria como en la investigación. A continuación, resumimos algunos trabajos anteriores a este proyecto relacionados con la medición de ruido usando *apps*.

2.1 Proyectos terminales o de integración:

- **Hernández Nieves, María Sara. "Clasificación de llantos de bebé con wavelets". [8]**

  En este trabajo, la autora identifica 5 tipos de llantos de bebe a partir del análisis del sonido usando el formalismo de la Transformada Wavelet Discreta (TWD). El registro del llanto se realizó a través de una aplicación simple desarrollada para dispositivo móvil y con la cual guardó los sonidos en un archivo con extensión .wav, para su posterior análisis por computadora.

  La semejanza con nuestro proyecto es el desarrollo de una aplicación para móvil, Sin embrago nosotros proponemos el registro de ruido ambiental en general. Por otro lado no se consultará a través de un archivo con extensión .wav, si no por un módulo de base de datos para consulta.

- **Espinosa Sánchez, Ana Rosa. "Simulación de la solución al problema directo de reflectrometría acústica". [9]**

  En esta investigación, se desarrolla el software en plataforma MATLAB 7.0.12. para simular la distorsión que sufre una señal acústica que es dispersada por una cavidad cilíndrica de dimensiones particulares. Este análisis es conocido en la literatura como el problema inverso de la reflectrometría acústica.

  La semejanza es que analiza señales acústicas distorsionadas, lo cual es un tipo de ruido.

2.2 Tesis:

- **López Fernández, Margarita Beatriz, "Utilización de sensores de bajo costo para el desarrollo de prácticas de laboratorio en tecnología: Análisis y evaluación del ruido mediante el uso de aplicaciones móviles". [10]**

  La autora muestra la utilidad de las aplicaciones ya existentes para la detección de ruido en dispositivos móviles con fines docentes. En esta investigación se señalan algunos problemas de los dispositivos y se proponen posibles soluciones.

  El proyecto que proponemos tiene como fin obtener un dispositivo muy similar al logrado en esta tesis, para lo cual, nosotros desarrollaremos nuestra propia aplicación, en lugar de utilizar las ya existentes. Lo anterior con la finalidad de ganar experiencia en el desarrollo de este tipo de software.

- **P. Samaniego Placencia, "Validación de técnicas de monitoreo para la estimación de contaminación acústica ambiental en la ciudad de Cuenca."[11]**

  Samaniego compara un dispositivo para registro de ruido ambiental integrado con un móvil y la aplicación NoiseTube, desarrollada por la compañía Sony, contra una simulación de

señales de ruido ambiental con un sonómetro profesional para un levantamiento confiable de datos.

Una vez más, la diferencia radica en que nosotros proponemos el desarrollo de nuestra propia aplicación como meta principal, con el fin de dejar la generación de desarrollo tecnológico dentro de la UAM-Azc.

- **S. Salamanca Salamanca, "Aplicación multimedia interactiva para dispositivos móviles, de análisis de parámetros Acústicos". [12]**

  En el proyecto de Salamanca consiste de una simulación de ruido ambiental que depende de 2 parámetros que son la potencia eléctrica y el efecto de directividad de la fuente haciendo una predicción, no es un software de diseño ambiental si no que obtiene valores teóricos donde no está la instrumentación, no evalúa la profundidad ni las características del entorno, como lo hace CadnaA (del inglés *Computer Aided Noise Abatement*). La semejanza es que proponemos trabajar con Android, y la diferencia es que no es una predicción sino una medida del ruido ambiental y se destinará un módulo de consulta en una base de datos.

2.3 Software:

- **Beomeochun-ro, Suseong-gu, Daegu, Republic of KOREA "Sonómetro: Sound Meter", 102/507, 126. [13]**

El software está disponible en la play store; con el nombre de: Sound Meter. Es semejante a lo que se desarrollará en este proyecto de integración. Por lo cual nos puede servir como punto de comparación. Esta aplicación registra datos de ruido y los compara con una escala de sonido de ruido ambiental. Se implementa la toma de datos y se compara con una escala de sonido en tiempo real. Se intentará que nuestro proyecto sea en la plataforma Android y esté destinando a la comparación de consultas mediante una base de datos.

Capítulo 3

Justificación

El desarrollo de una aplicación para dispositivo móvil con fines de registrar ruido ambiental es una oportunidad para el desarrollo de tecnología y software propios. Es importante mencionar que ya existen algunas aplicaciones con esta finalidad por lo que nuestro proyecto no es en sí mismo original, pero sí apoya la investigación generando nuestras propias herramientas de trabajo.

Desarrollar este tipo de aplicación permitirá medir la intensidad de ruido a la que se está expuesto de forma simple, práctica y económica y con ello, eventualmente prevenir la pérdida auditiva asociada. Comparando la intensidad del ruido con una base de datos, podríamos hacer una interfaz que asemeje un sonómetro profesional, lo cual es un tipo de *instrumento virtual*, aunque no tenga los componentes de un sonómetro real fabricado, realiza funciones similares (el termino instrumento virtual se usa en general, para referirse a desarrollos tecnológicos basados en un software que permite el uso de una computadora como instrumento de medición y fue acuñado en 2001 por nacional instruments). Consultando información almacenada en una base de datos el usuario podría analizar el ruido medido *a posteriori* y dar un uso final o una valoración del ruido.

No se pretende competir con un sonómetro profesional, pero sí desarrollar las capacidades de fabricar uno similar, para los dispositivos móviles. El éxito de este proyecto permitirá la detección y/o predicción rápida en dispositivos con parámetros característicos a partir de señales registradas en trabajos de campo.

Capítulo 4

Objetivos

4.1 General

Diseñar e implementar una aplicación para uso en dispositivos móviles, en plataforma Android, que mida señales de ruido ambiental.

4.2 Particulares

- Identificar los sensores de un dispositivo móvil, internos y externos, que permita interpretar la conversión de una señal acústica en una señal eléctrica, además de guardarla en el móvil.
- Desarrollar un módulo para el registro y grabación de una señal eléctrica como función del tiempo.
- Desarrollar un módulo de interfaz, en plataforma Android para el usuario.
- Desarrollar un módulo para la graficación de la señal registrada.
- Desarrollar un módulo de base de datos que permita guardar la señal registrada para su posterior análisis.
- Evaluar la aplicación móvil para plataforma Android.

Capítulo 5

Marco Teórico

5.1 ¿Qué es Android?

La evolución de los teléfonos ha sido uno de los principales recursos para los usuarios de hoy en día, la necesidad para poder comunicarse y trabajar ha hecho que lleguemos a la era de los smartphones y junto con ellos a las aplicaciones, las cuales son herramientas que nos facilitan la vida y son adaptables a nuestras circunstancias.

Android es el sistema operativo más común, es una versión de Linux que facilita su codificación, una de las diferencias de Android con los sistemas operativos anteriores es que evitan el uso de librerías que complicaban su uso al interactuar con GPS, trackball o touchscreens. No es desarrollado por Google como se ha mencionado en diferentes lugares, sin embargo es el que tiene el mayor participe de este proyecto, apoyado por la Open Handset Alliance (OHA) una asociación de fabricantes de Software, Hardware y compañías telefónicas.

La primera aparición oficial de Android fue en febrero de 2009 con el nombre de Banana Bread. Los nombres de Android son inspirados en la repostería y cada versión va teniendo nombres diferentes de manera alfabética.

5.2 ¿Qué es una app (Aplicación)? [5]

Es un programa diseñado para realizar una o varias tareas dependiendo el área al que vaya dirigida. La abreviaturía de la palabra Aplicación es “app”.

La app es un software diseñado para smartphones, tablets y otros dispositivos, la importancia del mismo es facilitar en cualquier momento la resolución de una tarea determinada o ayudarnos en operaciones y gestiones del día a día. Por mencionar algunas funciones, existen aplicaciones para comunicarse, para ubicarnos, para usos comerciales, etc.

La elección de Android para este proyecto de integración es por la disposición de utilizar un software libre, por que empieza el auge, la asimilación y el desarrollo de tecnología con este sistema operativo en la UAM-Azcapotzalco.

5.3 Componentes de un celular [6]

Cada smartphone tiene diferentes componentes hardware que permiten su correcto funcionamiento, y todo en conjunto establece un software (Sistema Operativo), y a su vez una app, que se muestra a través de una interfaz gráfica para el usuario final. Las piezas más comunes dentro de un smartphone son:

- Batería: Proporciona la alimentación, los tamaños y las especificaciones varían, existe un estándar de rango de voltaje que es de 3.7 v., y es recargable.
- Antena: Encargada de amplificar e interceptar la señal de la red.
- LCD: Programado para dar la visualización de la información.
- Cámara: Utilizado para capturar imágenes.
- On / Off: interruptor de encendido y apagado, también están otros interruptores como el de volumen.

- Micrófono: Es un componente que intercepta y emite los sonidos al receptor, imita la voz humana, también es conocido como portavoz, este dispositivo es muy pequeño por su diferente embalaje[1], pues está diseñado para los móviles, grabadoras de bolsillo.

5.4 Micrófonos Electret [7]

Los micrófonos han revolucionado desde 1900 a la fecha, hoy en día no se necesita de polarización o Phantom Power, resultando en el modelo de micrófono de condensador conocido como transductor electret.

Utiliza un electrodo que por definición está polarizado, es decir, está cargado desde su fabricación y puede durar así más de 50 años. Posee una respuesta de frecuencia de 50Hz a 15000 kHz y son omnidireccionales, lo cual permite una mayor sensibilidad

Su composición consiste en dos placas, una fija perforada externa y otra móvil de plástico con una cara de contacto de metal, la cual es inducida a una carga electrostática y a una temperatura de 200°C durante su fabricación para polarizarla; dos electrodos metálicos paralelos, un aislante de material dieléctrico polarizado, un tubo de ecualización y dos contactos de salida.

Su uso se ha extendido a distintos dispositivos los cuales incluyen megáfonos, computadoras, grabadoras portátiles y pos supuesto celulares.

Existen tres tipos de Micrófonos electret:

- De tipo lámina o tipo diafragma: el más común y de menor calidad ya que usa una capa de electret como diafragma.
- Electret posterior: el electret se encuentra al posterior de la placa de la cápsula del micrófono y el diafragma no tiene carga.
- Electret frontal: no tiene placa posterior y el condensador está conformado por el diafragma y la superficie interior de la cápsula.

El uso del elecret presenta varias ventajas sobre el resto de los micrófonos, entre las cuales encontramos que: posee una respuesta de frecuencia de todo el rango audible, la señal que entrega es más alta, tiene una sensibilidad en el rango de -50 y -70 dB, utiliza de 1.2 a 9 V y tiene un tamaño pequeño, lo cual resulta conveniente.

Sin embargo, también presenta algunas desventajas: el preamplificador se satura con una presión sonora alta, tiene una baja respuesta en tonos altos, es susceptible a cambios de temperatura y presencia de polvo y tiene una alta impedancia aunque menor a los otros micrófonos de condensador.

[1] Los micrófonos de celulares se adaptan a la carcasa, son económicos y resistentes.

Capítulo 6

Desarrollo del Proyecto

En este capítulo se explica el desarrollo de la App móvil en Android Studio (SoundB), en la Sección 6.1, 6.2. 6.3 y 6.4 se muestran los casos de uso con diagramas del Lenguaje Modelado Unificado (UML), que describen los pasos y las actividades que deberán llevar a cabo para el proceso, posteriormente se describen las primitivas del UML, y finalmente las pruebas de la misma y su funcionamiento.

6.1 UML del caso General

En la Figura 1, mostramos como el usuario abre la aplicación móvil que despliega la interfaz mientras que las consultas de los datos se visualizan en la pág. Web extraídas desde la BD, y en la Tabla 2, se muestran los elementos de la misma.

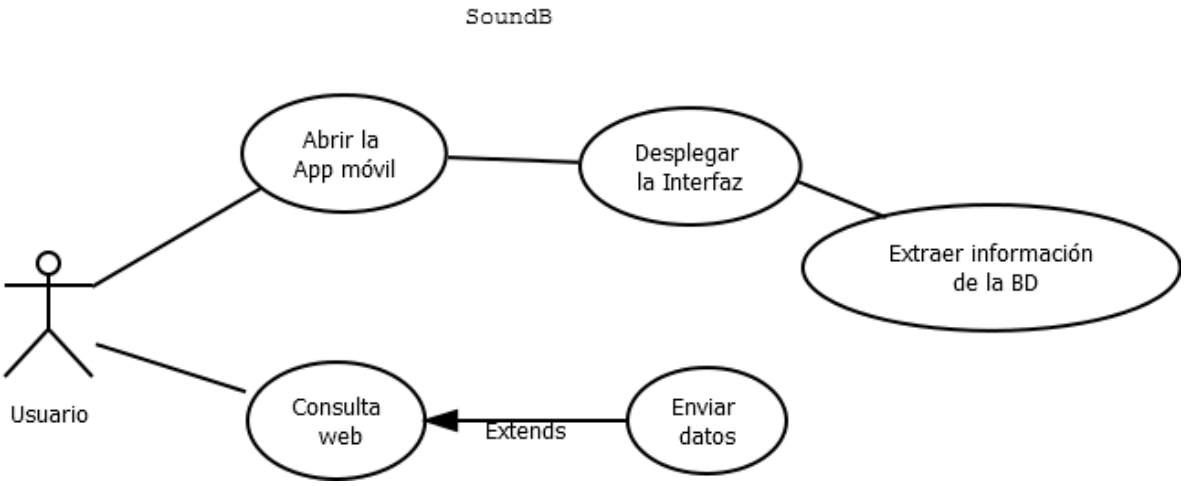


Figura 1: UML General.

Nombre:	Caso de uso general
ID:	cu-01
Actores:	Usuario
Pre-condiciones:	Ninguna
Post-condiciones:	El usuario podrá extraer y enviar información para la manipulación de datos.
Eventos:	1. El usuario abre la aplicación móvil (SoundB). 2. SoundB despliega la pantalla principal. 3. El usuario abre la pág. web. 4. El usuario hace una consulta web. 4.1 Los datos se envían desde la BD a la web.
Caminos alternativos:	Ninguna

Tabla 2: Primitiva General.

6.1.1 Interfaz gráfica del módulo general

El usuario abre la aplicación móvil (SoundB), la cual muestra la interfaz presentada en la Figura 2, mientras que el código de dicha interfaz se visualiza en los Apéndices B y B1.

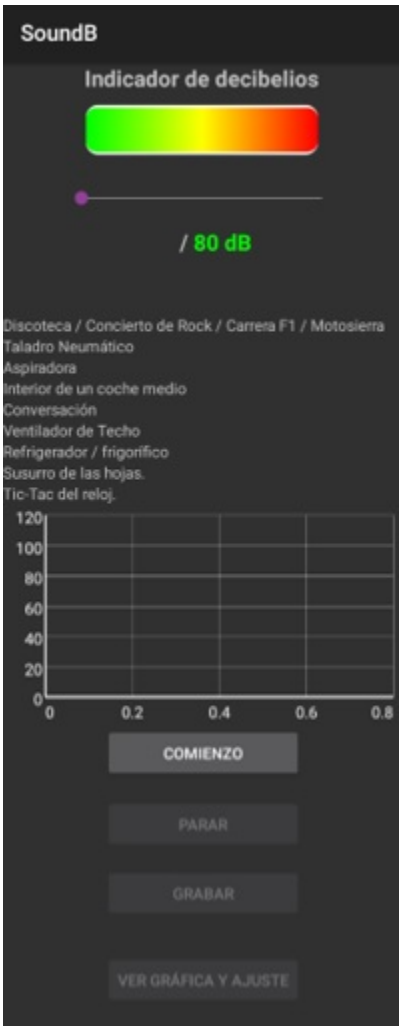


Figura 2: Interfaz Principal.

El usuario abre la pág. Web y para saber que la consulta fue exitosa, muestra “conexión exitosa” como en la Figura 19.

La página web muestra todos los datos de ruido, mostrada en la Figura 20.

Los datos se envían desde la BD a la web, visualizada en la Figura 21.

6.2 UML Visualizar Datos

En la Figura 6 se muestra el diagrama que pasa por la interfaz de la aplicación móvil y los pasos a seguir se visualizan en la Tabla3.

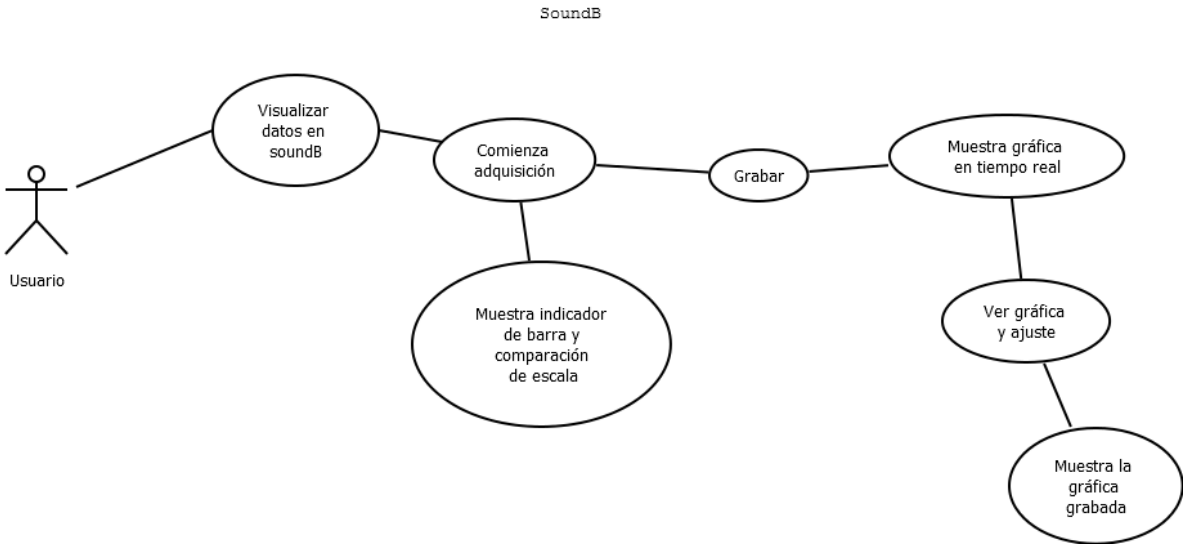


Figura 3: UML Visualizar Datos.

Nombre:	Caso de uso visualizar datos
ID:	cu-02
Actores:	Usuario
Pre-condiciones:	Contar con la aplicación abierta
Post-condiciones:	El usuario podrá ver la(s) gráfica(s) obtenida(s)
Eventos:	1. Visualizar datos en SoundB. 2. La App muestra la gráfica vacía y el indicador sin datos. 3. El usuario selecciona comienzo. 4. La App despliega la gráfica dinámica y el indicador de decibelios, comparando la información obtenida. 5. El usuario selecciona grabar. 6. El usuario selecciona ver gráfica y Ajuste. 7. La App despliega los datos obtenidos y la gráfica de la misma. 8. El usuario selecciona guardar datos en la nube para subir los datos a la BD.
Caminos alternativos:	4.1 El usuario puede seleccionar parar, para dejar de manera estática la app. 7.1 El usuario puede seleccionar mínimos cuadrados para ver el ajuste de la gráfica. 7.2 El usuario puede seleccionar ver ambas gráficas para ver la gráfica obtenida y el ajuste.

Tabla 3: Primitiva Visualizar Datos.

6.2.1 Interfaz gráfica del módulo para visualizar datos

Con la App abierta se muestra la gráfica vacía y el indicador sin datos (Figura 2). El usuario selecciona comienzo (Figura 4).

Figura 4: Botón Comienzo.

La App despliega la gráfica dinámica y el indicador de decibelios, comparando la información obtenida como se aprecia en la Figura 5.

Figura 5: Interfaz Dinámica.

El usuario puede seleccionar parar, para dejar de manera estática la app (Figura 6).

Figura 6: Botón parar.

El usuario selecciona grabar (Figura 7).

Figura 7: Botón de grabar.

El usuario selecciona ver gráfica y Ajuste (Figura 8).

Figura 8: Botón de ver gráfica y ajuste.

La App despliega los datos obtenidos y la gráfica de la misma (Figura 9).

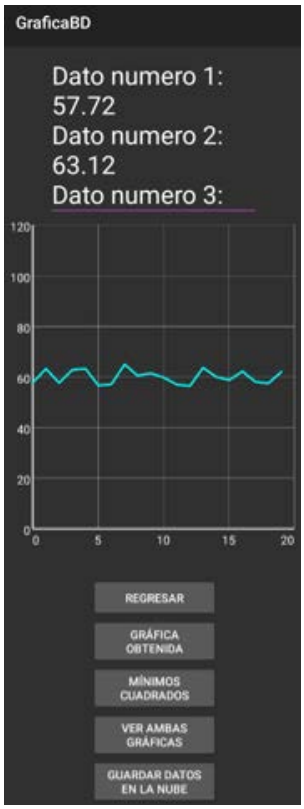


Figura 9: Segunda interfaz de los datos obtenidos.

El usuario puede seleccionar mínimos cuadrados para ver el ajuste de la gráfica (Figura 10 y 11).

Figura 10: Botón para ver mínimos cuadrados.

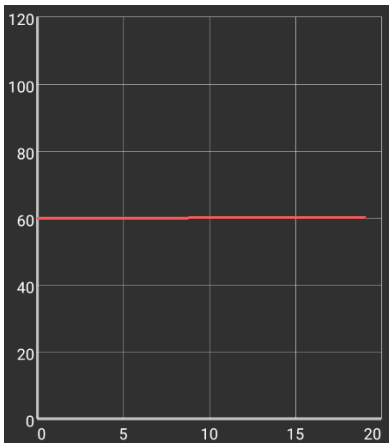


Figura 11: Gráfica de mínimos cuadrados.

El usuario puede seleccionar ver ambas gráficas para ver la gráfica obtenida y el ajuste (Figura 12 y 13).

Figura 12: Botón para ver gráfica y ajuste.

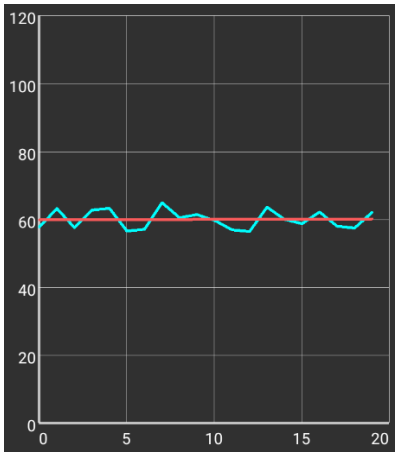


Figura 13: Gráfica de los datos obtenidos y el ajuste.

El usuario selecciona guardar datos en la nube para subir los datos a la BD (Figura 14).

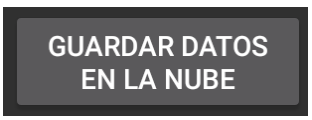


Figura 14: Botón para subir los datos a la BD.

6.3 UML Obtener Datos

Para Obtener los datos mostramos el siguiente diagrama (Figura 15), donde los botones seleccionados por el usuario realizan este procedimiento, en la primitiva se visualizan los datos (Tabla 4).

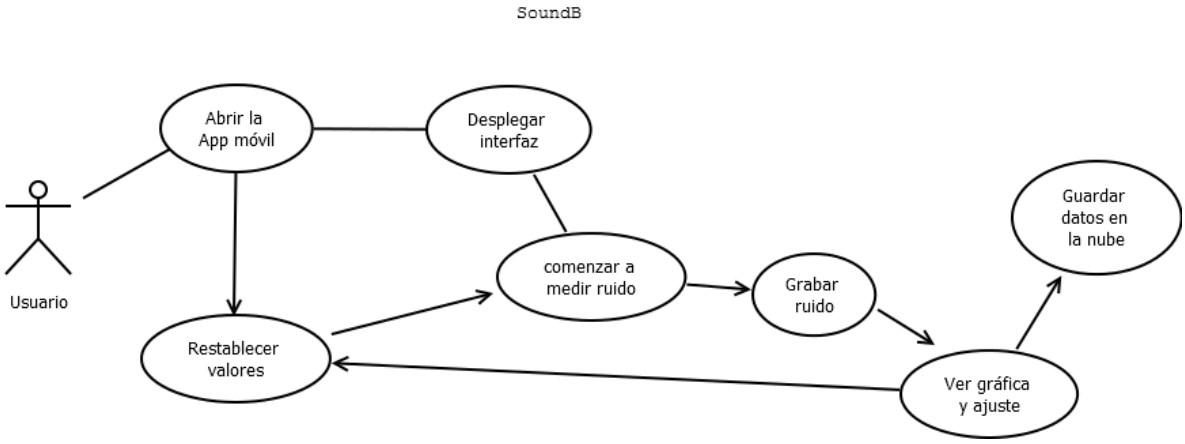


Figura 15: UML Obtener Datos.

Nombre:	Caso de uso Obtener datos
ID:	cu-03
Actores:	Usuario
Pre-condiciones:	Visualizar datos en SoundB.
Post-condiciones:	1. Los datos son mostrados en la App móvil. 2. Los datos serán almacenados en la BD (nube).
Eventos:	1. El usuario selecciona comienzo. 2. El usuario selecciona grabar. 3. La App obtendrá datos de ruido en tiempo real. 4. El usuario selecciona ver gráfica y Ajuste. 5. La App muestra datos en un intervalo de 20 seg. 6. El usuario selecciona guardar datos en la nube.
Caminos alternativos:	4.1 El usuario puede seleccionar regresar, para restablecer los valores.

Tabla 4: Primitiva Obtener Datos.

6.3.1 Interfaz gráfica del módulo para obtener datos

Para obtener los datos, previamente estarán visualizados en la aplicación móvil y saber que los datos serán almacenados en la BD, el código de obtención de datos está en el Apéndice C.

El usuario selecciona comienzo (Figura 4).

Posteriormente el usuario selecciona grabar (Figura 7).

La App obtendrá datos de ruido en tiempo real y se mostrará una barra de progreso como en la figura 16, que el usuario podrá visualizar.

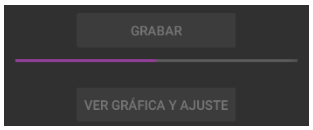


Figura 16: Barra de Progreso en la extracción de datos.

El usuario selecciona ver gráfica y ajuste (Figura 8).

La App muestra datos en intervalo de 20 segundos. (Figura 9).

El usuario puede seleccionar regresar, para restablecer los valores (Figura 17).

Figura 17: Botón regresar para restablecer valores.

6.4 UML Consulta

A través de una pág. Web el usuario visualiza los datos almacenados en la BD, a esto se le llama una consulta y es el diagrama de la Figura 18. En la tabla 5, se visualizan los eventos del diagrama.

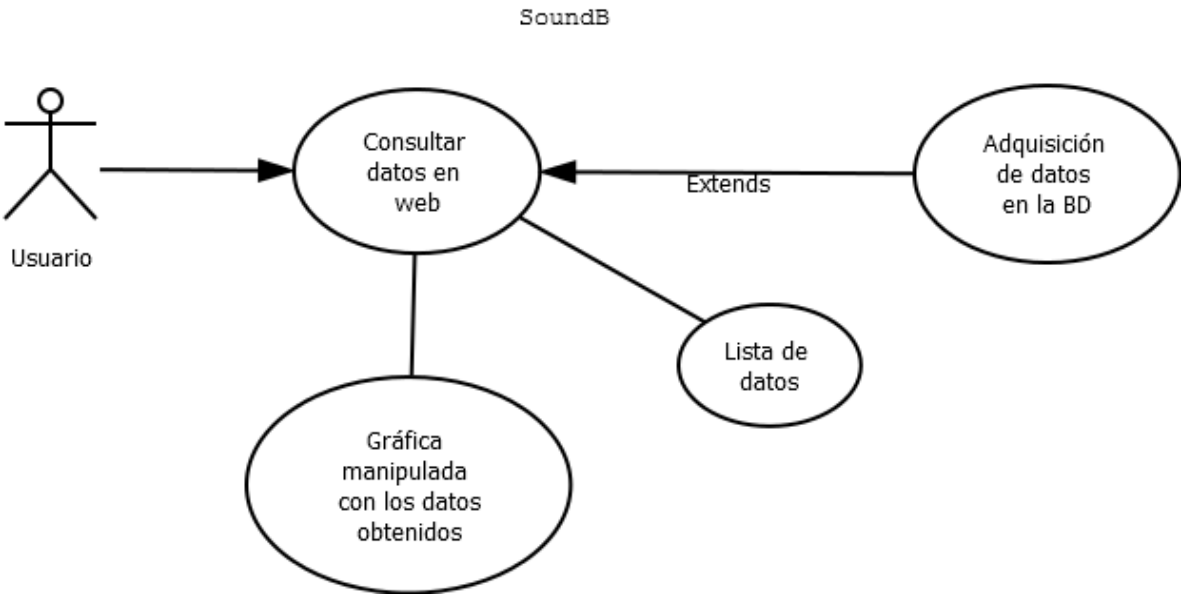


Figura 18: UML Consulta.

Nombre:	Caso de uso Consulta
ID:	cu-04
Actores:	Usuario
Pre-condiciones:	Contar con la página web abierta.
Post-condiciones:	La lista de la BD con información obtenida será desplegada.
Eventos:	1. El usuario se posiciona en la pág. web. 2. El usuario hace una consulta web sobre datos de ruido. 3. La página extrae los datos de la BD. 4. La página web muestra todos los datos de ruido.
Caminos alternativos:	Ninguno

Tabla 5: Primitiva Consulta.

6.4.1 Interfaz gráfica del módulo de consulta

Para realizar la prueba de consulta, previamente tendremos abierta la página web.

El usuario abre la pág. Web y para saber que la consulta fue exitosa, muestra "conexión exitosa" como en la Figura 19.

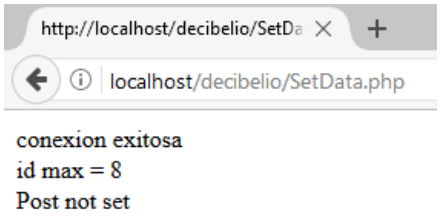


Figura 19: Localhost SetData.

La página web muestra todos los datos de ruido, mostrada en la Figura 20.

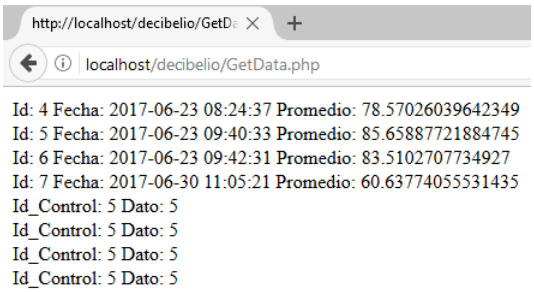


Figura 20: Localhost GetData.

Los datos se envían desde la BD a la web, visualizada en la Figura 21.

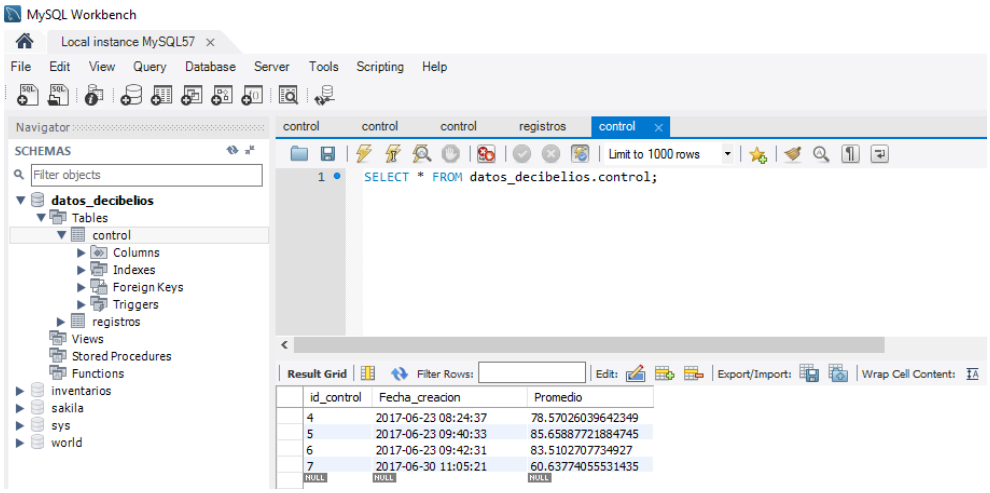


Figura 21: Datos en MySQL.

Capítulo 7

Pruebas y Resultados

7.1 Pruebas

Las pruebas realizadas fueron hechas en el centro de enlace estudiantil de química C.E.C.I.Q. dentro de la universidad, las pruebas también corresponden a los UML de visualizar datos, obtener datos y consulta.

7.1.1 Pruebas de Visualizar Datos

Toma de datos entre las 7 de la mañana y entre las 12 del día, en el celular se abre la aplicación móvil (SoundB), la interfaz que nos arroja es una gráfica vacía y un indicador sin datos.

Dentro de nuestra aplicación móvil puchamos el botón comienzo (Figura 4), en el mismo caso fue para las 7 am. y 12 pm.

La aplicación móvil empezó a desplegar la gráfica dinámica y el indicador de decibelios también empezó a detectar datos y los compara con alguna escala.

Figura 22: Interfaz Dinámica entre las 7 am.

Figura 23: Interfaz Dinámica entre las 12 pm.

Para ambos casos apretamos el botón de grabar (Figura 7), aparece una barra de progreso que nos indica que se están obteniendo datos (Figura 16) y la aplicación nos habilita el botón ver gráfica y ajuste (Figura 8), al presionarlo nos mostrará los datos recopilados (Figura 24 y 25).

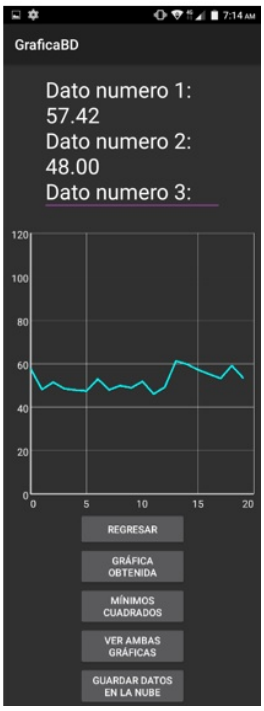


Figura 24: Datos recopilados entre las 7 am.

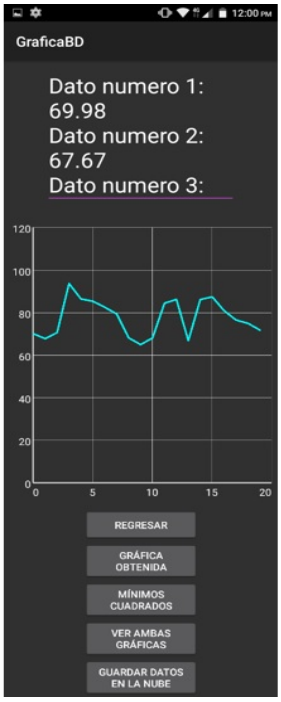


Figura 25: Datos recopilados entre las 12 pm.

Seleccionando el botón mínimos cuadrados (Figura 10), observamos el ajuste de la gráfica (Figura 26 y 27).

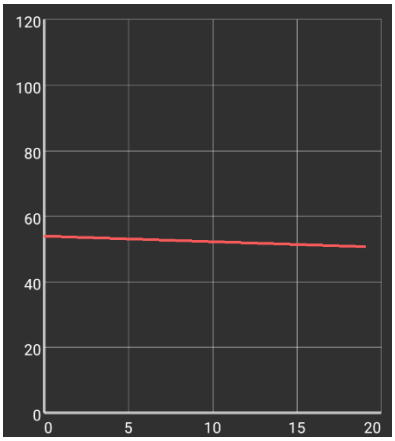


Figura 26. Gráfica de ajuste entre las 7 am.

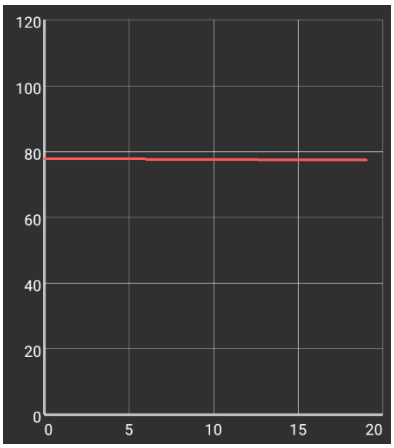


Figura 27. Gráfica de ajuste entre las 12 pm.

Al seleccionar el botón ver ambas gráficas (Figura 12), visualizamos las siguientes gráficas (Figuras 28 y 29).

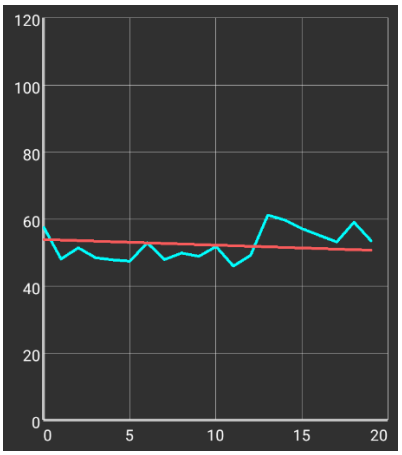


Figura 28. Gráfica de los datos obtenidos y el ajuste entre las 7 am.

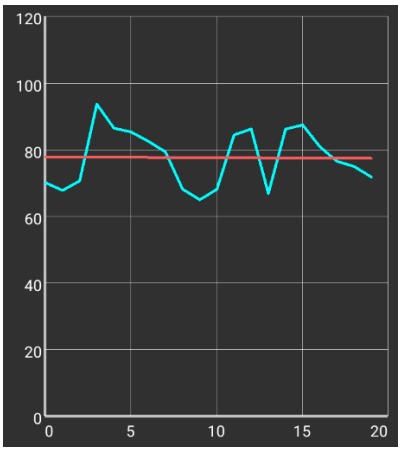


Figura 29. Gráfica de los datos obtenidos y el ajuste entre las 12 pm.

Por último, seleccionamos el botón guardar datos en la nube para subir los datos a la BD (Figura 14), una vez terminado el proceso el botón se deshabilitará (Figura 30).

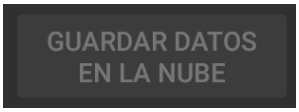


Figura 30: Botón deshabilitado al subir los datos a la BD.

7.1.2 Pruebas de Obtener Datos

En la obtención de datos, realizaremos algo similar que en el punto 7.1.1, pero en esta parte lo tenemos que ver de una manera más sólida como los datos enviados a la Base de Datos, que finalmente terminan en la Figura 31 y 32 con los datos dentro de MySQL.

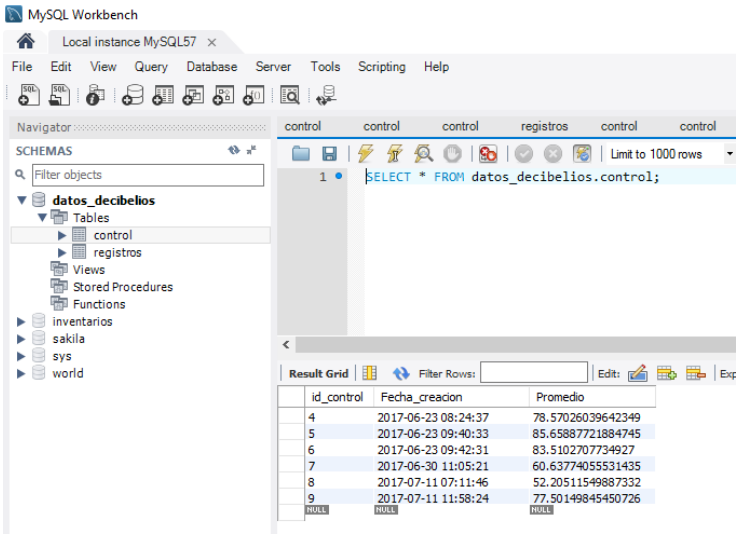


Figura 31: Datos de control guardados en MySQL.

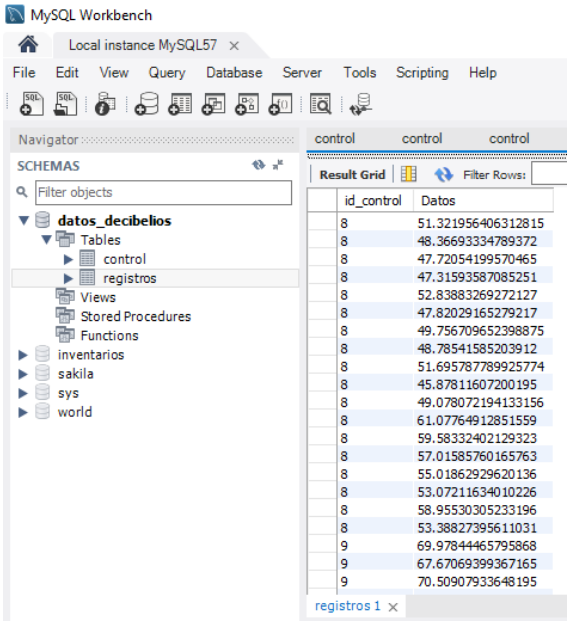


Figura 32: Datos de registros guardados en MySQL.

7.1.3 Pruebas de Consulta

Para realizar la prueba de consulta, previamente tendremos abierta la página web.

El usuario abre la pág. Web y para saber que la consulta fue exitosa, introduce la url http://localhost/decibelio/SetData.php en el navegador, así para esta prueba aparece el número 9 que fue la toma de datos entre las 7 am. mostrada en la Figura 33 y la Figura 34 la de las 12 pm.

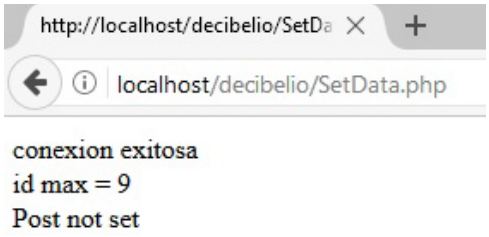


Figura 33: Consulta exitosa SetData entre las 7 am.

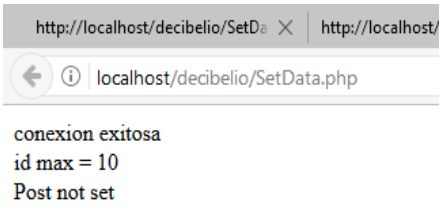


Figura 34: Consulta exitosa SetData entre las 12pm.

Todos los datos se envían desde la BD a la web y se puede ver en la Figura 35 y 36. La prueba realizada entre las 7 am, es la que dice Id: 8 Fecha: 2017-07-11 07:11:46 52.20511549887332. Y la prueba de las 12 pm es Id:9 Fecha: 2017-07-11 11:58:24 Promedio: 77.50149845450726, además de incluir Id correspondiente a ese valor (Figuras 37 y 38).

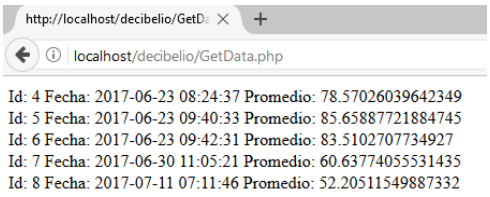


Figura 35: Consulta exitosa GetData entre las 7 am.

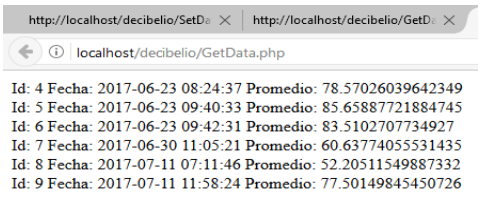


Figura 36: Consulta exitosa GetData entre las 12 pm.

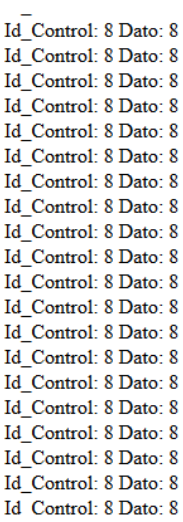


Figura 37: Consulta exitosa GetData Id entre las 7 am.

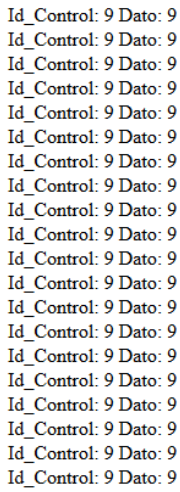


Figura 38: Consulta exitosa GetData Id entre las 12 pm.

7.1.4 Pruebas con Sound Meter

Aquí hacemos una comparación con la aplicación móvil que ya está en el mercado y que se puede descargar de la Play Store y nuestra aplicación móvil (SoundB), también se tomó registros entre las 7 am y entre las 12 pm (Figura 39 y 40).

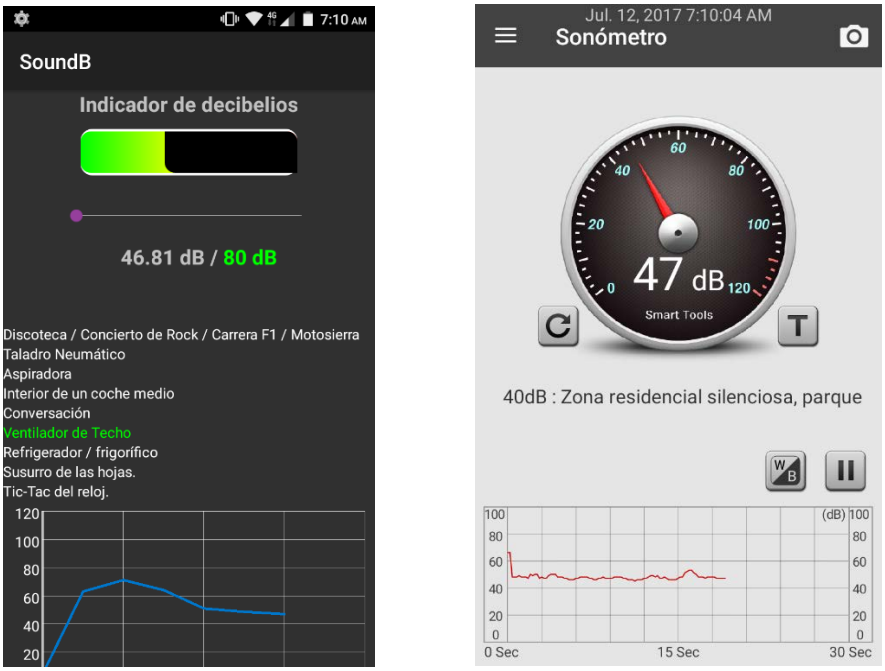


Figura 39: Comparación de Aplicaciones entre las 7 am.

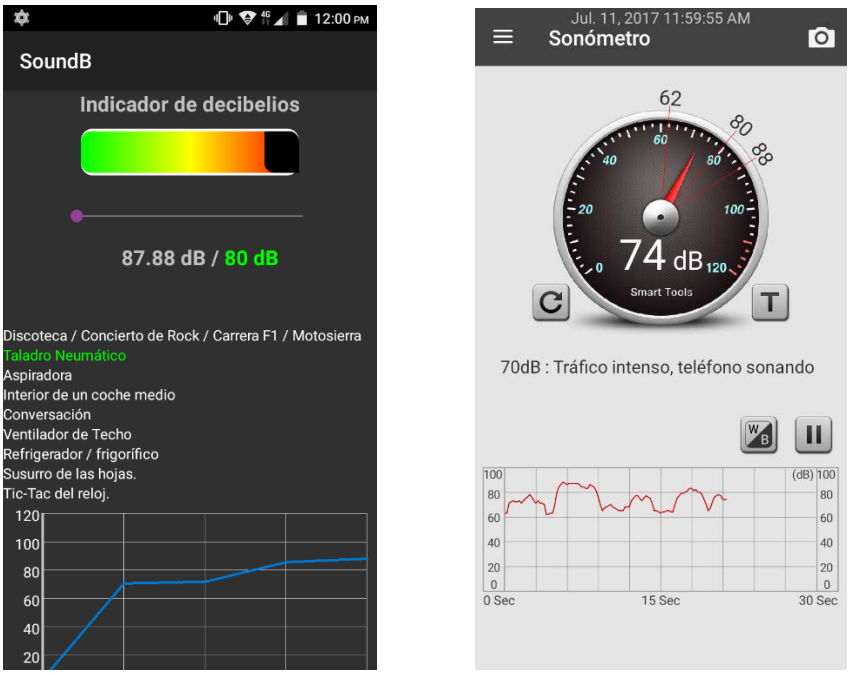


Figura 40: Comparación de Aplicaciones entre las 12 pm.

7.1.5 Pruebas con un Sonómetro Profesional

Se instaló la aplicación en diferentes dispositivos se corrieron al mismo tiempo a lado de un sonómetro profesional sólo para ver las escalas de cada uno (Figura 41).

Figura 41: Toma de ruido con varios dispositivos móviles y un sonómetro profesional.

7.2 Análisis y discusión de resultados.

La toma de sonido tanto en la mañana (entre las 7 am.) como en la tarde (entre las 12 pm.), muestra la comparación de niveles registrados en tiempo real (Figuras 22 y 23), en la toma dentro de las 7 am. se presenta una escala de dB por debajo de los 70 dB y la recta de la gráfica parece descender, además de que en la BD se muestra un promedio de 52.20511549887332, mientras que el registrado a las 12 pm. La recta de la gráfica parece ascender por arriba de los 70 dB así como el indicador y en la BD se muestra un promedio mayor que el otro de 77.50149845450726.

Las comparaciones nos muestran que en la mañana dentro del C.EC.I.Q. se presenta menor índice de ruido que en la tarde, a modo de observación hay menos personas en la mañana dentro del espacio que en la tarde, por ello se observa esa comparación de la aplicación móvil (SoundB), además de compararlo con la aplicación en la Play store (Sound Meter) y un sonómetro profesional.

Como resultado final se creó una interfaz amigable para el usuario, sencillo de entender y creada en la plataforma de Android Studio. En la interfaz se puede observar el indicador de decibelios que muestra en tiempo real el sonido, un botón de comienzo, uno para pararlo y otro para grabar en un lapso de tiempo y poder visualizar la toma de sonido en una gráfica, la cual es ajustable y la información que genera se almacena en MySQL. De manera final y exitosa le damos al usuario una herramienta con la cual puede tomar decisiones más rápidas en el área donde se desempeña y evitar los daños causados por exposición al ruido.

Capítulo 8

Conclusiones

Los objetivos del proyecto se concluyeron de manera satisfactoria al desarrollar una aplicación en Android que servirá como herramienta y que generará un desarrollo tecnológico propio para la universidad, esto contribuye con el cumplimiento de los objetivos perseguidos por el Laboratorio de Sistemas Dinámicos y Prototipos (LSDP) de la UAM Azcapotzalco.

En las pruebas obtenidas, cada uno presenta un desarrollo satisfactorio, se muestra la interfaz amigable que da como resultado la obtención y almacenado de los datos, así como la visualización de gráficas, permitiendo al usuario final poder manipularlos, mientras que, al estar comparando resultados con aplicaciones en el mercado, también arroja datos similares, incluso comparándolo con un sonómetro profesional.

Esta herramienta permite al usuario reducir costos, ser accesibles dentro de un dispositivo móvil y tenerlo en todo momento para su posterior análisis o registro de datos, llevando al usuario a una toma de decisiones más rápida, el proyecto queda abierto a recibir mejoras en el diseño de la interfaz y en agregar nuevas características para que la aplicación sea cada vez más parecida a un sonómetro profesional.

Referencias Bibliográficas

Páginas web:

[1] A. C.G, "Pérdida de audición o hipoacusia", *Infoaudifonos.net*, 2016. [En Línea]. Disponible: http://www.infoaudifonos.net/perdida-auditiva. [Último Acceso: 22- Agto - 2016].

[2] "Efectos a la salud por ruido", 2016. [En Línea]. Disponible: http://salud.edomex.gob.mx/cevece/doc/Documentos/Efecs_ruido.pdf. [Último Acceso: 22- Agto - 2016].

[3] www.ordenjuridico.gob.mx, 2016. [En Línea]. Disponible: http://www.ordenjuridico.gob.mx/Documentos/Estatal/Distrito%20Federal/wo97775.pdf. [Último Acceso: 16- Jun - 2016].

[4] "IEC 61672-1 (electroacústica sonómetros parte 1) | Marco Torres Céspedes - Academia.edu", *Academia.edu*, 2016. [En Línea]. Disponible: http://www.academia.edu/8958482/IEC_61672-1_electroacustica_sonometros_parte_1_. [Último Acceso: 22- Agto - 2016].

[5] "Línea VerdeAstillero", *Lineaverdeastillero.com*, 2017. [En Línea]. Disponible: http://www.lineaverdeastillero.com/lv/consejos-ambientales/apps-ambientales/que-es-una-app.asp. [Último Acceso: 10- Jul- 2017].

[6] V. →, "Partes de los celulares", *Curso de celulares*, 2017. [En Línea]. Disponible: https://cursodecelulares.wordpress.com/2011/05/13/piezas-en-telefonos-moviles/. [Último Acceso: 10- Jul- 2017].

[7] "Informe sobre Micrófonos Electret", *Scribd*, 2017. [En Línea]. Disponible: https://es.scribd.com/doc/151559153/Informe-sobre-Microfonos-Electret. [Último Acceso: 10- Jul- 2017].

Proyectos de Integración o terminales:

[8] M. Hernández Nieves, "Clasificación de llanto de bebé con wavelets", Proyecto Terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, 2012.

[9] A. Espinosa Sánchez, "Simulación de la solución al problema directo y el problema inverso de reflectrometría acústica", Proyecto Terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, 2013.

Tesis:

[10] M. López Fernández, "análisis y evaluación del ruido mediante el uso de aplicaciones móviles", Utilización de sensores de bajo costo para el desarrollo de prácticas de laboratorio en tecnología: Análisis y evaluación del ruido mediante el uso de aplicaciones móviles, Universidad de Valladolid. Escuela Técnica Superior de Ingeniería Informática, 2014.

[11] P. Samaniego Placencia, "Validación de técnicas de monitoreo para la estimación de contaminación acústica ambiental en la ciudad de Cuenca.", Maestría en Geomática con mención en Ordenamiento Territorial, Universidad del Azuay, 2015.

[12] S. Salamanca Salamanca, "Aplicación multimedia interactiva para dispositivos móviles, de análisis de parámetros Acústicos", Ingeniería de Sonido, Universidad de San Buenaventura, 2013.

Software:

[13] "Android boy's Lab: Sound Meter (v1.6) manual", *Androidboy1.blogspot.mx*, 2015 [En Línea]. Disponible: http://androidboy1.blogspot.mx/2015/08/sound-meter-v16.html [Último Acceso: 16- Jun- 2016].

APÉNDICE A

El decibel (dB) es una unidad de medida adimensional relativa, esto es, expresa la razón entre dos señales, la que se desea medir y una señal de referencia. Así, si se mide la potencia de ambas señales, digamos, P1 y P2 se tiene la siguiente definición de un decibel (1 dB):

$$dB = 10 \log_{10} (P1/P2) \quad \text{.....(1)}$$

Esta unidad de medida se usa comúnmente en acústica y telecomunicaciones para determinar la relación entre una señal bajo estudio y una de referencia (de muy poca intensidad). Esta última usualmente corresponde al umbral mínimo de percepción del sonido (20 micropascales). Como la escala de sonido es logarítmica, es importante notar que cero belios, corresponde a medir una señal cuya magnitud es igual a la magnitud de referencia; un belio equivale a 10 decibelios y lo cual significa que la potencia del primero es 10 veces la magnitud de referencia. Mientras que, dos belios representan un aumento de cien veces en la potencia; tres belios, equivalen a un aumento de mil veces y así sucesivamente, lo cual puede obtenerse de la definición de la ecuación (1).

En términos de presión acústica definimos un decibel como:

$$L_{dB} = 10 \log (p/p_0)^2$$

Figura 42: Definición de la presión acústica **[3]**.

p es la presión acústica medida
p_0 es la presión acústica de referencia, igual a 20 µPa
L_{dB} es el nivel de la presión acústica medida expresado en dB

APÉNDICE B

La aplicación que se desarrolló para los dispositivos móviles, se ve reflejado en el siguiente código Java implementado en Android Studio (Activity.Java).

Botones

```
private Button GrabarDatos, Comienza, Para, VerGrafica;
```

Desactiva el botón y no puede oprimirse

```
GrabarDatos.setEnabled(false);

Comienza = (Button) findViewById(R.id.Comenzar);

Para = (Button) findViewById(R.id.Parar);
Para.setEnabled(false);

VerGrafica = (Button) findViewById(R.id.Grafica);
VerGrafica.setEnabled(false);

Barra_Tiempo = (ProgressBar) findViewById(R.id.barraTiempo);

decibelsTx = (TextView) findViewById(R.id.decibels);
barDB = (RelativeLayout) findViewById(R.id.bardb);
seekbarDB = (SeekBar) findViewById(R.id.seekbar_db);
```

```
    decibeliosSeekbar = (TextView) findViewById(R.id.decibelios_seekbar);

    rui1=(TextView)findViewById(R.id.Ruido1);
    rui2=(TextView)findViewById(R.id.Ruido2);
    rui3=(TextView)findViewById(R.id.Ruido3);
    rui4=(TextView)findViewById(R.id.Ruido4);
    rui5=(TextView)findViewById(R.id.Ruido5);
    rui6=(TextView)findViewById(R.id.Ruido6);
    rui7=(TextView)findViewById(R.id.Ruido7);
    rui8=(TextView)findViewById(R.id.Ruido8);
    rui9=(TextView)findViewById(R.id.Ruido9);

    seekbarDB.setOnSeekBarChangeListener(this);

    // we get graph view instance
```

Obtenemos el elemento de gráfica

```
    graph = (GraphView) findViewById(R.id.graph);
```

Datos para dibujar la gráfica

```
    series = new LineGraphSeries<DataPoint>();
    graph.addSeries(series);
    // customize a little bit viewport
    Viewport viewport = graph.getViewport();
    viewport.setYAxisBoundsManual(true);
    viewport.setMinY(0);
    viewport.setMaxY(120);
    viewport.setScrollable(true);
```

Activa y desactiva los botones

```
    Comienza.setEnabled(false);
    Para.setEnabled(true);
    GrabarDatos.setEnabled(true);
  }
```

Detiene la escucha del micrófono

```
  public void paraEstado(View view){
    escucha.setListening(false);
    escucha.Reset();
    stop();
    escucha = null;
```

Activa y desactiva los botones

```
    Comienza.setEnabled(true);
    Para.setEnabled(false);
    GrabarDatos.setEnabled(false);
  }
```

Detiene la escucha de micrófono

```
    if (escucha != null) {
      escucha.setListening(false);
```

```
    escucha.Reset();
    stop();
    escucha = null;
}
```

Activa y desactiva los botones

```
Comienza.setEnabled(true);
Para.setEnabled(false);
GrabarDatos.setEnabled(false);
```

Cambia el color de los textos según el valor de decibelios

```
    if(value<=20){
      rui1.setTextColor(Color.GREEN);

      rui2.setTextColor(Color.WHITE);
      rui3.setTextColor(Color.WHITE);
      rui4.setTextColor(Color.WHITE);
      rui5.setTextColor(Color.WHITE);
      rui6.setTextColor(Color.WHITE);
      rui7.setTextColor(Color.WHITE);
      rui8.setTextColor(Color.WHITE);
      rui9.setTextColor(Color.WHITE);
    }else if(value>20 && value<=30){
      rui2.setTextColor(Color.GREEN);

      rui1.setTextColor(Color.WHITE);
      rui3.setTextColor(Color.WHITE);
      rui4.setTextColor(Color.WHITE);
      rui5.setTextColor(Color.WHITE);
      rui6.setTextColor(Color.WHITE);
      rui7.setTextColor(Color.WHITE);
      rui8.setTextColor(Color.WHITE);
      rui9.setTextColor(Color.WHITE);
    }else if(value>30 && value<=40){
      rui3.setTextColor(Color.GREEN);

      rui1.setTextColor(Color.WHITE);
      rui2.setTextColor(Color.WHITE);
      rui4.setTextColor(Color.WHITE);
      rui5.setTextColor(Color.WHITE);
      rui6.setTextColor(Color.WHITE);
      rui7.setTextColor(Color.WHITE);
      rui8.setTextColor(Color.WHITE);
      rui9.setTextColor(Color.WHITE);
    }else if(value>40 && value<=50){
      rui4.setTextColor(Color.GREEN);

      rui1.setTextColor(Color.WHITE);
      rui2.setTextColor(Color.WHITE);
      rui3.setTextColor(Color.WHITE);
      rui5.setTextColor(Color.WHITE);
      rui6.setTextColor(Color.WHITE);
      rui7.setTextColor(Color.WHITE);
      rui8.setTextColor(Color.WHITE);
      rui9.setTextColor(Color.WHITE);
```

```
}else if(value>50 && value<=60){
   rui5.setTextColor(Color.GREEN);

   rui1.setTextColor(Color.WHITE);
   rui2.setTextColor(Color.WHITE);
   rui3.setTextColor(Color.WHITE);
   rui4.setTextColor(Color.WHITE);
   rui6.setTextColor(Color.WHITE);
   rui7.setTextColor(Color.WHITE);
   rui8.setTextColor(Color.WHITE);
   rui9.setTextColor(Color.WHITE);
}else if(value>60 && value<=70){
   rui6.setTextColor(Color.GREEN);

   rui1.setTextColor(Color.WHITE);
   rui2.setTextColor(Color.WHITE);
   rui3.setTextColor(Color.WHITE);
   rui4.setTextColor(Color.WHITE);
   rui5.setTextColor(Color.WHITE);
   rui7.setTextColor(Color.WHITE);
   rui8.setTextColor(Color.WHITE);
   rui9.setTextColor(Color.WHITE);
}else if(value>70 && value<=80){
   rui7.setTextColor(Color.GREEN);

   rui1.setTextColor(Color.WHITE);
   rui2.setTextColor(Color.WHITE);
   rui3.setTextColor(Color.WHITE);
   rui4.setTextColor(Color.WHITE);
   rui5.setTextColor(Color.WHITE);
   rui6.setTextColor(Color.WHITE);
   rui8.setTextColor(Color.WHITE);
   rui9.setTextColor(Color.WHITE);
}else if(value>80 && value<=100){
   rui8.setTextColor(Color.GREEN);

   rui1.setTextColor(Color.WHITE);
   rui2.setTextColor(Color.WHITE);
   rui3.setTextColor(Color.WHITE);
   rui4.setTextColor(Color.WHITE);
   rui5.setTextColor(Color.WHITE);
   rui6.setTextColor(Color.WHITE);
   rui7.setTextColor(Color.WHITE);
   rui9.setTextColor(Color.WHITE);
}else if(value>100){
   rui9.setTextColor(Color.GREEN);

   rui1.setTextColor(Color.WHITE);
   rui2.setTextColor(Color.WHITE);
   rui3.setTextColor(Color.WHITE);
   rui4.setTextColor(Color.WHITE);
   rui5.setTextColor(Color.WHITE);
   rui6.setTextColor(Color.WHITE);
   rui7.setTextColor(Color.WHITE);
   rui8.setTextColor(Color.WHITE);
}
```

Función para pasar los valores a un texto y poder mostrarlo en un elemento en pantalla

```
  public void setDatos() {
    for (int i = 0; i < datos.length; i++) {
      muestradatos = muestradatos + "Dato numero " + (i + 1) + ": " + new
Formatter().format("%03.2f", datos[i]).toString() + "\n";
    }
  }
```

Dibuja la gráfica que se obtiene de los datos recogidos del micrófono

```
  public void graficaNormal() {
    series = new LineGraphSeries<DataPoint>(new DataPoint[]{
        new DataPoint(0, datos[0]),
        new DataPoint(1, datos[1]),
        new DataPoint(2, datos[2]),
        new DataPoint(3, datos[3]),
        new DataPoint(4, datos[4]),
        new DataPoint(5, datos[5]),
        new DataPoint(6, datos[6]),
        new DataPoint(7, datos[7]),
        new DataPoint(8, datos[8]),
        new DataPoint(9, datos[9]),
        new DataPoint(10, datos[10]),
        new DataPoint(11, datos[11]),
        new DataPoint(12, datos[12]),
        new DataPoint(13, datos[13]),
        new DataPoint(14, datos[14]),
        new DataPoint(15, datos[15]),
        new DataPoint(16, datos[16]),
        new DataPoint(17, datos[17]),
        new DataPoint(18, datos[18]),
        new DataPoint(19, datos[19])
    });
    series.setColor(Color.CYAN);
    Gp.addSeries(series);
  }
```

Función para dibujar la gráfica a partir de los datos obtenidos del micrófono y poderla dibujar al apretar un botón

```
  public void graficaNormal(View view) {
    limpiar();
    series = new LineGraphSeries<DataPoint>(new DataPoint[]{
        new DataPoint(0, datos[0]),
        new DataPoint(1, datos[1]),
        new DataPoint(2, datos[2]),
        new DataPoint(3, datos[3]),
        new DataPoint(4, datos[4]),
        new DataPoint(5, datos[5]),
        new DataPoint(6, datos[6]),
        new DataPoint(7, datos[7]),
        new DataPoint(8, datos[8]),
        new DataPoint(9, datos[9]),
        new DataPoint(10, datos[10]),
        new DataPoint(11, datos[11]),
```

```
            new DataPoint(12, datos[12]),
            new DataPoint(13, datos[13]),
            new DataPoint(14, datos[14]),
            new DataPoint(15, datos[15]),
            new DataPoint(16, datos[16]),
            new DataPoint(17, datos[17]),
            new DataPoint(18, datos[18]),
            new DataPoint(19, datos[19])
    });
    series.setColor(Color.CYAN);
    Gp.addSeries(series);
}
```

Función que dibuja la gráfica de mínimos cuadrados

```
public void graficaMinimosC(View view) {
    limpiar();

    for (int i = 0; i < datos.length; i++) {
        x += i;
        y += datos[i];
        xy += (i * datos[i]);
        x2 = Math.pow(i, 2);
    }
    //calculo de la pendiente m
    m = (xy - ((x * y) / datos.length)) / (x2 - ((Math.pow(x, 2)) / datos.length));

    //calculo de la intercepcion en y
    //media de x
    xm = x / datos.length;
    //media de y
    ym = y / datos.length;

    //intercepcion
    yi = ym - (m * xm);

    series = new LineGraphSeries<DataPoint>(new DataPoint[]{
            new DataPoint(0, (m * 0) + yi),
            new DataPoint(1, (m * 1) + yi),
            new DataPoint(2, (m * 2) + yi),
            new DataPoint(3, (m * 3) + yi),
            new DataPoint(4, (m * 4) + yi),
            new DataPoint(5, (m * 5) + yi),
            new DataPoint(6, (m * 6) + yi),
            new DataPoint(7, (m * 7) + yi),
            new DataPoint(8, (m * 8) + yi),
            new DataPoint(9, (m * 9) + yi),
            new DataPoint(10, (m * 10) + yi),
            new DataPoint(11, (m * 11) + yi),
            new DataPoint(12, (m * 12) + yi),
            new DataPoint(13, (m * 13) + yi),
            new DataPoint(14, (m * 14) + yi),
            new DataPoint(15, (m * 15) + yi),
            new DataPoint(16, (m * 16) + yi),
            new DataPoint(17, (m * 17) + yi),
            new DataPoint(18, (m * 18) + yi),
            new DataPoint(19, (m * 19) + yi)
```

```
    });
    series.setColor(Color.rgb(249, 91, 91));
    Gp.addSeries(series);

  }

  /**
   * funcion para limpiar las variables usadas par amostrar los datos
   */
  public void limpiar() {
    y = 0;
    x = 0;
    xy = 0;
    x2 = 0;
    Gp.removeAllSeries();
  }
```

Función para mostrar ambas gráficas a la vez

```
  public void graficas(View view) {
    limpiar();
    series = new LineGraphSeries<DataPoint>(new DataPoint[]{
          new DataPoint(0, datos[0]),
          new DataPoint(1, datos[1]),
          new DataPoint(2, datos[2]),
          new DataPoint(3, datos[3]),
          new DataPoint(4, datos[4]),
          new DataPoint(5, datos[5]),
          new DataPoint(6, datos[6]),
          new DataPoint(7, datos[7]),
          new DataPoint(8, datos[8]),
          new DataPoint(9, datos[9]),
          new DataPoint(10, datos[10]),
          new DataPoint(11, datos[11]),
          new DataPoint(12, datos[12]),
          new DataPoint(13, datos[13]),
          new DataPoint(14, datos[14]),
          new DataPoint(15, datos[15]),
          new DataPoint(16, datos[16]),
          new DataPoint(17, datos[17]),
          new DataPoint(18, datos[18]),
          new DataPoint(19, datos[19])
    });
    series.setColor(Color.CYAN);
    Gp.addSeries(series);

    for (int i = 0; i < datos.length; i++) {
      x += i;
      y += datos[i];
      xy += (i * datos[i]);
      x2 = Math.pow(i, 2);
    }
    //calculo de la pendiente m
    m = (xy - ((x * y) / datos.length)) / (x2 - ((Math.pow(x, 2)) / datos.length));

    //calculo de la intercepcion en y
```

```
    //media de x
    xm = x / datos.length;
    //media de y
    ym = y / datos.length;

    //intercepcion
    yi = ym - (m * xm);

    series = new LineGraphSeries<DataPoint>(new DataPoint[]{
            new DataPoint(0, (m * 0) + yi),
            new DataPoint(1, (m * 1) + yi),
            new DataPoint(2, (m * 2) + yi),
            new DataPoint(3, (m * 3) + yi),
            new DataPoint(4, (m * 4) + yi),
            new DataPoint(5, (m * 5) + yi),
            new DataPoint(6, (m * 6) + yi),
            new DataPoint(7, (m * 7) + yi),
            new DataPoint(8, (m * 8) + yi),
            new DataPoint(9, (m * 9) + yi),
            new DataPoint(10, (m * 10) + yi),
            new DataPoint(11, (m * 11) + yi),
            new DataPoint(12, (m * 12) + yi),
            new DataPoint(13, (m * 13) + yi),
            new DataPoint(14, (m * 14) + yi),
            new DataPoint(15, (m * 15) + yi),
            new DataPoint(16, (m * 16) + yi),
            new DataPoint(17, (m * 17) + yi),
            new DataPoint(18, (m * 18) + yi),
            new DataPoint(19, (m * 19) + yi)
    });
    series.setColor(Color.rgb(249, 91, 91));
    Gp.addSeries(series);
}

/**
 * método para pasar a la actividad principal
 *
 * @param view parametro para que reciba el evento del click del boton.
 */
public void regresa(View view) {
    Intent i = new Intent(this, DbIndicatorActivity.class);
    startActivity(i);
}

/**
 * método para guardar datos
 */
public void guardaBD(View view) {
    CallAPI e = new CallAPI();
    e.execute();
    BGuarda.setEnabled(false);
}
```

APÉNDICE B1

Código de Android Studio parte de la interfaz desde xml.

Desde el main.xml se mana a llamar al string.xml para que se puedan visualizar las comparaciones

```
</RelativeLayout>

    <LinearLayout
      android:id="@+id/Ruido"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:orientation="vertical"
      tools:ignore="RtlHardcoded">

      <TextView
        android:id="@+id/Ruido9"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:text="@string/Ruido9"
        tools:ignore="LabelFor" />

      <TextView
        android:id="@+id/Ruido8"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:text="@string/Ruido8"
        tools:ignore="LabelFor" />

      <TextView
        android:id="@+id/Ruido7"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:text="@string/Ruido7"
        tools:ignore="LabelFor" />

      <TextView
        android:id="@+id/Ruido6"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:text="@string/Ruido6"
        tools:ignore="LabelFor" />

      <TextView
        android:id="@+id/Ruido5"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:text="@string/Ruido5"
        tools:ignore="LabelFor" />
```

```
    <TextView
      android:id="@+id/Ruido4"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:ems="10"
      android:text="@string/Ruido4"
      tools:ignore="LabelFor" />

    <TextView
      android:id="@+id/Ruido3"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:ems="10"
      android:text="@string/Ruido3"
      tools:ignore="LabelFor" />

    <TextView
      android:id="@+id/Ruido2"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:ems="10"
      android:text="@string/Ruido2"
      tools:ignore="LabelFor" />

    <TextView
      android:id="@+id/Ruido1"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:ems="10"
      android:text="@string/Ruido1"
      tools:ignore="LabelFor" />


    <com.jjoe64.graphview.GraphView
      android:id="@+id/graph"
      android:layout_width="match_parent"
      android:layout_height="200dip" />

  </LinearLayout>
```

Parte que muestra los botones de la segunda interfaz en xml

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:orientation="vertical">


    <EditText
      android:id="@+id/Tamaño"
      android:layout_width="250dp"
      android:layout_height="200dp"
      android:layout_marginBottom="30dp"
      android:layout_marginTop="5dp"
      android:gravity="top"
      android:inputType="textMultiLine"
```

```
    android:scrollbars="vertical"
    android:text="@string/tamano"
    android:textSize="30sp" />

<com.jjoe64.graphview.GraphView
    android:id="@+id/Vizualizador"
    android:layout_width="match_parent"
    android:layout_height="400dp"
    android:layout_marginBottom="30dp" />

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical">

    <Button
        android:id="@+id/ExitMenu"
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:onClick="regresa"
        android:text="@string/Return"
        tools:ignore="ButtonStyle" />

    <Button
        android:id="@+id/GraficaN"
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:onClick="graficaNormal"
        android:text="@string/GraficaN"
        tools:ignore="ButtonStyle" />

    <Button
        android:id="@+id/GraficaL"
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:onClick="graficaMinimosC"
        android:text="@string/GraficaL"
        tools:ignore="ButtonStyle" />

    <Button
        android:id="@+id/GraficaAmbas"
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:onClick="graficas"
        android:text="@string/GraficaA" />

    <Button
        android:id="@+id/Guarda"
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:onClick="guardaBD"
        android:text="@string/GuardaBD" />
```

```
</LinearLayout>
```

APÉNDICE C

Variable de la clase que controla el micrófono

```
private MediaRecorder mRecorder = null;
```

Barra de progreso para saber cuándo terminó de grabar los datos

```
private ProgressBar Barra_Tiempo;

//Variable del escuchador (EAR)
private Ear escucha = null;

//variable para indicar cuando debe de grabar los datos
private boolean grabar = false;


//arreglo para guardar los datos
private double x[] = new double[20];
private int contador = 0;

//variables para la grafica
private LineGraphSeries<DataPoint> series;
private int lastX = 0;
private GraphView graph;
```

Clase de inicio de actividad, instancia todos los objetos de la ventana en el back para poder manipularlos

```
/**
 * Called when the activity is first created.
 **/
@Override
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.main);
```

Obtener los elementos de la ventana por medio del id

```
    GrabarDatos = (Button) findViewById(R.id.GrabarDatos);
```

Obtenemos el elemento de gráfica

```
    graph = (GraphView) findViewById(R.id.graph);
```

Método que inicializa la escucha

```
public void start() {

    //Inicializamos los parametros del grabador
    mRecorder = new MediaRecorder();
```

```
        mRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
        mRecorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
        //No indicamos ningún archivo ya que solo queremos escuchar
        mRecorder.setOutputFile("/dev/null");
        mRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
        try {
           mRecorder.prepare();
        } catch (IllegalStateException e) {
           Log.e("error", "IllegalStateException");
        } catch (IOException e) {
           Log.e("error", "IOException");
        }
        mRecorder.start();
    }

    //Para la escucha
    public void stop() {

      if (mRecorder != null) {
        mRecorder.stop();
        mRecorder.release();
        mRecorder = null;
      }
      escucha = null;
    }

    public void grabaDatos(View view) {
      GrabarDatos.setEnabled(false);
      Para.setEnabled(false);
      Barra_Tiempo.setVisibility(View.VISIBLE);
      grabar = true;
    }
```

Función que ejecuta el botón de comenzar para que empiece a obtener datos del micrófono

```
    public void comienzaEstado(View view){
      escucha = new Ear();
      escucha.setListening(true);
      start();
      escucha.execute();
```

Detiene la escucha del micrófono

```
    public void paraEstado(View view){
      escucha.setListening(false);
      escucha.Reset();
      stop();
      escucha = null;
```

Función que pasa a la actividad donde se ve la gráfica con los 20 datos obtenidos y pasa esos 20 datos para ser manipulados

```
    public void ver_Grafica(View view) {
```

Detiene la escucha de micrófono

```
    if (escucha != null) {
       escucha.setListening(false);
       escucha.Reset();
       stop();
       escucha = null;
    }
```

Manda los 20 datos al bundle para ser usados por la otra actividad

```
    Bundle mBundle = new Bundle();
    mBundle.putDoubleArray("DatosG", x);
```

Inicia la siguiente actividad

```
    Intent i = new Intent(this, Grafica.class);
    i.putExtras(mBundle);
    VerGrafica.setEnabled(false);
    startActivity(i);
  }


  //Devuelve la mayor amplitud del sonido captado desde la última vez que se llamó al método
  public double getAmplitude() {
    if (mRecorder != null)
       return (mRecorder.getMaxAmplitude());
    else
       return 0;
  }

  public class Ear extends AsyncTask<Void, Double, Void> {

    private boolean listening;

    @Override
    protected Void doInBackground(Void... arg0) {

       while (isListening()) {
```

Se da 500 milisegundos para evitar errores de lectura

```
          SystemClock.sleep(500); // Si es menor casi siempre da 0
```

Obtiene la amplitud para tener un valor

```
          Double amplitude = 20 * Math.log10(getAmplitude() / 32768.0);
          publishProgress(amplitude);
       }

       return null;
    }

    @Override
    protected void onProgressUpdate(Double... values) {
       Double value = values[0];
```

Valores aleatorios para corregir la lectura de los valores

```
        Random r = new Random();
        double randomValue = .15 + (.20 - .15) * r.nextDouble();
        double randomValue2 = .05 + (.10 - .05) * r.nextDouble();
```

Corrección de las medidas cuando superan el límite de decibeles

```
        if (value < -80) {
           value = new Double(-80);
        } else if (value > 0) {
           value = new Double(0);
        }
        Double DBBarra=value-(value*randomValue)-(80*randomValue2);
        value=((value+80)*1.299)-(value*randomValue)-(130*randomValue2);
```

Función para grabar los 20 datos que se mandaran a la otra actividad

```
        if (grabar != false) {
           if (contador < 20) {
              x[contador] = value;
              contador++;
              Barra_Tiempo.setProgress(contador);
           } else {
              //guardar datos en la base de datos
              contador = 0;
              grabar = false;
              Para.setEnabled(true);
              GrabarDatos.setEnabled(true);
              VerGrafica.setEnabled(true);
              Barra_Tiempo.setVisibility(View.INVISIBLE);
              Barra_Tiempo.setProgress(contador);
           }
        }

        String db = new Formatter().format("%03.2f", value).toString();

        decibelsTx.setText(db + " dB");

        updateBar(DBBarra,value);

     }

     public void updateBar(Double db, Double Graph) {
        Double width;

        series.appendData(new DataPoint(lastX++, Graph), true, 10);
        graph.addSeries(series);

        // Factor de escala para convertir a dB
        final float scale = getResources().getDisplayMetrics().density;

        width = (db * 250 * scale) / -80; // Anchura en pixeles


        RelativeLayout.LayoutParams lyParams = new
RelativeLayout.LayoutParams(width.intValue(), barDB.getHeight());
        lyParams.addRule(RelativeLayout.ALIGN_PARENT_RIGHT);
```

```
        barDB.setLayoutParams(lyParams);

      }

      public void Reset() {
        Double width;

        final float scale = getResources().getDisplayMetrics().density;

        width = (1.0 * scale * 250);

        RelativeLayout.LayoutParams lyParams = new
RelativeLayout.LayoutParams(width.intValue(), barDB.getHeight());
        lyParams.addRule(RelativeLayout.ALIGN_PARENT_RIGHT);
        barDB.setLayoutParams(lyParams);

        decibelsTx.setText("");
      }

      public boolean isListening() {
        return listening;
      }

      public void setListening(boolean listening) {
        this.listening = listening;
      }
    }

    @Override
    public void onProgressChanged(SeekBar seekBar, int progress,
                        boolean fromUser) {
      decibeliosSeekbar.setText((80 + progress) + " dB");
    }

    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {
      // TODO Auto-generated method stub

    }

    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {
      // TODO Auto-generated method stub

    }

}
```

Función para pasar los valores a un texto y poder mostrarlo en un elemento en pantalla

```
    public void setDatos() {
      for (int i = 0; i < datos.length; i++) {
        muestradatos = muestradatos + "Dato numero " + (i + 1) + ": " + new
Formatter().format("%03.2f", datos[i]).toString() + "\n";
      }
    }
```

```
/**
 * método para guardar datos
 */
public void guardaBD(View view) {
   CallAPI e = new CallAPI();
   e.execute();
   BGuarda.setEnabled(false);
}
```

Para mandar los 20 datos de forma sencilla, se mandan como si fueran un texto separados por una “,” así separarlos por esa “,” en el lado de php

```
public String formaCadena() {
   String x = "";
   for (int i = 0; i < datos.length; i++) {
      x += Double.toString(datos[i]);
      if (i != datos.length - 1) {
         x += ",";
      }
   }
   return x;
}
```

Calcula el promedio

```
public String Promedio() {
   Double prom = 0.0;
   for (int i = 0; i < datos.length; i++) {
      prom += datos[i];
   }
   prom = prom / datos.length;
   return Double.toString(prom);
}
```

Clase para mandar los datos a la página web

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

   public CallAPI() {
      //set context variables if required
   }

   @Override
   protected void onPreExecute() {
      super.onPreExecute();
   }

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

      String resultToDisplay = "";
      // Create a new HttpClient and Post Header
```

Crea un cliente http para mandar los datos por medio del método post

```
      HttpClient httpclient = new DefaultHttpClient();
```

```
        HttpPost httppost = new
```
Dirección del página web `HttpPost("http://192.168.0.102/decibelio/SetData.php");`
```
        String promedio = Promedio();
        String datos = formaCadena();

        try {
          // agregamos los valores que queremos mandar
          List<NameValuePair> nameValuePairs = new ArrayList<NameValuePair>(2);
          nameValuePairs.add(new BasicNameValuePair("Promedio", promedio));
          nameValuePairs.add(new BasicNameValuePair("Datos", datos));
          httppost.setEntity(new UrlEncodedFormEntity(nameValuePairs));

          // Execute HTTP Post Request
```
Obtenemos una respuesta del servidor
```
          HttpResponse response = httpclient.execute(httppost);

        } catch (ClientProtocolException e) {
          // TODO Auto-generated catch block
        } catch (IOException e) {
          // TODO Auto-generated catch block
        }
        return resultToDisplay;
    }

    @Override
    protected void onPostExecute(String result) {
      //Update the UI
    }
  }
}
```