2017

APLICACIÓN MÓVIL PARA LA IDENTIFICACIÓN DE LA AVIFAUNA, TOMANDO COMO CRITERIO TAXONÓMICO EL CANTO.

PRESENTADO POR:
DAVID ACOSTA PEÑA ESTUDIANTE DE INGENIERÍA ELECTRÓNICA
BRAYAN HERNANDEZ R. ESTUDIANTE DE INGENIERÍA ELECTRÓNICA
DIRECTOR: ING. LUIS L. CAMARGO ARIZA

CONTENIDO

RESUMEN

En el estudio de las aves, el canto como criterio taxonómico para identificación de aves se podría considerar tan importante y confiable como los criterios moleculares. Cada especie o subespecie dentro de una familia produce un canto particularmente diferente a la de las demás especies. La Sierra Nevada de Santa Marta posee muchas aves endemicas de Latinoamérica y por ende posee especies que no se pueden encontrar en otro lugar, por lo que atrae a muchas personas de carácter investigador o aficionados a conocer la ornitofauna presente en ella. El proyecto consiste en desarrollar una aplicación útil para el reconocimiento de las distintas especies de aves a través de su canto característico, esta aplicación va dirigida a todo público ya sea profesional o aficionado, es decir, es una aplicación general. Además con esto se pretende fomentar el ecoturismo y brindar un apoyo tecnológico de fácil acceso a los distintos guías turísticos, y al mismo tiempo contribuirá crear una conciencia de preservación de las distintas especies de aves.

Este proyecto es importante para la preservación de las especies de aves que existen en la Sierra Nevada de Santa Marta reconociendo a través de la aplicación móvil la diversidad de cantos de las aves que habitan el ecosistema. De igual forma es un medio para el inventario de aves que existenten ya sea que se encuentren en vía de extinción o no, preservando en el tiempo su sonoridad expresada en el canto que los identifican como aves.

CAPÍTULO 1.

IDENTIFICACIÓN

1.1TÍTULO

Aplicación móvil para la identificación de la avifauna, tomando como criterio taxonómico el Canto.

1.2. PLANTEMAIENTO DEL PROBLEMA.

En Colombia el avistamiento de aves tiene un gran atractivo debido a que es el primer país del mundo en cuanto a diversidad de aves de 1900 especies [1], lo cual se puede aprovechar mucho desde el punto de vista ecoturistico, investigativo y de conservación; es por ello que las herramientas tecnológicas son importantes en tanto facilitan el acceso a este tipo de conocimiento, potenciando el avistamiento de aves como pasatiempo.

La Sierra Nevada de Santa Marta posee gran variedad de animales salvajes, consistente en especies de aves, mamíferos, peces e insectos. Se han detectado unas 628 especies de aves en los diferentes ecosistemas de la eco región, que significa 35 % de las especies de aves de Colombia en 1,48 % de la superficie del país. [2] Para la Sierra Nevada de Santa Marta en los límites con el magdalena se han hecho listados de varias localidades tales como: Minca-pozo azul (600-700msnm), la Y (1550msnm), reserva el Dorado (2000msnm), San Lorenzo Ridge (2200msnm). [3].

Orniturismo, turismo de observación de aves o aviturismo, es la actividad que implica desplazarse desde un sitio de origen hacia un destino específico con el interés de observar la avifauna local en su entorno natural. Usualmente deja un incentivo económico para el destino, la comunidad receptora y los guías locales.

Muchas de las personas que van a avistar aves por diversión no tienen los conocimientos necesarios en ornitología suficientes para lograr identificar aves por su canto o por su morfología, por lo que recurren a personas capacitadas en este campo o en guias especializado lo cual genera un aumento en los costos a la actividad de orniturismo.

Basados en lo anterior y a la no existencia de una aplicación móvil desarrollada capaz de identificar las aves por medio de su canto y mostrar una reseña del ave que se ha escuchado cuando sin embargo existen aplicación capaces de ayudar a las personas a encontrar aves, es decir lugares geográficos que ya se han investariado como sitios de avistamiento, tales como BirdLife; esta aplicación informa sobre la vida, el comportamiento, la distribución y el estado de conservación de las 563 especies de aves que, de forma habitual u ocasional, están presentes o han sido citadas en el territorio español. También describe rutas por 25 enclaves de gran importancia ornitológica. Los textos se acompañan de numerosos recursos: cantos, vídeos, fotografías, ilustraciones y mapas. Con este contenido la obra se convierte en la más completa guía especializada en ornitología realizada hasta ahora para dispositivos móviles en España [4].

¿Cómo identificar las aves que se hallan en el ecosistema natural por su canto característico a través de una aplicación móvil?

¿Qué método permite identificar las aves por su canto en el hábitat natural sin ser conocedor de las características de las especies biológicas?

1.3. JUSTIFICACIÓN

Los dispositivos móviles inteligentes o como popularmente se les conoce Smartphone, actualmente son dispositivos a lo que muchas personas tienen acceso como lo muestra el estudio de penetración de Smartphone en América Latina entre 2013 y 2018 realizado por eMarketer donde muestra que en Colombia para el año 2014 contaba con 14.4 millones de estos dispositivos con un crecimiento respecto al año anterior de 23 porciento y con una penetración comparable con la de los países desarrollados del 106 por ciento [5], de la mano a esta tecnología también fue creciendo el mundo de las aplicaciones móviles [6]. Entre estas aplicaciones encontramos las grabadoras de voz las cuales son fundamentales si del estudio de canto de aves se habla.

La bio-acústica en el estudio de las aves es relevante debido a que por el canto se puede determinar el comportamiento del ave [7]. Las grabaciones constituyen una herramienta importante en el estudio, manejo y conservación de las especies. La precisión de la grabación define la calidad de la misma. Por tanto el sistema digital de grabación en nuestro caso servirá para enriquecer nuestra base de datos, y nos acercara a dicha precisión, Parker, (1991). Valiéndonos de la calidad del Smartphone con el que cuenta el investigador o aficionado. Autores citan en sus trabajos la importancia de las vocalizaciones de las aves y como grabarlas, como es el caso de Vielliard [7], además señalan su preocupación por la pérdida del hábitat, explicando que dicha perturbación es más rápida que los trabajos científicos.

Teniendo en cuenta que algunas personas de las que hacen avistamiento de aves por diversión carecen del conocimiento necesario para lograr identificar las aves o el dinero requerido para pagar a un experto se hace necesario desarrollar una aplicación móvil que facilite el orniturismo; y que esta a su vez le sirva a los investigadores de modo que a largo plazo se podrán realizar estudios de monitoreo de las aves residentes y migratorias no solo de la Sierra Nevada de Santa Marta, sino a nivel nacional en los diferentes biomas de la franja tropical; sino en localidades internacionales. Así se podrá brindar una herramienta, no sólo como atractivo para el ecoturismo y desarrollo de las regiones como actividad principal, sino el monitoreo de aves amenazadas, vulnerables o endémicas, a medida que se enriquezca la base de datos de la aplicación.

1.4. OBJETIVOS

1.4.1. Objetivo general.

Desarrollar una Aplicación móvil para la identificación de la avifauna tomando como criterio taxonómico el Canto.

1.4.2. Objetivos Específicos

- Identificar los cantos de la avifauna presente en el tramo Minca-Pozo Azul a partir de las bases de datos.

- Establecer las bases de datos de los espectros de los cantos de las aves presentes en el tramo Minca Pozo Azul y recolectar datos específicos sobre estas como su nombre, lugar de avistamiento y si se encuentra en peligro de extinción o no.

- Desarrollar una aplicación que capture y analice el audio del canto del ave.

- Desarrollar una aplicación móvil donde el usuario pueda ver una imagen del ave que canto, una descripción de la misma y análisis del audio del canto.

- Realizar pruebas de laboratorio en un entorno controlado para verificar que tan eficiente es la aplicación a la hora de identificar las aves para retroalimentar los procesos de funcionamiento de la aplicación, para luego hacer la prueba en el entorno real.

1.5 ALCANCES Y LIMITACIONES

Dentro del desarrollo del proyecto se encontraran las siguientes: 1) la limitación se encuentra dentro de la cantidad de aves presentes dentro de la base de datos debido a que esto no permite a la aplicación identificar cualquier ave que te encuentres mientras se camina por cualquier zona geográfica de Colombia por poner un ejemplo. 2) Otra limitación es dado el caso donde dos especies de aves se encuentran una cerca de la otra cantando, para el aplicativo va a ser muy difícil reconocer exactamente que especie es la que se encuentra cantando. 3) existen varias especies de aves que mimetizan a otras, logrando de este modo engañar al aplicativo generando una identificación falsa.

La base de datos con la que se trabajará en este proyecto solo incluye las aves presentes en tramo Minca Pozo Azul, Siendo este nuestro único sector de muestreo, se puede decir que si se amplia la base de datos a un número mayor de aves esta aplicación puede ser utilizada alrededor del mundo por las distintas personas que de una u otra manera quieran ver aves por diversión o para investigación, adicionalmente se aspira que este proyecto de grado sirva como base para el desarrollo de nuevas investigaciones que permitan la integración de tecnologías como las aplicaciones móviles para facilitar el estudio y observación de la biodiversidad que se encuentra en el planeta entero y generar conciencia sobre la importancia que tiene la conservación de las epecies.

CAPÍTULO 2. MARCO TEÓRICO

2.1. ANTECEDENTES

Con el auge de la tecnología de los teléfonos inteligentes la cual se está aprovechando para investigar mucho en todos los campos de las ciencias como es el caso de la ornitología donde se puede encontrar que científicos de la Queen Mary University of London están desarrollando una aplicación llamada "Warblr" la cual se dice será lanzada en la primera mitad del presente año 2015 para iOS y Android con un costo de 3 dólares, la cual contará en primera instancia con una base de datos de 80 especies de Reino Unido aunque todavía se encuentra a prueba sus desarrolladores aseguran que tiene una efectividad identificando especies de 95% [8].

En Colombia en la Universidad Nacional sede Manizales el grupo de procesamiento y reconocimiento de señales teniendo en cuenta el proyecto ARBIMON creado en la Universidad de Puerto Rico y con colaboración de profesionales en electrónica y biología se desarrolló un software que busca simular el razonamiento humano que consiste en comparar dos elementos y mirar sus características generales y de este modo lograr identificarlo de forma satisfactoria como cuando se reconoce a alguien por su rostro. En el estudio realizado en la Reserva de Río Blanco de Manizales hubo un acierto del 97.87% con lo que se demuestra que tiene una gran precisión y según palabras del investigador José Francisco Ruíz Muñoz el error solo se presentó en aves cuyo canto es muy similar [9].

Todos estos proyectos son desarrollados en un entorno completamente conectado, es decir, el sonido es captado por el dispositivo móvil y es analizado en un servidor que se encarga de comparar con la base de datos para saber que especie representa este sonido, haciendo todo el análisis para conocer por completo el espectro y de este modo saber todas las minucias que componen a ese sonido.

En el caso de la aplicación móvil "Warblr" de la Queen Mary University fue desarrollada para iOS el cual a pesar de ser un sistema operativo difundido en el mundo no es el que se encuentra más accesible al público en general debido al alto costo de los dispositivos que utilizan este sistema operativo.

Lo que se ha desarrollado es muy similar a lo que en este proyecto se presentará, pero el valor agregado que se le va a dar en este proyecto es que teniendo en cuenta que en el lugar donde se va a aplicar la disponibilidad de cobertura de señal móvil no es muy eficiente por lo que se va a trabajar en un entorno semi desconectado, es decir, cuando no se encuentre cobertura de la señal este se va a hacer todo el procesamiento de la señal con los elementos procedentes dentro del móvil, para ser más exacto con su procesador y la memoria RAM. Cuando este tenga cobertura de señal móvil y acceso a internet, todos y cada uno de los procesos serán llevados a cabo en un servidor central, en caso de existir una actualización en la base de datos, esta se va a actualizar.

Además de trabajar en un entorno semiconectado, cada usuario tendrá una cuenta que registrará todos y cada uno de las aves que este ha avistado y la posición geográfica de donde lo hizo, lo cual podrá compartir con el resto de personas que utilizan esta aplicación y de este modo lograr un entorno más interactivo.

El sistema operativo para el que se desarrollara esta aplicación va a ser un sistema operativo muy masificado y que pertenece al software de licencia abierta cuyo nombre es Android y fue desarrollado por Google INC. La principal ventaja de trabajar con este tipo de software, es que no hay necesidad de comprar una licencia lo que reduce los

costos de desarrollo de la misma y por lo tanto lo hace más accesible al público en general, lo que ayuda a masificarla y poder llegar a más personas en el mundo.

2.2. TAXONOMÍA DEL CANTO DE LAS AVES

2.2.3 CLASIFICACIÓN DE LAS AVES.

La clasificación de las aves es una temática muy compleja y discutible. La mayoría de los biólogos acuerdan que existen alrededor de 9.700 especies de aves en total, que pertenecen al orden Aves. Pero exactamente como se relacionan entre sí es todavía un debate abierto. Durante siglos los científicos usaron las características físicas internas y externas para clasificarlas, agrupándolas de acuerdo a su estructura esquelética, la forma del pico, el tamaño, color y otros rasgos visibles [10].

En cuanto a su clasificación, Las aves se agrupan en dos súper órdenes:

Paleognathae (mandíbulas antiguas), que son las aves no voladoras, y Neognatae (mandíbulas nuevas), que son las aves voladoras.

Las aves no voladoras (Paleognathae), están agrupadas en dos órdenes, seis familias y 55 especies. Estas son aves terrestres, de gran tamaño, que se caracterizan porque no tienen quilla en el esternón, lugar donde se insertan los músculos de las alas que permiten el vuelo. Las Tinamiformes, que habitan en Colombia, son una excepción, son las únicas aves de este súper orden que vuelan. El tinamú representa un enigma para la ciencia, es curioso que no haya perdido la capacidad de volar, como ocurrió con el resto de las aves terrestres que son igual de primitivas.

Las aves voladoras (Neognathae), se agrupan en 20 órdenes, representados por más de 132 familias. Dentro de este súper orden, las más numerosas son las Passeriformes con más de 40 familias y 5.000 especies. Las aves más antiguas de este grupo son las Anseriformes y las Galliformes.

A continuación solo se presentan los órdenes y familias de aves, que por su tipo de alimentación, frugívoros, nectarívoros, insectívoros y carnívoros, pueden afectar como plagas a la producción agroforestal, o por el contrario beneficiar este tipo de agro ecosistemas por el efecto de los polinizadores, por el control de las poblaciones de insectos o por la depredación de roedores u otros mamíferos dañinos a la agricultura [11].

Orden Apodiformes (aves con patas son muy pequeñas)
- Apodidae: Como los vencejos que son en su mayoría insectívoros.

- Trochilidae: Es la familia de los colibríes que cumplen con la polinización de muchas plantas agrícolas Orden Caprimulgiformes (aves crepusculares insectívoras).

- Caprimulgidae: chotacabras, guardacaminos.

- Nyctibiidae: biemparado

- Steatornithidae: guácharo

Orden Columbiformes (Es el grupo de las palomas, que son granívoras y pueden afectar los cultivos de cereales)
- Columbidae: caminera, paloma, paloma perdiz, torcaza, tórtola, tortolita

- Pteroclididae: ganga

Orden Falconiformes (aves rapaces, de rapiña, carroñeras diurnas, depredadoras de posibles mamíferos plagas de la agricultura)
- Accipitridae: águila, aguililla, aguilucho, azor, caracolero, gavilán, gavilancito

- Cathartidae: cóndor, gallinazo, guala, rey de los gallinazos

- Falconidae: cacao, caracara, cernícalo, esmerejón, halcón, halcón montés, pigua

- Pandionidae: águila pescadora

- Sagittariidae: serpentario

Orden Passeriformes (aves cantoras, muchas frugívoras e insectívoras) - Alaudidae: alondra

- Bombycillidae: ampelis

- Cardinalidae: arrocero, azulillo, azulón, picogordo, saltador

- Certhiidae: agateador

- Cinclidae: mirlo acuático

- Conopophagidae: zumbador

- Cotingidae: campanero, cotinga, cuaba, frutero, gallito de roca, guardabosques, pájaro capuchino, toropisco

- Corvidae: carriquí, corneja, cuervo, grajilla, urraca

- Donacobiidae: cucarachero de laguna

- Emberizidae: arrocero, canario, cardenal, cardonero, espiguero, gorrión, gorrión montés, pinzón, sabanero, semillero

- Estrilididae: munia, copetón de Java

- Formicariidae: gallito, tororoi

- Fringillidae: clorofonia, eufonia, jilguero

- Furnariidae: canastero, castillero, chamicero, cínclodes, coludito, corretroncos,guadañero, hojarasquero, hornero, moñudo, musguero, picodegancho, raspahojas, saltarrocas, trapecista, trepatroncos, xenops

- Hirundinidae: golondrina

- Icteridae: arrendajo, chamón, chirlobirlo, mariamulata, oropéndola, soldadito, tordo, tordo arrocero, turpial

- Laniidae: alcaudón

- Menuridae: ave lira

- Mimidae: pájaro gato, sinsonte

- Motacillidae: bisbita

- Muscicapidae: mosquitero, ruiseñor

- Oriolidae: oropéndola del Viejo Mundo

- Oxyruncidae: picoagudo

- Paradisaeidae: ave del paraíso

- Paridae: carbonero

- Parulidae: abanico, arañero, candelita, cebrita, reinita

- Passeridae: copetón

- Pipridae: saltarín

- Pittidae: pita

- Ploceidae: tejedor

- Polioptilidae: curruca

- Pycnonotidae: bulbul

- Regulidae: reyezuelo

- Rhinocryptidae: tapaculo

- Sapayoidae: saltarín

- Sittidae: trepador

- Sturnidae: estornino

- Sylviidae: trino

- Thamnophilidae: batará, hormiguerito, hormiguero

- Thraupidae: azulejo, buscaquiches, chambergo, chococito, clorornis, conirrostro, dacnis, gorrión afelpado, güicha, hemispingus, lanio, mielero, montero, musguerito, parlotero, picaflor, pintasilgo, pizarrita, pollodemonte, rosita, semillero, tángara, toche, trinadora, viuva, zarcerito

- Tityridae: cotinga, plañidera, saltarín, titira

- Troglodytae: cucarachero

- Turdidae: mayo, mirla, solitario, zorzal

- Tyrannidae: atila, atrapamoscas, bichofué, cachudito, doradito, dormilona, elenia, spatulilla, monjita, pibí, picochato, picoplano, picodepala, piquitorcido, pitajo, plañidera, sisirí, suelda, tachuri, tijereta, tiranuelo, titira, titiribí, viudita

Orden Piciformes (ave insectívoras principalmente)
- Bucconidae: bigotudo, bobo, monja, monjita

- Picidae: carpinterito, carpintero, chupasavia

- Ramphastidae: pichí, pichilingo, terlaque, tucán, tucancito

Orden Strigiformes (aves rapaces nocturnas) - Strigidae: autillo, buhíto, búho, currucutú, mochuelo - Tytonidae: lechuza [12].

2.2.4 EL CANTO DE LAS AVES

Este es un término empleado para designar una amplia variedad de manifestaciones vocales de las aves, que sirven para transmitir informaciones relacionadas con el apareamiento. Unas pocas especies son mudas (por ejemplo, las cigüeñas y ciertos pelícanos y buitres), pero la mayoría emiten sonidos de uno y otro tipo.

Particularmente en un grupo, el de los pájaros cantores o canoros (suborden oscines del orden pasariformes), las vocalizaciones están ordenadas para formar una secuencia reconocible de notas, en general de más de un tipo y que guardan cierta relación unas con otras. Esta relación suele ser relativamente fija, y origina un patrón específico de vocalización conocido como “canto” del ave en cuestión.

No siempre es fácil distinguir entre el “canto” y la “llamada” de un ave. La nota de llamada suele ser única (a veces se trata de una misma nota repetida), como en el caso de las llamadas de vuelo de los pinzones o las de comida de los paros, pero ciertas vocalizaciones que funcionalmente se incluyen en la categoría de cantos son poco más o menos que notas de llamada repetidas. Como ejemplos cabe citar el canto de ciertos chorlitos y palomas. De hecho, tales aves son incapaces de emitir un sonido similar al de las verdaderas canoras, ya que solamente éstas tienen siringe, órgano vocal de complejidad suficiente para producir sonidos diversos y variables. Así, pues, el canto de las no oscines ha de limitarse a simples arreglos y modificaciones de las notas de llamada.

En los verdaderos pájaros canoros, una especie determinada puede exhibir una amplia gama de vocalizaciones, desde simples notas de llamada hasta un canto elaborado, con patrones de notas de llamada, relativamente sencillos, en medio. Al principio de la estación de cría, antes de haberse desarrollado totalmente el canto, éste, sólo se compone, a veces, de unas notas simples. Sin embargo, el canto verdadero suele ser fácilmente reconocible por su complejidad musical.

La principal diferencia entre el canto y otros tipos de sonidos emitidos por las aves se basa en las circunstancias en que se producen. En la mayor parte de los casos, el canto está relacionado con la reproducción. Es una parte importante del comportamiento territorial, ya que la mayoría de los pájaros canoros lo emplean para establecer, delimitar y defender el territorio de cría. Es frecuente que el macho cante situado en un lugar especial, de forma que sus oyentes pueden localizarlo con mayor facilidad. Si, al principio de la estación de cría, un macho recién llegado a una zona oye a otro cantar vigorosa y repetidamente desde el mismo árbol, tendrá la seguridad de que se trata del

propietario de aquel territorio, y, en consecuencia, se irá a otra parte. Por lo tanto, se puede considerar el canto como una forma de evitar las luchas por un territorio. Sólo se producen combates cuando no hay espacio suficiente para todas las parejas en celo, o, en menor grado, en los límites entre unos y otros territorios.

Parece que el canto es también importante para mantener los lazos entre las parejas y para que éstas sincronicen sus actividades durante la época de cría. Como otras funciones relacionadas con la reproducción, está regulado por la secreción de hormonas sexuales en los testículos o en el ovario, y suele considerarse acertadamente como parte importante de la exhibición sexual. Por el contrario, los otros tipos de vocalizaciones se emplean en circunstancias que están poco o nada relacionadas con la actividad sexual. Como ejemplos tenemos las ya citadas llamadas de vuelo o de alimentación, así como las numerosas vocalizaciones que sirven para establecer la comunicación entre padres e hijos. Se trata, pues, de medios encaminados a la supervivencia del individuo y no a la conservación de la especie, que es lo que constituye la finalidad primordial del canto.

Debemos hacer mención de los sonidos instrumentales producidos por las aves, que a no ser por su origen, cabría incluir entre la categoría de Canto, entendido este como una serie de notas, generalmente de más de un tipo, emitidas en sucesión y relacionadas de forma que se puede reconocer una secuencia o pauta en el tiempo. Desde el punto de vista funcional, estos sonidos relacionados con la exhibición sexual, ciertamente pueden considerarse como una forma de canto. Entre ellos se incluye el tamborileo que producen contra el suelo las plumas de la cola de la agachadiza al soplar el viento entre ellas cuando el ave se calla, el que produce el carpintero cuando golpea rápidamente con el pico las ramas u otras estructuras resonantes, el ruidoso aleteo que acompaña el cacareo de los faisanes y el batir de alas de los pichones durante sus vuelos de exhibición.

El canto de algunas especies es mucho más complejo. En el PETIRROJO (*Erithacus rubecula*), por ejemplo, el repertorio contiene más de 1300 motivos diferentes, y hay individuos que emplean varios centenares de ellos. Esta diversidad depende en gran parte de las combinaciones y permutas de los elementos que constituyen cada motivo. Se introducen asimismo cambios de organización de los motivos y parece que el ave es capaz de improvisar hasta cierto punto, sobre el tema de cada motivo.

El petirrojo nos proporciona un ejemplo de ave cuyo canto resulta musical y agradable a cualquier oído humano. Cualquiera que sea la definición de música, la mayoría de las personas consideran musicales los cantos de muchas especies de aves. El MIRLO (*Turdus merula*), el RUISEÑOR (*Luscinia megarhynchos*), el tinamú de América Central, el carnicero policromo de Australia y el solitario de las montañas Rocosas constituyen solo algunos ejemplos de las muchas aves notables por uno u otro aspecto de sus cantos, que resultan muy agradables. La cadencia de las unidades que constituyen el canto, es evidentemente uno de los factores importantes de su efecto, tanto sobre otras aves como sobre los seres humanos. Suele ser única para cada especie: así ocurre con la aceleración que puede apreciarse en el canto del pinzón o del lagópodo escandinavo, la variación observable al comparar las dos notas por segundo del MOSQUITERO común (*Phylloscopus collybita*), las 300 notas por ocho segundos del CHOCHÍN (*Troglodytes troglodytes*), o los cambios de velocidad y variaciones de motivo del ruiseñor o de la curruca capirotada.

Pero en algunas especies intervienen en el canto más de un individuo. En varias especies de horneros (furnáridos) y motmots (momótidos), ambos miembros de la pareja cantan a dúo. Otros grupos aún llegan a más, ya que los dos sexos alternan su contribución al canto conjunto, en lo que demuestran una perfecta sincronización. Así sucede, por ejemplo, en algunos tiranos (tiránidos), chochines (troglodítidos),

alcaudones (lánidos) y barbudos (capitónidos). En muchas de estas especies la sincronización entre las dos aves llega a tal punto que sólo parece cantar un animal. El mimetismo invita también a algunas aves a emplear las vocalizaciones de otras, dentro de las características de su propio canto. Es bien conocida la capacidad imitativa de los loros, los cuervos y los minás, pero existen otras muchas especies que, en condiciones naturales, incorporan en sus cantos las vocalizaciones de aves distintas, e incluso otros sonidos. El estornino pinto, *Sturnus vulgaris*, el sílvido, *Acrocephalus palustris*, el Minus *Polyglottos* y el jardinero capilla, *Chamydera maculata*, constituyen ejemplos sobresalientes de ello. Se ha comunicado el caso de un sinsonte que entonaba parte de los cantos de otras 30 especies por lo menos. Y un estornino que anida en el tejado de la casa del autor de estas líneas imita a más de 22 especies de aves propias de Inglaterra, así como a ovejas, perros y seres humanos. Pero es probable que los mejores imitadores sean las aves del paraíso, las aves lira y los jardineros australianos, que mimetizan no sólo a una amplia variedad de otras aves y mamíferos, sino a gran cantidad de sonidos inanimados, desde la bocina de un automóvil hasta el agua que cae en un cubo. Sin embargo, hasta el momento se desconoce la función de tal desarrollo en la capacidad imitativa.

Además de la posesión de plumas y de las ventajas derivadas de este hecho, es posible que la característica más sobresaliente de las aves, en conjunto, sea el uso de vocalizaciones en el canto. La capacidad de cantar ha llegado en ellas a un grado más elevado que en cualquier otro grupo de animales, con la excepción del hombre, y en la mayor parte de las especies desempeñan un papel fundamentalmente en su existencia [13].

2.3 ANDROID

Android es un sistema operativo en principio pensado para teléfonos móviles, al igual que iOS, Symbian y Blackberry OS, a diferencia que este principio está basado en Linux, un núcleo de sistema operativo libre, gratuito y multiplataforma.

Su lenguaje de programación es una variación de Java llamada Dalvik. Es un sistema operativo que permite la creación de distintas aplicaciones con las cuales se puede manejar todos los periféricos del teléfono móvil basados en un lenguaje de programación sencillo como es Java. Debido a la sencillez y a la cantidad de herramientas de programación gratuitas permite que exista una gran cantidad de aplicaciones que permiten una gran experiencia al usuario [14].

Desarrollo de aplicaciones Android Studio

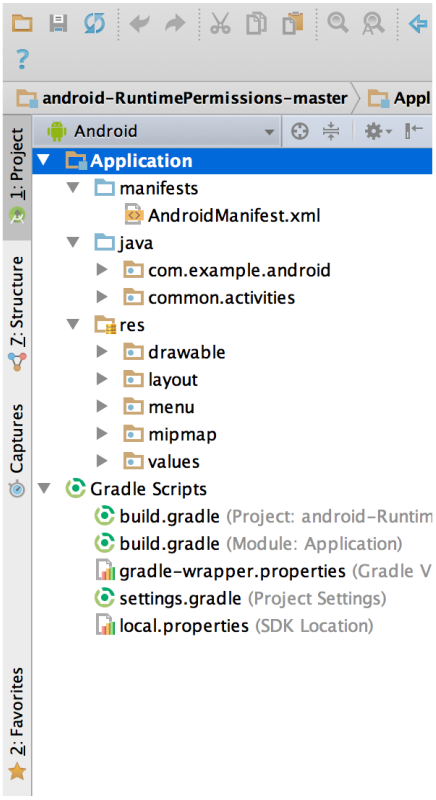
Android Studio es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones para Android y se basa en IntelliJ IDEA. Además del potente editor de códigos y las herramientas para desarrolladores de IntelliJ, Android Studio ofrece aún más funciones que aumentan tu productividad durante la compilación de aplicaciones para Android, como las siguientes:

- Sistema de compilación flexible basado en Gradle.
- Un emulador rápido con varias funciones.
- Un entorno unificado en el que puedes realizar desarrollos para todos los dispositivos Android.
- Instant Run, para aplicar cambios mientras tu app se ejecuta sin la necesidad de compilar un nuevo APK.

- Integración de plantillas de código y GitHub, para ayudarte a compilar funciones comunes de las aplicaciones e importar ejemplos de código.
- Gran cantidad de herramientas y frameworks de prueba.
- Herramientas Lint para detectar problemas de rendimiento, uso, compatibilidad de versión, etc.
- Compatibilidad con C++ y NDK
- Soporte integrado para Google Cloud Platform, que facilita la integración de Google Cloud Messaging y App Engine.

En esta página encontrarás una introducción a las funciones básicas de Android Studio. Para acceder a un resumen de los últimos cambios, consulta Notas de la versión de Android Studio.

Estructura del proyecto



Cada proyecto en Android Studio contiene uno o más módulos con archivos de código fuente y archivos de recursos. Entre los tipos de módulos se incluyen los siguientes:

- Módulos de aplicaciones para Android
- Módulos de bibliotecas
- Módulos de Google App Engine

De forma predeterminada, en Android Studio se muestran los archivos de tu proyecto en la vista de proyectos de Android, como se muestra en la Figura 1. Esta vista está organizada en módulos para que puedas acceder rápidamente a los archivos de origen claves de tu proyecto.

Todos los archivos de compilación son visibles en el nivel superior de **Secuencias de comando de Gradle** y cada módulo de la aplicación contiene las siguientes carpetas:

- **manifiestos**: contiene el archivo AndroidManifest.xml.
- **java**: contiene los archivos de código fuente de Java, incluido el código de prueba JUnit.
- **res**: Contiene todos los recursos, como diseños XML, cadenas de IU e imágenes de mapa de bits.

  La estructura del proyecto para Android en el disco difiere de esta representación plana. Para ver la estructura de archivos real del proyecto, selecciona **Project** en la lista desplegable **Project** (en la figura 1 se muestra como **Android**).

  También puedes personalizar la vista de los archivos del proyecto para concentrarte en aspectos específicos del desarrollo de tu app. Por ejemplo, al seleccionar la vista **Problems** de tu proyecto, aparecerán enlaces a los archivos de origen que contengan errores conocidos de codificación y sintaxis, como una etiqueta de cierre faltante para un elemento XML en un archivo de diseño [15].

2.4 CARACTERÍSTICAS TÉCNICAS DE LOS TELÉFONOS INTELIGENTES.

El teléfono inteligente es un tipo de teléfono móvil construido sobre una plataforma informática móvil, con mayor capacidad de almacenar datos y realizar actividades, semejante a la de una minicomputadora, y con una mayor conectividad que un teléfono móvil convencional. El término «inteligente», que se utiliza con fines comerciales, hacen referencia a la capacidad de usarse como un computador de bolsillo, y llega incluso a reemplazar a una computadora personal en algunos casos.

Generalmente, los teléfonos con pantallas táctiles son los llamados “teléfonos inteligentes”, pero el soporte completo al correo electrónico parece ser una característica indispensable encontrada en todos los modelos existentes y anunciados desde 2007. Casi todos los teléfonos inteligentes también permiten al usuario instalar programas adicionales, habitualmente incluso desde terceros, hecho que dota a estos teléfonos de muchísimas aplicaciones en diferentes terrenos; sin embargo, algunos teléfonos son calificados como inteligentes aun cuando no tienen esa característica.

Entre otros rasgos comunes está la función multitarea, el acceso a Internet vía Wi-Fi o redes 4G, 3G o 2G, función multimedia (cámara y reproductor de videos/mp3), a los programas de agenda, administración de contactos, acelerómetros, GPS y algunos programas de navegación, así como ocasionalmente la habilidad de Y leer documentos de negocios en variedad de formatos como PDF y Microsoft Office [16].

CARACTERÍSTICAS TÉCNICAS	
CPU	Hisilicon Kirin 910T Quad-core 1.8GHz
Sistema Operativo	Android (de Google), iOS (de Apple), Windows Phone (de Microsoft) y BlackBerry OS (de BlackBerry.
Memoria	RAM : 1 GB hasta 3 GB ROM : 1GB hasta 32 GB
Network	FDD LTE: B1/B3/B7/B8/B20 UMTS: B1/ B2/B5/B8 GSM 850/900/1800/1900MHz
GPS	GPS/A-GPS/Glonass
Conectividad	NFC, WiFi 802.11b/g/n, Bluetooth 4.0 with BLE, Micro USB 2.0

Sensores	G-sensor Acelerómetro Sensor de proximidad y luz Sensor de luz ambiental Compás
Camara	Camera Principal: 13MP AF BSI F2.0, con Flash Camera Frontal: 8MP FF
Audio	Grabación de Audio: MP3, MIDI, AMR-NB, AMR-WB, AAC, AAC+, eAAC+, AAC-LC, FLAC, WMA2-9, RA Formatos de Audio Compatibles: *.mp3, *.mid, *.amr, *.3gp, *.mp4, *.m4a, *.wav,*.ogg, *.ra
Video	Grabación de Video: MPEG-4, H.264, H.263, VP8, RV7-10, Xvid, WMV9 Formatos de Video: *.3gp, *.mp4, *.m4a, *.rm, *.rmvb, *.wmv Imágenes: PNG, GIF (Static only), JPEG, BMP Formatos de imágenes: *.png, *.gif (Static only), *.jpeg, *.bmp Orificio de audífono de 3.5mm

2.5. CARACTERÍSTICAS DEL SONIDO

El sonido es una onda mecánica la cual necesita de un medio para propagarse, se dice que es una onda longitudinal. El sonido se maneja bajo distintas frecuencias, las que el odio humano puede percibir están en el rango de los 20Hz a los 20KHz.

El sonido cumple con 3 características que hacen que el oído humano pueda distinguir entre que está produciendo el sonido, las cuales son:

Intensidad: La intensidad del sonido percibido, o propiedad que hace que éste se capte como fuerte o como débil, está relacionada con la intensidad de la onda sonora correspondiente, también llamada intensidad acústica. La intensidad acústica es una magnitud que da idea de la cantidad de energía que está fluyendo por el medio como consecuencia de la propagación de la onda.

La intensidad es medida en decibeles los cuales se miden de forma logarítmica, por ejemplo un susurro tiene 10 db y las olas en la costa una intensidad de 40 db, lo cual quiere decir que la intensidad del sonido de las olas es 1000 veces mayor que el de un susurro correspondiendo a un aumento de 30 db. Para que se produzca dolor con intensidad sonora este debe estar en los 120 db.

Tono: es la cualidad del sonido mediante la cual el oído le asigna un lugar en la escala musical, permitiendo, por tanto, distinguir entre los graves y los agudos. La magnitud física que está asociada al tono es la frecuencia. Los sonidos percibidos como graves corresponden a frecuencias bajas, mientras que los agudos son debidos a frecuencias altas. Así el sonido más grave de una guitarra corresponde a una frecuencia de 82,4 Hz y el más agudo a 698,5 hertzs.

Timbre: es la cualidad del sonido que permite distinguir sonidos procedentes de diferentes instrumentos, aun cuando posean igual tono e intensidad. Debido a esta misma cualidad es posible reconocer a una persona por su voz, que resulta característica de cada individuo.

El sonido muy pocas veces es un sonido puro, por ejemplo la voz humana, los instrumentos musicales y los cantos de las aves dan lugar a un sonido rico que posee muchas vibraciones complejas. Cada una de estas vibraciones complejas puede considerarse como un conjunto de vibraciones armónicos simples de una frecuencia y amplitud determinadas, cada una de estas vibraciones complejas puede considerarse como un conjunto de vibraciones de armónicos simples de una frecuencia y amplitud

determinadas. Está mezcla da el sonido característico de cada sonido, lo que se conoce como el timbre.

2.6 TRANSFORMADA DE FOURIER.

Para una función no periódica P→ ∞

$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-i\omega t}dt \qquad f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)e^{i\omega t}d\omega$$

La primera integral que obtiene *F* (*ω*) se denomina transformada de Fourier de *f (t*), y la segunda se denomina transformada inversa de Fourier.

El cuadrado f^2 (*t*) nos da una idea de cómo la energía contenida en la onda se distribuye en el tiempo, mientas que F^2 (*ω*) nos da una idea de cómo la energía se distribuye en el espectro de frecuencias. Naturalmente,

$$\int_{-\infty}^{\infty} |f(t)|^2 dt = \int_{-\infty}^{\infty} |F(\omega)|^2 d\omega$$

Esta herramienta matemática toma la señal, que en este caso es una señal sonora y la representa en función de la frecuencia, es decir es útil para hallar los armónicos a dicha función.

Como se trabajará en un medio digital, es decir se trabajará con paquetes de datos discretos realmente se va a utilizar es la transformada discreta de Fourier, pero cumple la misma función que la transformada de Fourier en tiempo continuo.

2.7 PATRÓN MODELO-VISTA-CONTROLADOR (MVC).

Este patrón fue descrito por primera vez en 1979 por trygveReenskaug [17] e introducido como parte de la versión Smalltalk-80 del lenguaje de programación Smalltalk.

Fue diseñado con el fin de facilitar la programación de sistemas múltiples y de sincronización de datos. Su principal ventaja es que el Modelo, la vista y los controladores se tratan como entidades separadas, de modo que todos los cambios efectuados en el modelo se ven inmediatamente reflejados en la vista.

Al incorporar esta arquitectura se puede trabajar cada parte por separado y luego al final unirlas todas en tiempos de ejecución, haciendo de este modo posible cambiar con mayor facilidad si alguna de las partes de la aplicación no funciona.

2.7.1 DEFINICIÓN DE LAS PARTES.

El modelo es donde se manejan todos los datos del programa. Este no conoce las vistas ni los controladores, es el mismo sistema quien se encarga de interconectarlos.

La vista es donde se muestran todos los datos del modelo y los muestra al usuario. Interactúa preferentemente con el controlador pero en algunos casos puede interactuar con el modelo con una referencia a este.

El controlador es el encargado de darle sentido a las órdenes del usuario, actuando sobre los datos del modelo, logrando establecer toda la interacción necesaria entre la vista y el modelo, incluso cuando uno de estos efectúa algún cambio [18].

CAPÍTULO 3. MARCO METODOLÓGICO

3.1. TIPO DE INVESTIGACIÓN

Esta investigación es de tipo práctica, debido a que toma un conocimiento puro para desarrollar una aplicación móvil, cuya finalidad es la identificación de aves que permite mejorar el acervo cultural de la protección de la avifauna, permitiendo también el desarrollo integral de la región Caribe Colombiana cuya principal fuente de ingreso es el turismo.

3.2. DISEÑO METODOLÓGICO

A continuación se muestra la metodología a desarrollar la cual consta de 5 pasos que se explicaran a continuación y que se pueden ver con claridad en la figura 1.

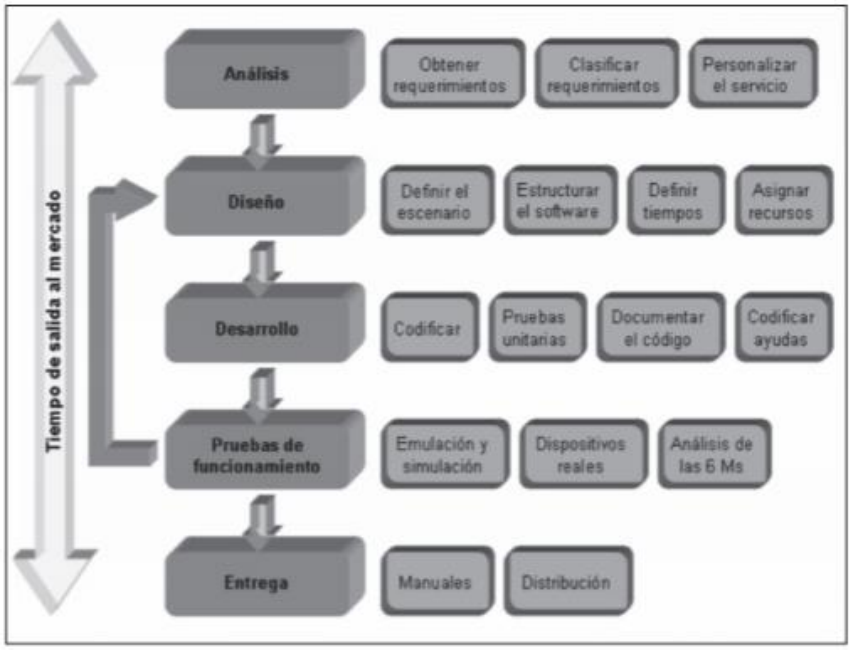


Figura 1: Etapas de la Metodología para el Desarrollo de Aplicación Móvil. (Tomado del artículo Metodología para el desarrollo de aplicaciones móviles)[19]

En la fase de análisis se hace un reconocimiento de lo que el cliente quiere que tenga la aplicación móvil y luego de identificar cada uno de los requerimientos estos se clasifican y se trabajan según su importancia.

Para identificar el problema que el cliente quiere solucionar a través de las tecnologías móviles se le hace como especie de una entrevista para que este exprese el problema o la función que el cliente requiere que realice la aplicación.

Después de conocer los requerimientos del cliente se pasa a clasificarlos según el entorno, el mundo, funcionales y no funcionales.

En el entorno podemos encontrar los requerimientos técnicos y de sistema operativo, la tecnología utilizada para la transferencia de datos y en caso de ser necesario también el DBMS (*data base managent system*).

En el mundo se encuentra todo lo que tiene que ver con la interacción del usuario con el aplicativo móvil, aquí se ve todos los requerimientos de interfaz gráfica del usuario (IGU), la forma como el software va a generar los datos de salida, el formato de los datos y la gama tecnológica de los teléfonos móviles de los usuarios a los que va dirigida el servicio.

Los requerimientos funcionales son todos aquellos que demandan una función dentro del sistema.

Los requerimientos no funcionales son la estabilidad, la portabilidad, el rendimiento, el tiempo de salida al mercado y el costo.

Luego viene la etapa de diseño que es donde se plasma el pensamiento del problema de la solución mediante diagramas o esquemas indagando en la mejor alternativa al integrar aspectos técnicos, funcionales, sociales y económicos. A esta etapa se puede volver cuando no se obtienen los resultados deseados en las pruebas.

Todo eso se descompone en cuatro fases: definir el escenario, estructurar el software, definir tiempos y asignar recursos.

Definir el escenario: las aplicaciones móviles pueden ser desarrolladas teniendo en cuenta tres escenarios que dependen del sistema de conexión y sincronización con un servidor o aplicación central, el proceso de sincronización se realiza para insertar, modificar o borrar información. Entre estos tres escenarios encontramos: 1) desconectado: todos los procesos de la aplicación se realizan en la teléfono móvil, luego si se requiere se puede conectar a una aplicación central por el proceso de sincronización. 2) semiconectado: los procesos se puede realizar en el dispositivo sin estar conectado pero en algún momento este debe estar conectado para terminar el proceso. En los entornos anteriormente mencionados se recomienda utilizar los protocolos y tecnologías que se ajusten a las capacidades técnicas del dispositivo móvil. 3) conectado: el teléfono móvil debe estar siempre conectado a la aplicación central o al servidor para un correcto funcionamiento, generalmente utilizan protocolo de hipertexto (HTTP).

Estructurar software: se deben utilizar algunos diagramas de modelado de lenguaje (UML), según indiquen las necesidades de la aplicación, como se muestra en la figura 2.

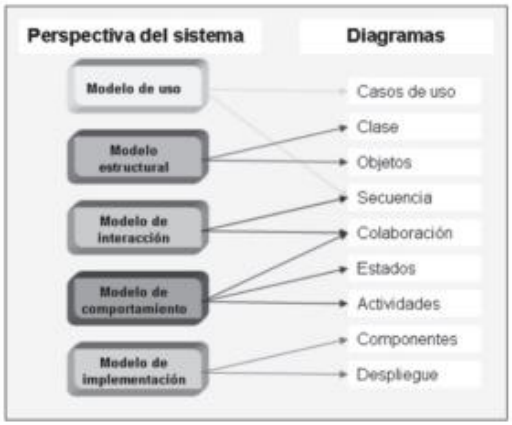


Figura 2: Posibles Diagramas Para El Desarrollo De Aplicaciones Móviles [19]

Se sugiere tomar los requerimientos obtenidos en la etapa anterior y traducirlos a un diagrama que describa en forma objetiva el servicio a implementar.

En la etapa de desarrollo es donde se toma el diseño obtenido en la etapa anterior y se plasma en software. En esta etapa se divide en las siguientes fases:

Codificar: se escriben en el lenguaje de programación, cada una de las partes seleccionadas en los diagramas realizados en la etapa de diseño.

Pruebas unitarias: se prueba cada aspecto del software-objeto, clase, actividad, documento, entre otros- de forma individual, luego se pone en funcionamiento el conjunto completo para comparar los resultados obtenidos con los esperados.

Documentar el código: a medida que se va avanzando en el código y probando cada sección del mismo, se va redactando una pequeña documentación sobre el desarrollo.

En la etapa de pruebas de funcionamiento lo que se busca es mirar cómo funciona la aplicación en distintos entornos y para eso se realizan las siguientes actividades:

Emulación y simulación: se realizan pruebas simulando el escenario y emulando el dispositivo a modo de explorar todas las funciones de la aplicación móvil, trabajando incluso con datos erróneos a modo de comprobar la robustez del software. En caso de encontrar fallos se debe regresar a la etapa de codificación en la etapa de desarrollo y solucionar los problemas, y cuando pase estás pruebas, pasar a la fase de prueba en dispositivos reales.

Dispositivos reales: se realizan pruebas de campo con dispositivos reales para ver el desempeño de la aplicación móvil. En caso de que no cumpla con lo que el cliente pide o se encuentren fallos de tiempos de ejecución se debe volver a la parte de diseño para solucionar dichos problemas.

Análisis de las 6 M’s: para valorar el potencial de éxito del servicio, se sugiere buscar un grupo de expertos en el campo del desarrollo móvil para que utilicen el método de evaluación de las 6 M’s y califiquen la presencia de los 6 atributos de la aplicación desarrollada.

En la etapa de entrega luego de depurar la aplicación y atendidos todos los requerimientos de última hora del cliente se da por finalizada la aplicación y se entrega el ejecutable, el código fuente, la documentación y el manual del sistema.

Manuales: el objetivo es el entrenamiento, una aplicación móvil debe constar de un manual del sistema donde se indique el proceso de instalación, la atención a posibles fallas en el tiempo de ejecución y las especificaciones técnicas mínimas de hardware y software que requiere el equipo, para el funcionamiento adecuado del aplicativo móvil [19].

CAPÍTULO 4 DESARROLLO DEL TRABAJO

4.1 ANÁLISIS

El análisis que se realizó para el desarrollo de la tesis comenzó con la identificación del problema y como este se puede solucionar a través de una aplicación móvil. Hablando con el estudiante de biología Ricardo Martínez quien sugiere que el aplicativo móvil debía ser capaz de identificar un ave teniendo en cuenta su canto característico, debido a que las aves poseen distintos cantos, él afirma que sólo se trabajara con ese canto que es el que las aves comúnmente emiten, y que a su vez los usuarios puedan compartir las aves que avistaron en una red social como twitter, Facebook o instagram, además que puedan llevar un control sobre las aves que han visto. Teniendo en cuenta lo que nos cuenta el estudiante de biología que las personas que van a realizar avistamiento de aves cuentan con dispositivos móviles de gama media-alta y gama alta, no se va a tener mayor inconveniente con problemas de latencia.

De acuerdo a lo anterior se recomienda que el dispositivo móvil a usar por el usuario para garantizar el correcto funcionamiento del aplicativo móvil cuente con el sistema operativo (SO) Android desde su versión 4.1 Jelly Bean en adelante, puesto que estos contienen un hardware más actualizado que les permite realizar las tareas de análisis de frecuencia de una forma más rápida y tienen micrófonos de mejor calidad, las cuales son las piezas fundamentales para el funcionamiento de la aplicación móvil. Pero la aplicación móvil fue desarrollada para Android desde la versión 2.3.3 gingerbread en adelante para acceder a un mayor número de personas.

En la parte de la interfaz de usuario (IGU) se realizó de una forma muy intuitiva y con pasos fáciles de mecanizar, donde los datos de salida se generan en forma de nuevas pantallas que muestran toda la información necesaria que requiere el usuario, teniendo en cuenta que el aplicativo va dirigido a personas que posean dispositivos móviles que van desde la gama media-alta a la gama alta, para que puedan tener la menor latencia en la obtención de los datos.

Los requerimientos funcionales de la aplicación son:

- Identificar el ave teniendo en cuenta su canto característico.
- Poder visualizar las aves disponibles para identificar con la aplicación.
- Cuando se encuentre un ave, que el usuario pueda certificar o rechazar la coincidencia encontrada.
- Cuando se encuentre un ave que el usuario pueda compartir en alguna red social.
- Que el usuario pueda ver las aves que ha encontrado con la fecha en que la ha encontrado.

Luego de analizar los requerimientos no funcionales es necesario que el aplicativo móvil sea estable y sea portable para que los usuarios puedan llevarla a donde quieran y puedan usarla sin ningún problema.

4.2 DISEÑO

Basado en el diagrama de casos de usos que se ve a continuación se diseñó el aplicativo móvil.

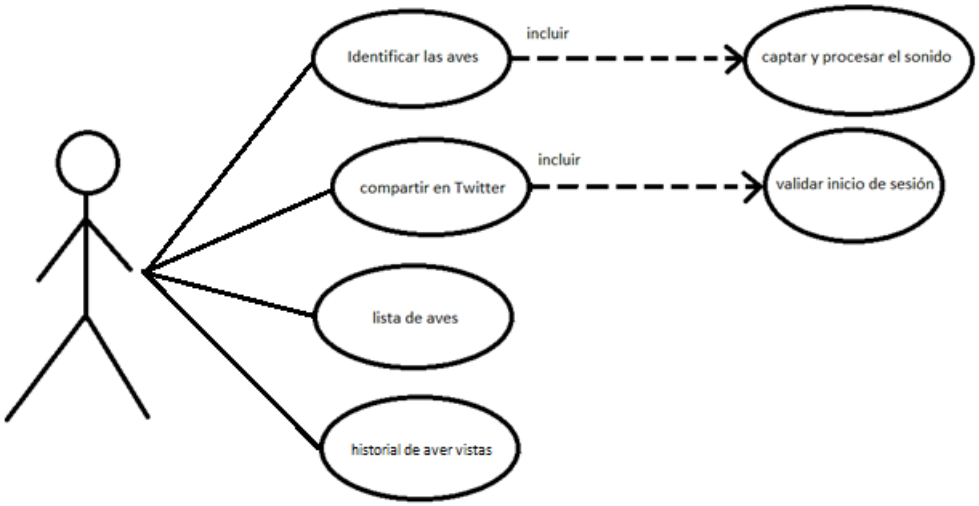


Figura 3. Diagrama de Usos de la Aplicación

La aplicación se diseñó para un entorno completamente desconectado, debido a que las zonas donde por lo general las personas van a hacer este tipo de actividades se encuentran fuera de la cobertura de redes móviles, por lo que cada proceso se realizara dentro del dispositivo móvil, pero cuando requiera una actualización a modo de agregar más aves a la cantidad que esté pueda identificar o cuando se quiera agregar funciones adicionales, la aplicación debe ser conectada a internet.

Teniendo en cuenta el diagrama UML que se ve en la figura 3 se procede a traducir esos requerimientos del cliente a software que es lo que se ve en la siguiente etapa del desarrollo de la metodología que se llama desarrollo.

Esto se realizó teniendo en cuenta el patrón Modelo-Vista-Controlador (MVC) teniendo en cuenta el siguiente flujo del proceso:

1. La aplicación móvil actúa como un componente de vista, que hace una solicitud de datos que se encuentran dentro de la misma, que los comparará con datos externos a la misma.

2. Teniendo en cuenta los datos externos y comparándolos con los datos internos de la aplicación, hace una llamada del objeto apropiado para luego ser mostrado al controlador.

3. Luego de que se entrega el objeto apropiado, el controlador responde de nuevo a la aplicación móvil, la cual responde de forma acertada a lo que el controlador solicita.

4.3 DESARROLLO

El sistema fue desarrollado a partir de seis (6) Layout, de los cuales tres (3) son principales, que siguen una secuencia lógica para que la aplicación cumpla con su objetivo y sea muy intuitivo para los usuarios

Se decidió fraccionar la aplicación en tres (3) ventanas, muy sencillas y fáciles de utilizar en las que encontramos una pantalla principal en la cual solo debemos presionar un botón llamado “encuentre muchas aves” como se ve en la figura 4, para pasar a la parte de análisis del sonido.

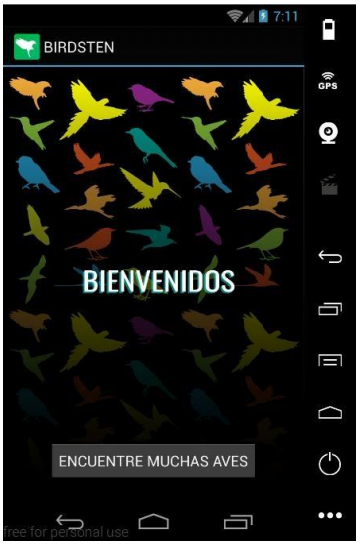


Figura 4. Panatalla Inicio de la aplicacion

Luego en la pantalla donde se realizará todo el análisis de sonido (figura 5), en esta parte el proyecto sufrió un cambio, puesto que en principio se tenía planeado utilizar una grabación con un tiempo limitado de 20 segundos, pero es muy difícil poder captar en ese espacio corto de tiempo el sonido que realmente se quiere tener como lo pudo comprobar el estudiante de biología Ricardo Martínez cuando se le dio la aplicación para que fuera a minca a probar en el campo. Por lo que se procedió a cambiar por un análisis en tiempo real y de este modo mostrar una gráfica para que el usuario tenga la sensación de que la aplicación funciona de forma adecuada.

En este apartado se tomó como referencia una aplicación de código abierto en la que se puede ver el espectro del sonido en tiempo en real, la cual sirvió de plantilla para el desarrollo de esta aplicación [20].

Donde se encontró una librería que realiza la transformada de Fourier a través del algoritmo de la transformada rápida de Fourier, como se ve en el siguiente código:

```
/*----------------------------------------------------------
 radfg: Real FFT's forward processing of general factor
----------------------------------------------------------*/
void radfg(int ido, int ip, int l1, int idl1, double cc[],
           double c1[], double c2[], double ch[], double ch2[],
           final double wtable[], int offset) {
  final double twopi=2.0D*Math.PI; //6.28318530717959;
  int     idij, ipph, i, j, k, l, j2, ic, jc, lc, ik, is, nbd;
  double  dc2, ai1, ai2, ar1, ar2, ds2, dcp, arg, dsp, ar1h, ar2h;
  int iw1 = offset;

  arg=twopi / ip;
  dcp=Math.cos(arg);
  dsp=Math.sin(arg);
  ipph=(ip+1)/ 2;
  nbd=(ido-1)/ 2;
  if(ido !=1) {
    for(ik=0; ik<idl1; ik++) ch2[ik]=c2[ik];
    for(j=1; j<ip; j++)
      for(k=0; k<l1; k++)
        ch[(k+j*l1)*ido]=c1[(k+j*l1)*ido];
    if(nbd<=l1) {
      is=-ido;
      for(j=1; j<ip; j++) {
        is+=ido;
        idij=is-1;
        for(i=2; i<ido; i+=2) {
          idij+=2;
          for(k=0; k<l1; k++) {
            ch[i-1+(k+j*l1)*ido]=
              wtable[idij-1+iw1]*c1[i-1+(k+j*l1)*ido]
                                 +wtable[idij+iw1]*c1[i+(k+j*l1)*ido];
            ch[i+(k+j*l1)*ido]=
              wtable[idij-1+iw1]*c1[i+(k+j*l1)*ido]
                                 -wtable[idij+iw1]*c1[i-1+(k+j*l1)*ido];
          }
        }
      }
    } else {
      is=-ido;
      for(j=1; j<ip; j++) {
        is+=ido;
        for(k=0; k<l1; k++) {
          idij=is-1;
          for(i=2; i<ido; i+=2) {
            idij+=2;
            ch[i-1+(k+j*l1)*ido]=
              wtable[idij-1+iw1]*c1[i-1+(k+j*l1)*ido]
                                 +wtable[idij+iw1]*c1[i+(k+j*l1)*ido];
            ch[i+(k+j*l1)*ido]=
              wtable[idij-1+iw1]*c1[i+(k+j*l1)*ido]
                                 -wtable[idij+iw1]*c1[i-1+(k+j*l1)*ido];
          }
        }
      }
    }
    if(nbd>=l1) {
      for(j=1; j<ipph; j++) {
        jc=ip-j;
        for(k=0; k<l1; k++) {
          for(i=2; i<ido; i+=2) {
            c1[i-1+(k+j*l1)*ido]=ch[i-1+(k+j*l1)*ido]+ch[i-1+(k+jc*l1)*ido];
            c1[i-1+(k+jc*l1)*ido]=ch[i+(k+j*l1)*ido]-ch[i+(k+jc*l1)*ido];
            c1[i+(k+j*l1)*ido]=ch[i+(k+j*l1)*ido]+ch[i+(k+jc*l1)*ido];
```

```
            c1[i+(k+jc*l1)*ido]=ch[i-1+(k+jc*l1)*ido]-ch[i-1+(k+j*l1)*ido];
          }
        }
      }
    } else {
      for(j=1; j<ipph; j++) {
        jc=ip-j;
        for(i=2; i<ido; i+=2) {
          for(k=0; k<l1; k++) {
            c1[i-1+(k+j*l1)*ido]=
              ch[i-1+(k+j*l1)*ido]+ch[i-1+(k+jc*l1)*ido];
            c1[i-1+(k+jc*l1)*ido]=ch[i+(k+j*l1)*ido]-ch[i+(k+jc*l1)*ido];
            c1[i+(k+j*l1)*ido]=ch[i+(k+j*l1)*ido]+ch[i+(k+jc*l1)*ido];
            c1[i+(k+jc*l1)*ido]=ch[i-1+(k+jc*l1)*ido]-ch[i-1+(k+j*l1)*ido];
          }
        }
      }
    }
  } else {
    for(ik=0; ik<idl1; ik++) c2[ik]=ch2[ik];
  }
  for(j=1; j<ipph; j++) {
    jc=ip-j;
    for(k=0; k<l1; k++) {
      c1[(k+j*l1)*ido]=ch[(k+j*l1)*ido]+ch[(k+jc*l1)*ido];
      c1[(k+jc*l1)*ido]=ch[(k+jc*l1)*ido]-ch[(k+j*l1)*ido];
    }
  }

  ar1=1;
  ai1=0;
  for(l=1; l<ipph; l++) {
    lc=ip-l;
    ar1h=dcp*ar1-dsp*ai1;
    ai1=dcp*ai1+dsp*ar1;
    ar1=ar1h;
    for(ik=0; ik<idl1; ik++) {
      ch2[ik+l*idl1]=c2[ik]+ar1*c2[ik+idl1];
      ch2[ik+lc*idl1]=ai1*c2[ik+(ip-1)*idl1];
    }
    dc2=ar1;
    ds2=ai1;
    ar2=ar1;
    ai2=ai1;
    for(j=2; j<ipph; j++) {
      jc=ip-j;
      ar2h=dc2*ar2-ds2*ai2;
      ai2=dc2*ai2+ds2*ar2;
      ar2=ar2h;
      for(ik=0; ik<idl1; ik++) {
        ch2[ik+l*idl1]+=ar2*c2[ik+j*idl1];
        ch2[ik+lc*idl1]+=ai2*c2[ik+jc*idl1];
      }
    }
  }
  for(j=1; j<ipph; j++)
    for(ik=0; ik<idl1; ik++)
      ch2[ik]+=c2[ik+j*idl1];

  if(ido>=l1) {
    for(k=0; k<l1; k++) {
      for(i=0; i<ido; i++) {
        cc[i+k*ip*ido]=ch[i+k*ido];
      }
    }
```

```
  } else {
    for(i=0; i<ido; i++) {
      for(k=0; k<l1; k++) {
        cc[i+k*ip*ido]=ch[i+k*ido];
      }
    }
  }
  for(j=1; j<ipph; j++) {
    jc=ip-j;
    j2=2*j;
    for(k=0; k<l1; k++) {
      cc[ido-1+(j2-1+k*ip)*ido]=ch[(k+j*l1)*ido];
      cc[(j2+k*ip)*ido]=ch[(k+jc*l1)*ido];
    }
  }
  if(ido==1) return;
  if(nbd>=l1) {
    for(j=1; j<ipph; j++) {
      jc=ip-j;
      j2=2*j;
      for(k=0; k<l1; k++) {
        for(i=2; i<ido; i+=2) {
          ic=ido-i;
          cc[i-1+(j2+k*ip)*ido]=ch[i-1+(k+j*l1)*ido]+ch[i-1+(k+jc*l1)*ido];
          cc[ic-1+(j2-1+k*ip)*ido]=ch[i-1+(k+j*l1)*ido]-ch[i-1+(k+jc*l1)*ido];
          cc[i+(j2+k*ip)*ido]=ch[i+(k+j*l1)*ido]+ch[i+(k+jc*l1)*ido];
          cc[ic+(j2-1+k*ip)*ido]=ch[i+(k+jc*l1)*ido]-ch[i+(k+j*l1)*ido];
        }
      }
    }
  } else {
    for(j=1; j<ipph; j++) {
      jc=ip-j;
      j2=2*j;
      for(i=2; i<ido; i+=2) {
        ic=ido-i;
        for(k=0; k<l1; k++) {
          cc[i-1+(j2+k*ip)*ido]=ch[i-1+(k+j*l1)*ido]+ch[i-1+(k+jc*l1)*ido];
          cc[ic-1+(j2-1+k*ip)*ido]=ch[i-1+(k+j*l1)*ido]-ch[i-1+(k+jc*l1)*ido];
          cc[i+(j2+k*ip)*ido]=ch[i+(k+j*l1)*ido]+ch[i+(k+jc*l1)*ido];
          cc[ic+(j2-1+k*ip)*ido]=ch[i+(k+jc*l1)*ido]-ch[i+(k+j*l1)*ido];
        }
      }
    }
  }
}
```

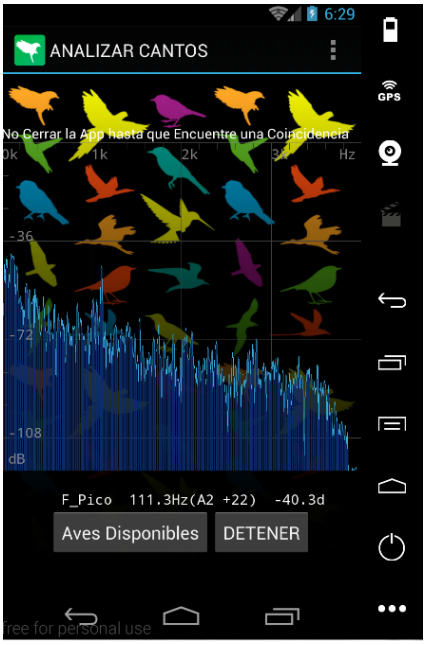


Figura 5. Pantalla Grafica de Analisis de Sonido

Dentro de esta pantalla también podemos encontrar un botón llamado “lista de aves” que permite ver la cantidad de aves que se encuentran disponibles para ser identificadas a través de la aplicación como se puede observar en la figura 6. También en caso de un fallo de la aplicación a la hora de hacer el análisis del sonido se colocó un botón de detener, el cual finaliza el proceso y te retorna a la pantalla principal.

A modo de comparar los datos obtenidos por la transformada de fourier con los datos optenidos de los estudios previos de sonido que se hicieron, se utilizo la comparación del armonico principal de cada una de las aves, con el código que se puede ver a continuación.

2 Algoritmo Para El Ca Calculo De La Maxima Amplitud De Frecuencia

```
void calculatePeak() {
    getSpectrumAmpDB();
    // Find and show peak amplitude
    maxAmpDB  = 20 * Math.log10(0.125/32768);
    maxAmpFreq = 0;
    for (int i = 1; i < spectrumAmpOutDB.length; i++) {  // skip the direct
current term
        if (spectrumAmpOutDB[i] > maxAmpDB) {
            maxAmpDB  = spectrumAmpOutDB[i];
            maxAmpFreq = i;
        }
    }
    maxAmpFreq = maxAmpFreq * sampleRate / fftLen;

    if (sampleRate / fftLen < maxAmpFreq && maxAmpFreq < sampleRate/2 -
sampleRate / fftLen) {
        int id = (int)(Math.round(maxAmpFreq/sampleRate*fftLen));
        double x1 = spectrumAmpOutDB[id-1];
        double x2 = spectrumAmpOutDB[id];
        double x3 = spectrumAmpOutDB[id+1];
        double c = x2;
        double a = (x3+x1)/2 - x2;
        double b = (x3-x1)/2;
        if (a < 0) {
            double xPeak = -b/(2*a);
            if (Math.abs(xPeak) < 1) {
                maxAmpFreq += xPeak * sampleRate / fftLen;
```

```
                    maxAmpDB = (4*a*c - b*b)/(4*a);
                }
            }
        }
    }
```

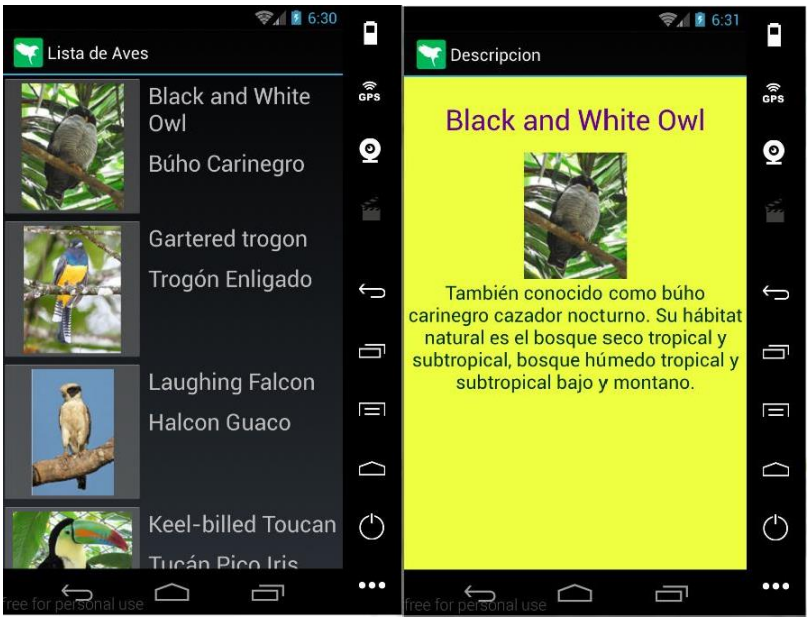


Figura 6. En la parte Izquierda e aprecia la pantalla que muestra la lista de ave disponibles. En la parate Derecha se observa la pantalla en la que se vizualiza la descricion y Nombre de cada Ave.

Cuando se presiona el botón de aves disponibles para visualizar las aves que se pueden identificar con la aplicación móvil, se abre una nueva pantalla que nos permite visualizar el nombre en inglés y en español de cada una de las aves, acompañadas de una imagen de la misma la cual se puede presionar para que nos muestre la descripción de esa ave, como se ve en la figura 6.

En la ventana donde se puede observar el espectro del sonido es donde se realiza lo fundamental de la aplicación, es aquí donde se realiza el análisis en frecuencia utilizando el algoritmo de la transformada rápida de Fourier (FFT), por medio del cual se utiliza la menor cantidad posible recursos garantizando de este modo que dispositivos de baja potencia al igual que los más potentes puedan realizar la tarea sin mayor contratiempo.

Al instante cuando se consiga una similitud entre los valores que nos arroja el análisis en frecuencia y los que se tiene ya definidos según el análisis de sonido que se realizó previo a cada canto de ave, entonces nos mostrara la tercera pantalla(figura 7), con la lógica presente en el siguiente código:

3 Algoritmos Procesamiento de los Datos obtenidos por el Analisis de Frecuencia

```
import android.annotation.TargetApi;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Build;
import android.os.Bundle;
import android.support.annotation.StringRes;
import android.view.View;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import java.util.Calendar;

import static java.security.AccessController.getContext;


public class Descri_ave extends Activity implements View.OnClickListener {
    TextView titulo, descripcion;
    ImageView imagen;
    ImageButton correcto, compartir, incorrecto;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.descri_ave);
        titulo = (TextView) findViewById(R.id.texttitulo);
        descripcion = (TextView) findViewById(R.id.textDescri);
        imagen = (ImageView) findViewById(R.id.imageView5);
        incorrecto = (ImageButton) findViewById(R.id.incorr);
        correcto = (ImageButton) findViewById(R.id.confir);
        compartir = (ImageButton) findViewById(R.id.compa);

        incorrecto.setOnClickListener(this);
        correcto.setOnClickListener(this);
        compartir.setOnClickListener(this);


        Intent uno = getIntent();
        Bundle llego = uno.getExtras();
        if (llego != null) {


            String cadena3 = (String) llego.get("nr_ave");

            switch (cadena3) {
                case "1":
                    titulo.setText(R.string.T_A_1);
                    imagen.setImageResource(R.drawable.pajaro1);
                    descripcion.setText(R.string.D_A_1);

                    break;

                case "2":
                    titulo.setText(R.string.T_A_2);
                    imagen.setImageResource(R.drawable.pajaro2);
                    descripcion.setText(R.string.D_A_2);
```

```
            break;
        case "3":
            titulo.setText(R.string.T_A_3);
            imagen.setImageResource(R.drawable.pajaro3);
            descripcion.setText(R.string.D_A_3);


            break;

        case "4":
            titulo.setText(R.string.T_A_4);
            imagen.setImageResource(R.drawable.pajaro4);
            descripcion.setText(R.string.D_A_4);
            break;
        case "5":
            titulo.setText(R.string.T_A_5);
            imagen.setImageResource(R.drawable.pajaro5);
            descripcion.setText(R.string.D_A_5);
            break;

        case "6":
            titulo.setText(R.string.T_A_6);
            imagen.setImageResource(R.drawable.pajaro6);
            descripcion.setText(R.string.D_A_6);
            break;
        case "7":
            titulo.setText(R.string.T_A_7);
            imagen.setImageResource(R.drawable.pajaro7);
            descripcion.setText(R.string.D_A_7);
            break;

        case "8":
            titulo.setText(R.string.T_A_8);
            imagen.setImageResource(R.drawable.pajaro8);
            descripcion.setText(R.string.D_A_8);
            break;
        case "9":
            titulo.setText(R.string.T_A_9);
            imagen.setImageResource(R.drawable.pajaro9);
            descripcion.setText(R.string.D_A_9);
            break;

        case "10":
            titulo.setText(R.string.T_A_10);
            imagen.setImageResource(R.drawable.pajaro10);
            descripcion.setText(R.string.D_A_10);
            break;
        case "11":
            titulo.setText(R.string.T_A_11);
            imagen.setImageResource(R.drawable.pajaro11);
            descripcion.setText(R.string.D_A_11);
            break;

        case "12":
            titulo.setText(R.string.T_A_12);
            imagen.setImageResource(R.drawable.pajaro12);
            descripcion.setText(R.string.D_A_12);
            break;
        case "13":
            titulo.setText(R.string.T_A_13);
            imagen.setImageResource(R.drawable.pajaro13);
            descripcion.setText(R.string.D_A_13);
            break;

        case "14":
```

```
                titulo.setText(R.string.T_A_14);
                imagen.setImageResource(R.drawable.pajaro14);
                descripcion.setText(R.string.D_A_14);
                break;
            case "15":
                titulo.setText(R.string.T_A_15);
                imagen.setImageResource(R.drawable.pajaro15);
                descripcion.setText(R.string.D_A_15);
                break;

            case "16":
                titulo.setText(R.string.T_A_16);
                imagen.setImageResource(R.drawable.pajaro16);
                descripcion.setText(R.string.D_A_16);
                break;
            case "17":
                titulo.setText(R.string.T_A_17);
                imagen.setImageResource(R.drawable.pajaro17);
                descripcion.setText(R.string.D_A_17);
                break;

            case "18":
                titulo.setText(R.string.T_A_18);
                imagen.setImageResource(R.drawable.pajaro18);
                descripcion.setText(R.string.D_A_18);
                break;
            case "19":
                titulo.setText(R.string.T_A_19);
                imagen.setImageResource(R.drawable.pajaro19);
                descripcion.setText(R.string.D_A_19);
                break;

            case "20":
                titulo.setText(R.string.T_A_20);
                imagen.setImageResource(R.drawable.pajaro20);
                descripcion.setText(R.string.D_A_20);
                break;

            case "21":
                titulo.setText(R.string.T_A_21);
                imagen.setImageResource(R.drawable.pajaro21);
                descripcion.setText(R.string.D_A_21);
                break;

            case "22":
                titulo.setText(R.string.T_A_22);
                imagen.setImageResource(R.drawable.pajaro22);
                descripcion.setText(R.string.D_A_22);
                break;

            case "23":
                titulo.setText(R.string.T_A_23);
                imagen.setImageResource(R.drawable.pajaro23);
                descripcion.setText(R.string.D_A_23);
                break;

            case "24":
                titulo.setText(R.string.T_A_24);
                imagen.setImageResource(R.drawable.pajaro24);
                descripcion.setText(R.string.D_A_24);
                break;

            case "25":
                titulo.setText(R.string.T_A_25);
                imagen.setImageResource(R.drawable.pajaro25);
```

```
                descripcion.setText(R.string.D_A_25);
                break;

            case "26":
                titulo.setText(R.string.T_A_26);
                imagen.setImageResource(R.drawable.pajaro26);
                descripcion.setText(R.string.D_A_26);
                break;

            case "27":
                titulo.setText(R.string.T_A_27);
                imagen.setImageResource(R.drawable.pajaro27);
                descripcion.setText(R.string.D_A_27);
                break;

            case "28":
                titulo.setText(R.string.T_A_28);
                imagen.setImageResource(R.drawable.pajaro28);
                descripcion.setText(R.string.D_A_28);
                break;

            case "29":
                titulo.setText(R.string.T_A_29);
                imagen.setImageResource(R.drawable.pajaro29);
                descripcion.setText(R.string.D_A_29);
                break;

            case "30":
                titulo.setText(R.string.T_A_30);
                imagen.setImageResource(R.drawable.pajaro30);
                descripcion.setText(R.string.D_A_30);
                break;
            default:
                break;

        }


    }
}
```

Figura 7. Panatalla que Muestra la aplaicacion al Encontrar un Ave

Luego aquí tenemos 3 botones que nos permiten 3 acciones diferentes, el botón rojo, es en caso de que el ave que muestre la aplicación, no sea el ave que el usuario realmente está viendo y regresa a la pantalla donde se analiza el espectro de sonido.

Luego está el botón del *check* el cual cada usuario puede presionar en caso de que la aplicación haya realizado una correcta identificación, y este botón le permitirá ver el historial de aves que ha identificado, para esto se utilizó la función Sharedpreferences de Java, como se muestra en el siguiente código:

4 Algoritmos Donde se crea y Guarda la Lista de Aves Vistas

```
case R.id.confir:
    Calendar fecha =Calendar.getInstance();
    int dia =fecha.get(Calendar.DAY_OF_MONTH);
    int mes =fecha.get(Calendar.MONTH);
    int ano =fecha.get(Calendar.YEAR);
    String dia1 =String.valueOf(dia);
    String mes1 =String.valueOf(mes+1);
    String ano1 =String.valueOf(ano);
    String dma = (dia1+"/"+mes1+"/"+ano1+".");
    SharedPreferences avevista =getSharedPreferences("Avevista",
Context.MODE_PRIVATE);
    String avecanto= avevista.getString("avecanto","");
    String cuando =avevista.getString("cuando","");
    String ima=avevista.getString("ima1","");
    SharedPreferences.Editor  editor= avevista.edit();
    editor.clear();
    editor.apply();
    editor.putString("avecanto",avecanto+" "+m_titulo);
    editor.putString("cuando",cuando+" "+dma);

    editor.commit();
      Toast.makeText(getBaseContext(),(m_titulo+"
"+dma),Toast.LENGTH_LONG).show();
    Intent intent1 = new Intent(Descri_ave.this,Aves_vistas.class);
    startActivity(intent1);

    break;
```

En algoritmos se puede ver como se guardan los datos, y en el siguiente se verá cómo se leen los datos (algoritmos 4):

5 Algoritmos para leer la lista de Aves Vistas

```
public class Aves_vistas extends Activity {
   Button borrar1,seguir,v_lis;
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.aves_vistas);
      borrar1=(Button)findViewById(R.id.borrar) ;
      seguir = (Button) findViewById(R.id.s_buscando);
      v_lis = (Button) findViewById(R.id.lista);
```

```
        seguir.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                Intent mainIntent1 = new Intent().setClass(Aves_vistas.this, AnalyzerActivity.class);
                startActivity(mainIntent1);
            }
        });
        v_lis.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                Intent mainIntent2 = new Intent().setClass(Aves_vistas.this,Lis_aves.class);
                startActivity(mainIntent2);
            }
        });


        borrar1.setOnClickListener(new View.OnClickListener() {
            @TargetApi(Build.VERSION_CODES.GINGERBREAD)
            @Override
            public void onClick(View v) {
               SharedPreferences avevista= getSharedPreferences("Avevista",Context.MODE_PRIVATE);
                SharedPreferences.Editor  editor= avevista.edit();
                editor.clear();
                editor.apply();
                editor.remove("avecanto");
                editor.remove("cuando");
                editor.commit();
                Intent Intent3 = new Intent().setClass(Aves_vistas.this, AnalyzerActivity.class);
                startActivity(Intent3);

            }


        });
        final LinearLayout layoutdinamico = (LinearLayout) findViewById(dinamico);
        final SharedPreferences avevista = getSharedPreferences("Avevista", Context.MODE_PRIVATE);
        String nom1 = avevista.getString("avecanto", "no ");
        String[] nomV = nom1.split(" ");
        String fec1 = avevista.getString("cuando", "no ");
        String[] fecV = fec1.split(" ");
        String imagen1 = avevista.getString("ima1", "no ");
        String[] imaV = imagen1.split(" ");


        for (int f = 1; f < nomV.length; f++) {
            String nombre = "Nombre del Ave: ";
            String fecha = "Fecha de Avistamiento:";
            TextView nomT=new TextView(this);
            TextView fecT=new TextView(this);


            nomT.setTextSize(15);
            nomT.setTextColor(Color.YELLOW);
            nomT.setText(nombre+""+ nomV[f]);
            layoutdinamico.addView(nomT);

            fecT.setTextColor(Color.YELLOW);
            fecT.setTextSize(15);
            fecT.setText( fecha+"" +fecV[f]);
            layoutdinamico.addView(fecT);


        }

    }
}
```

Luego de presionar este botón el usuario vera una lista de las aves que ha identificado, con las fechas en la que estos fueron avistados como se observa en la figura 8.

Donde también se encuentran tres botones los caules son:

- “Seguir buscando”: el que se presiona cuando él usuario quiera buscar más aves y lo envía a la pantalla donde se hace el análisis de sonido.
- “Lista de aves”: en este botón el usuario puede observar la lista de aves disponibles para identificar a través del aplicativo móvil.
- “Borrar historial”: este botón le permite al usuario borrar el historial de aves identificadas cuando este así lo quiera.

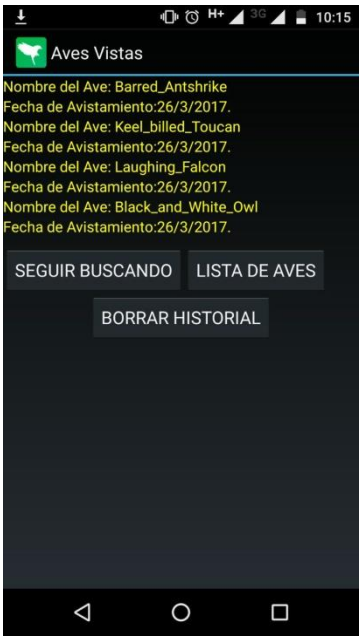


Figura 8. Pantalla donde se observa el Historial de Aves Vistas

Y por último está el botón con el logotipo de la red social Twitter donde se puede compartir el ave avistada de modo tal que todos nuestros amigos y personas que no conozcamos que estén usando la aplicación puedan ver que ave hemos visto.

Él código es el que vemos a continuación:

6 Algoritmo Para Compartir Nombre Del Ave Vista En Twitter

```
Intent intent = new Intent(Intent.ACTION_SEND);
intent.setType("text/plain");
intent.putExtra(Intent.EXTRA_TEXT, "ENCONTRE EL AVE:"+" "+m_titulo+ " "
+"#UNIMAGDALENA #BIRDSTEN");
intent.setPackage("com.twitter.android");
startActivity(intent);
```

Todos los códigos usados en el desarrollo de la aplicación han sido agregados como anexos.

4.4 PRUEBAS DE FUNCIONAMIENTO

En principio cuando se manejaba la aplicación con una grabación determinada de tiempo en las pruebas con entornos controlados se obtuvo buenos resultados iguales a los expuestos en la tabla 1 pero a la hora de llevar esta prueba a un entorno no controlado en donde no se sabía en qué momento un ave cantaría o no, era muy difícil proporcionar a la aplicación un dato real el cual analizar para poder hacer la comparación, basados en la prueba realizada por el estudiante de biología Ricardo Martinez. Debido a que se tuvo esa falencia fue necesario regresar de nuevo a la etapa de desarrollo en donde se paleo este problema utilizando un análisis en tiempo real, de modo tal que sin saber el momento exacto en el que el ave vaya a cantar se puede analizar el audio y comparar y al final utilizar una información veraz, como lo comprobó más adelante en una prueba de campo el mismo estudiante de Biología.

Después de la modificación se realizaron pruebas en entornos controlados teniendo resultados iguales a los obtenidos en la prueba anterior, luego con los entornos no controlados, con la ayuda de un micrófono que permite captar mucho mejor el sonido de las aves, teniendo en cuenta que no se observaron todas las aves presentes en la aplicación sino unas cuantas y sobre esas fue que se pudo verificar el funcionamiento de la aplicación, datos que se relacionarán mejor en el capítulo de resultados y discusión.

CAPÍTULO 5 RESULTADOS Y DISCUSIÓN

Ahora se van a exponer los resultados obtenidos de las diferentes pruebas realizadas a la aplicación móvil que se ha desarrollado.

5.1 PRUEBA COMPARACIÓN CON SOFTWARE ESPECIALIZADO

Partiendo de la comparación de los resultados del análisis de frecuencia realizado con el programa especializado de audio audacity y la aplicación desarrollada, exponiendo un ejemplo que se ve a continuación:

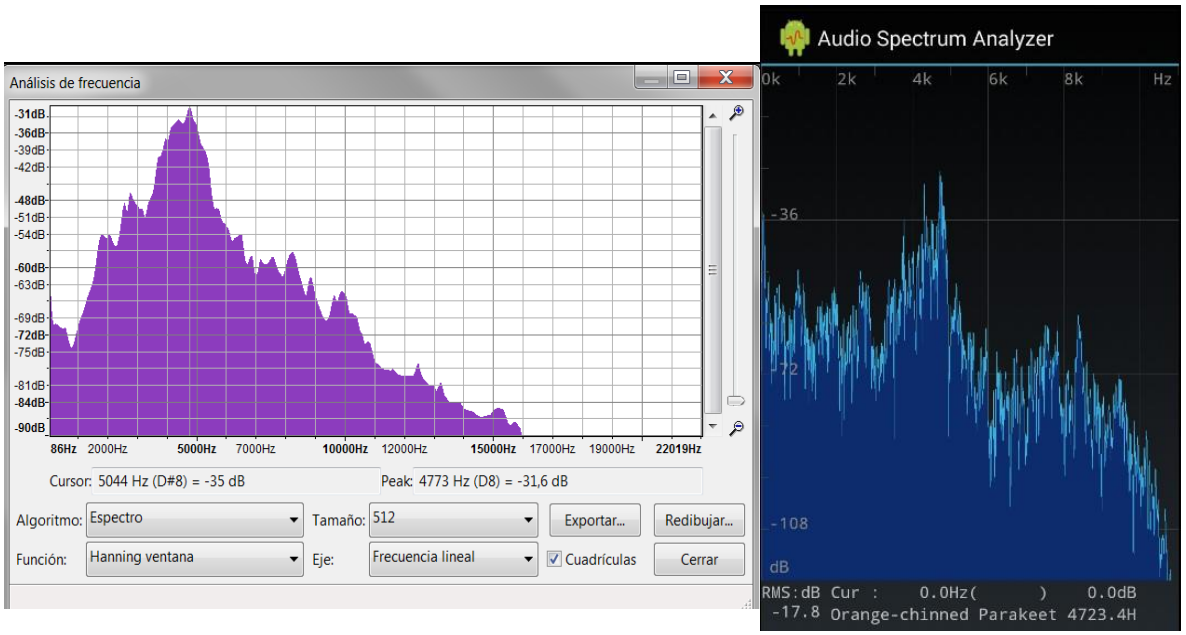


Figura 9. Parte Izquierda Analisis de Espectro en Audacity. Parte Derecha Analisis de espectro en la Aplicacion

En la imagen 8 se pueden apreciar del lado izquierdo los resultados obtenidos por el programa especializado de sonido Audacity, el cual es realizado teniendo en cuenta un sonido digitalizado, y a la derecha podemos ver el obtenido por la aplicación móvil que estaba en fase desarrollo, el cual es por sonido captado a partir del micrófono del dispositivo. Como se puede observar la diferencia de su fundamental no es muy grande, pero varia un poco y esto en parte es debido al ruido ambiente que capta el micrófono y a que la transformada realizada por Audacity tiene un tamaño de muestreo de 512 mientras que la aplicación móvil tiene una tasa de muestreo de 2048.

Basado en los resultados anteriores se decidió dar un rango de identificación para cada ave de más o menos 50 Hz de su frecuencia fundamental a modo de garantizar una mayor precisión a la hora de determinar que ave está cantando.

También es fácil de observar como la distribución del espectro es muy diferente, lo cual se atribuye a la cantidad de ruido ambiente que en ese momento teníamos con el aplicativo móvil, pero aun así se observa con claridad que el armónico principal captado por uno y el otro se encuentran muy próximos entre sí.

5.2 PRUEBA DE IDENTIFICACIÓN DE AVES

Se realizó una prueba con la aplicación móvil en donde se buscó la capacidad de la misma para identificar cada una de las aves, haciendo la prueba 10 veces con cada uno de los especímenes que el aplicativo móvil puede identificar, obteniendo como resultado lo que se puede observar en la tabla 1.

Tabla 1 Resultados Prueba de Funcionamiento de la Aplicación en Entorno Controlado

Ave	Acertados	Fallidos	%	Aves	Acertados	Fallidos	%
1	9	1	90	16	9	1	90
2	6	4	60	17	8	2	80
3	10	0	100	18	9	1	90
4	9	1	90	19	7	3	70
5	8	2	80	20	6	4	60
6	4	6	40	21	5	5	50
7	8	2	80	22	6	4	60
8	8	2	80	23	2	8	20
9	3	7	30	24	8	2	80
10	7	3	70	25	2	8	20
11	8	2	80	26	7	3	70
12	9	1	90	27	3	7	30
13	9	1	90	28	8	2	80
14	6	4	60	29	6	4	60
15	8	2	80	30	9	1	90
					porcentaje asertividad=		69

Teniendo en cuenta los resultados obtenidos en esta prueba se puede decir que la aplicación algunos especímenes los identifica con mucha facilidad y otros con mayor dificultad, lo cual tiene que ver con la similitud en términos de frecuencia que encuentran con otros cantos dentro del aplicativo, por lo que se pueden obtener en ciertos casos falsas identificaciones o la no identificación de dichos especímenes.

Debido a que la probabilidad de identificación de las aves es del 69% se dice que el aplicativo es funcional, pero dicha funcionalidad no se queda hasta los 30 especímenes que este aplicativo es capaz de identificar, sino que dicho principio se puede aplicar a muchas otras aves, solo es necesario realizar ciertas mejoras de software a la hora de filtrar el sonido, o trabajar con más armónicos, debido a que este aplicativo solamente trabaja con él armónico principal del canto del ave.

Adicional a este método de identificación de aves a través del canto, se puede añadir un reconocimiento por medio de procesamiento de imágenes, el cual puede ser un soporte en caso de que el sonido no sirva y se tenga una línea visual del ave que se quiere identificar, logrando de este modo reducir aún más el margen de error del aplicativo.

Luego de realizar estas pruebas la aplicación y las indicaciones de como instalarla y usarla se le entregaron al estudiante de Biología Ricardo Martinez para que el fuera al campo a probar dicha aplicación (Figura 10). Para dicha prueba él llevo un micrófono para garantizar una mejor escucha de las aves, pero también hizo una prueba utilizando nada más los micrófonos encontrados en el teléfono móvil. La mayor dificultad que se encontró con esta prueba es que no se pudieron avistar todas las aves presentes en el aplicativo, pero se trabajó con las que él pudo observar. Los resultados se encuentran presentes en la Tabla 2 para las pruebas hechas sin micrófono especializado y la Tabla 3 para las hechas con el micrófono especializado (Anexo2).

Figura10. Ricardo Martinez Realizando Pruebas Con La Aplicación en la de Minca.

Tabla 2.Resultados Pruebas Hechas Sin Micrófono Especializado

Ave	Acierto	Fallido
Gray-breasted Wood-Wren		X
Black-and-White Owl	X	
Keel-billed Toucan	X	
Black-and-White Owl		X
Laughing Falcon	X	
Rufous-tailed Jacamar		X
Barred Antshrike	X	
Yellow-backed Oriole	X	
Laughing Falcon		X
Great Kiskadee	X	
Cinnamon Becard	X	
Buff-throated Saltator		X
Black-crested Antshrike		X
Social Flycatcher	X	
Yellow-backed Oriole		X
Boat-billed Flycatcher	X	
Yellow Oriole	X	
Great Kiskadee		X
Barred Antshrike	X	
Gartered Trogon		X
Yellow-backed Oriole	X	
TOTAL	**12**	**9**
PORCENTAJE	**57.3%**	**42.7%**

Tabla 3.Resultados Pruebas Hechas Con Micrófono Especializado

Ave	Acierto	Fallido
Great Kiskadee	X	
Barred Antshrike	X	
Gartered Trogon		X
Yellow-backed Oriole	X	
Social Flycatcher	X	
Cinnamon Becard	X	
Yellow Oriole	X	
Laughing Falcon	X	
Black-and-White Owl	X	
Keel-billed Toucan	X	
Black-and-White Owl		X
Laughing Falcon	X	

Rufous-tailed Jacamar		X
Barred Antshrike	X	
Yellow-backed Oriole	X	
Laughing Falcon		X
Great Kiskadee	X	
Cinnamon Becard	X	
Buff-throated Saltator		X
Black-crested Antshrike		X
Social Flycatcher	X	
Yellow-backed Oriole		X
Boat-billed Flycatcher	X	
White-bearded Manakin		X
Yellow Oriole	X	
Cinnamon Becard	X	
Laughing Falcon		X
Great Kiskadee	X	
Social Flycatcher	X	
TOTAL	**19**	**10**
PORCENTAJE	**65.6%**	**34.4%**

Las pruebas fueron realizadas dos días, no se pudo comprobar con todos los especímenes disponibles para ser hallados con el aplicativo móvil, por lo que se trabajó con base en lo que vio y los datos que suministro el estudiante de biología Ricardo Martinez.

Cuando se compara la tabla 2 con la tabla 3 se aprecia que en la tabla 3 se encuentran más especímenes lo cual es producto de que el micrófono especializado logra captar sonidos de menor amplitud que los captados por el micrófono del teléfono celular, por lo que es recomendable equipar el celular con un micrófono de estos para garantizar un mejor funcionamiento del aplicativo.

Los resultados de la tabla 2 no son del todo satisfactorios, puesto que el nivel de asertividad fue muy bajo, pero era parte de los resultados esperados teniendo en cuenta el ruido ambiente y que el celular no puede estar muy cerca de las aves, debido a que estás se alejan cuando un grupo de personas se les acerca.

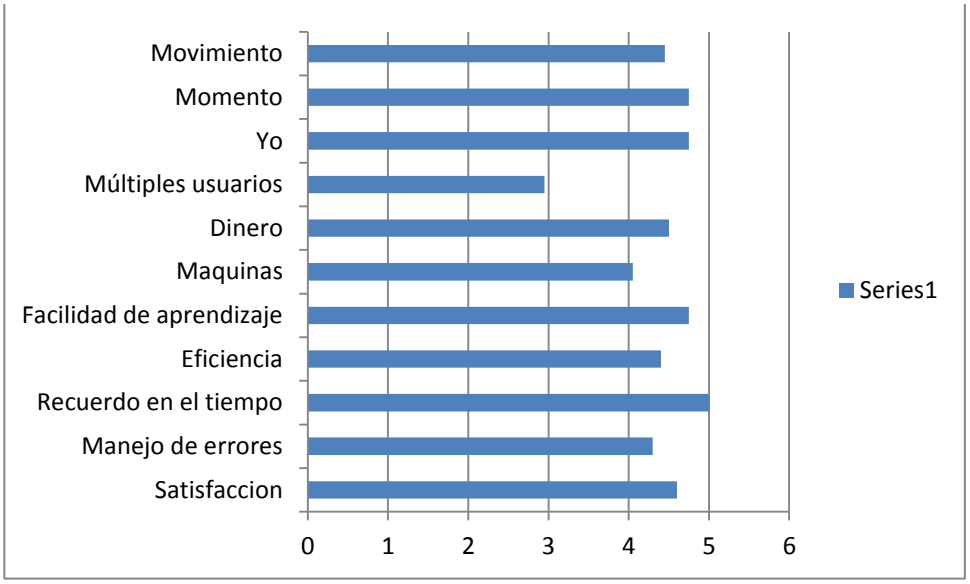
En la tabla 3 se puede ver que los resultados son cercanos a los obtenidos con las pruebas preliminares hechas en un entorno controlado las cuales se encuentran presentes en la tabla 1, con una ase0rtividad del 65.6%, con lo que se puede decir que el aplicativo con ayuda de un buen micrófono funciona de acuerdo a lo esperado según las pruebas preliminares.

5.3 PRUEBA DE USABILIDAD Y 6 M'S

Finalmente, la aplicación móvil fue evaluada a través del potencial de éxito de las 6M's y de la usabilidad. El potencial de éxito de las 6M's cuantificó seis atributos denominados movimiento (movement), momento (moment), yo (me), múltiples usuarios (multi-user), dinero (money) y máquinas (machines); mientras que la usabilidad cuantificó cinco atributos denominados facilidad de aprendizaje, eficiencia, recuerdo en el tiempo, manejo de errores y satisfacción. El instrumento aplicado en la evaluación incluyó dieciocho preguntas cerradas, y valoró el cumplimiento de cada atributo entre uno (muy en desacuerdo) y cinco (completamente de acuerdo).

La cuantificación se realizó con una muestra de veinte personas con conocimientos en la utilización e implementación de tecnologías, de los cuales todos fueron estudiantes de semestres superiores de ingeniería electrónica. Este proceso consistió en la interacción de los veinte usuarios con la aplicación móvil en un escenario controlado, haciendo uso de sus teléfonos celulares de modelos muy diversos, desde cualquier lugar donde estas

personas se encontraran; en donde cada persona a través del instrumento de evaluación, le asignó una puntuación entre uno y cinco a cada atributo, de acuerdo con el nivel de cumplimiento que experimentó, cuando interactuó con la aplicación. En la siguiente imagen se puede ver el valor promedio para cada atributo.



2 Grafica que Arrojan los Resultados de la Encuesta (Anexo 3)

De acuerdo con los datos obtenidos por esta evaluación se puede decir que la interfaz se ajustó a los requerimientos necesarios para la identificación de las aves (atributo: yo); le permitió a las personas movilizarse por muchas zonas sin comprometer el correcto funcionamiento de la aplicación (atributo: movimiento); se le permite al usuario verificar si el ave que la aplicación identifico realmente es esa y de este modo controlar los errores de funcionamiento (atributo: manejo de errores); el no uso de licencias y ser una alternativa para aquellas personas que no tienen los recursos necesarios para pagar un guía especializado (atributo: dinero); cada vez que el usuario requiera identificar un ave lo podrá hacer, con la limitante de la base de datos (atributo: momento); es intuitiva y la claridad en la información presentada en cada pantalla permite que el operario se familiarice rápidamente con la aplicación (atributo: facilidad de aprendizaje); estuvo sujeta la cantidad de ruido ambiente que tenían mientras reproducían los sonidos de las aves (atributo: Eficiencia); cumplió con las expectativas de los usuarios (atributo: satisfacción); puede ser usada desde la mayoría de dispositivos Android (atributo: máquinas); es intuitiva y muy mecánica en sus funciones (atributo: recuerdo en el tiempo); no cuenta con un módulo donde los diferentes usuarios de la aplicación puedan compartir entre ellos directamente, pero la posibilidad de compartir en twitter palea un poco está falencia (atributo: múltiples usuarios).

CAPÍTULO 6 RECURSOS

6.1 TABLAS DE PRESUPUESTOS

6.1.1. Tabla de Presupuesto Global

RUBROS GENERALES		RECURSOS APORTADOS POR: UNIVERSIDAD DEL MAGDALENA EFECTIVO	UNIVERSIDAD DEL MAGDALENA (CAPACIDAD INSTALADA)	OTRAS FUENTES	TOTAL
Personal:			6.500.000	500000	7.000.000
Insumos:				350000	350.000
Otros insumos:					0
Equipo	Compra			3.960.000	3.960.000
	Arriendo				0
	Uso				0
Servicios técnicos:					0
Salidas de campo:				300.000	300.000
Viajes Nacionales, Internacionales y Cursos de entrenamiento:					0
Software:					0
Realización talleres, foros:					0
Material bibliográfico especializado:			246,304.87		246,304.87
Publicaciones y patentes:				100.000	100.000
Imprevistos				150.000	150.000
TOTAL			**6.746.304,87**	**5.360.000**	**12.106.304,87**

6.1.2. Descripción del Rubro Personal (en $)							
NOMBRE	**Formación Académica**	**Función y Actividades a Desarrollar Dentro del Proyecto**	**Dedicación en Horas por Semanas**	**RECURSOS APORTADOS POR:**			**TOTAL**
				UNIVERSIDAD DEL MAGDALENA		**Otras fuentes ***	
				Efectivo	**Capacidad Instalada**		
David Acosta Peña	ESTUDIANTE	INVESTIGADOR	8			800000	800000
Brayan Hernández Romanos	ESTUDIANTE	INVESTIGADOR	8			800000	800000
Ricardo Martínez	ESTUDIANTE	INVESTIGADOR	4			400000	400000
Wilson Flores	INGENIERO	INVESTIGADOR	4			500000	500000
Luis L. Camargo Ariza	MAGISTER	TUTOR	4		4500000		4500000
TOTAL							**7000000**
Ricardo Martínez Estudiante de X semestre de Biología de la universidad del magdalena experto en el área de avistamiento de aves.							

6.1.3. Descripción del Rubro Insumos (en $)					
Nombre de los Insumos que se Planean Adquirir	**Justificación**	**RECURSOS APORTADOS POR:**			**Total**
		UNIVERSIDAD DEL MAGDALENA		**Otras fuentes***	
		Efectivo	**Capacidad Instalada**		
Papelería	Impresión de documentos			100000	100000
Uso de internet	Descarga de software e información			150000	150000
Otros				100000	100000
TOTAL					**350000**

6.1.4. Descripción del Rubro Equipos - Compras (en $)					
Nombre de los Equipos que se planean Adquirir	**Justificación**	**RECURSOS APORTADOS POR:**			**TOTAL**
		UNIVERSIDAD DEL MAGDALENA		**Otras Fuentes***	
		Efectivo	**Capacidad Instalada**		
Computadores	**Para la programación**			**3500000**	**3500000**
Celular gama media	**Pruebas de la aplicación**			**460000**	**460000**
TOTAL					**3960000**

6.1.5. Descripción del Rubro Salidas de Campo (en $)						
Descripción de los Ítem	**No. de Salidas de Campo a Realizar**	**Costo Promedio Unitario**	**RECURSOS APORTADOS POR:**			**TOTAL**
			UNIVERSIDAD DEL MAGDALENA		**Otras Fuentes***	
			Efectivo	**Capacidad Instalada**		
Transporte Intermunicipal	4	60000			240000	240000
Transporte Intermunicipal						
elementos de aseo						
Alimentación	4	15000			60000	60000
Alojamiento						
Otros Gastos Menores						
TOTAL						**300000**

6.1.6. Descripción del Rubro Software (en $)					
Nombre de los Software que se planean Adquirir	**Justificación**	**RECURSOS APORTADOS POR:**			**TOTAL**
		UNIVERSIDAD DEL MAGDALENA		**Otras Fuentes***	
		Efectivo	**Capacidad Instalada**		
Eclipse	Software libre				**0**
SDK de Android	Software libre				**0**
Java Oracle	Software libre				**0**
Ubuntu 14.04 OS	Software libre				**0**
Audacity	Software libre				**0**
Genymotion	Software libre				**0**
TOTAL					**0**

6.1.7. Descripción del Rubro Bibliografía (en $)					
Nombre de los libros, revistas, artículos y demás que se utilizaran de la Capacidad Instalada de la Universidad y/o nombre de los libros, revistas, artículos y demás que se compraran en el desarrollo del proyecto	**Justificación**	**RECURSOS APORTADOS POR:**			**TOTAL**
		UNIVERSIDAD DEL MAGDALENA		**Otras Fuentes***	
		Efectivo	**Capacidad Instalada**		
El gran libro de Android	**conocimientos básicos de programación Android**		**246,304.87**		**246,304.87**
TOTAL					**246,304.87**

CAPÍTULO 7. RECOMENDACIONES

- Se recomienda el uso de un micrófono especializado para eliminación de ruido y así obtener mayor eficiencia en la utilización de la aplicación.
- Ampliación de la cantidad de aves presentes en la aplicación, logrando de este modo que la aplicación pueda usarse en casi cualquier lugar del planeta.
- La utilización de filtros adaptativos mejorando la precisión al identificar cada ave.
- Añadir una función donde las distintas personas que usan la aplicación puedan interactuar entre sí, como por ejemplo un chat.

CONCLUSIONES

Los resultados obtenidos durante la prueba de asertividad de la aplicación móvil a la hora de identificar las aves presentes dentro de la base de datos del aplicativo móvil demostraron que la aplicación es funcional pero que se puede mejorar aún más optimizando la eficiencia de la misma y disminuyendo la brecha del error.

La aplicación móvil para la identificación de las aves a través del canto no busca reemplazar a los guías expertos, si no servir de apoyo para que estos tengan un respaldo en caso de no estar seguros de que ave es la que se encuentra presente en su entorno. También es muy útil para aquellas personas inexpertas que no cuentan con el dinero suficiente para pagar un guía especializado pero cuentan con un teléfono móvil con sistema operativo Android y les gusta salir a fotografiar aves.

Al ser una aplicación en un entorno desconectado, es decir, que ninguna de sus funciones dependen de internet, está puede ser usado en cualquier lugar garantizando de este modo que las personas puedan identificar aves incluso en zonas muy alejadas de la civilización, que es por lo general donde se realizan estas actividades referente al avistamiento de aves.

Los resultados estadísticos de evaluar los atributos de las 6M’s y la usabilidad, en la forma como se hizo, sustentan que la aplicación móvil es fácil de manipular, es atractiva al usuario, es eficiente y presenta un manejo adecuado de errores, garantizando la usabilidad de la misma; además, que la aplicación tiene potencial de éxito para ser implementada a gran escala, teniendo en cuenta que él aspecto de múltiples usuarios debe ser mejorado.

REFERENCIAS

[1] Clements (2014). Disponible en: http://www.birds.cornell.edu/clementschecklist/2014overview/.

[2] Bosques pre-montanos y montanos de la Sierra Nevada de Santa Marta. Disponible en:http://www.alpec.org/aves-montanas.html.

[3] E-bird y Xenocanto, Beckers J, Florez P (2013). Listado actualizado de las aves endémicas y casi-endémicas de Colombia.

[4] El correo del sol. (2017, Mayo 10). El correo del sol [online]. Disponible en: http://www.elcorreodelsol.com/articulo/presentan-una-app-gratuita-para-conocer-y-escuchar-las-aves-espanolas-en-el-movil

[5] Redacción tecnosfera. (2017, Mayo 10). En Colombia hay 14.4 millones de usuarios de "Smartphone". El tiempo [online]. Disponible en: http://www.eltiempo.com/archivo/documento/CMS-15066597

[6] Mobile: el mundo digital ya no es el mismo. Disponible en:http://www.google.com/intl/es-419/think/collections/mobile-collection- latam2014.html.

[7] Budney F y Grotke R, (2009). Taller De Bioacústica: Técnicas Para La Grabación De Sonidos Y Sus Aplicaciones.

[8] aplicación para smartphones permite identificar a las aves según su canto. Visto: 15-03-2015.Disponible en: http://www.latercera.com/noticia/tendencias/2014/10/659602407-9-aplicacion-para- smartphones-permite-identificar-a-las-aves-segun-sucanto.shtml 15-03-2015.

[9] Desarrollan un estudio para la identificación de aves a través de su canto. Disponible en:http://www.dicyt.com/noticias/desarrollan-un-estudio-para-la- identificación-deaves-a-traves-de-su-canto 15-03-2015.

[10] Clasificación de las aves. Disponible en: http://afaanimalesvertebrados.com/Proyecto/Proyecto/Clasificaacion.html.

[11] Hilty, S. L. Y Brown, W. L. 2001. A guide to the birds of Colombia.

[12] CLASIFICACIÓN DE LAS AVES. Disponible en: http://datateca.unad.edu.co/contenidos/30173/Contenido_en_EXE_30173/MODULO%2 0 CURSO%20SANIDAD%20AGROFORESTAL%2030173/clasificacin_de_las_aves.htm l.

[13] El canto de las aves. Disponible en: http://www.oya-es.net/reportajes/canto.html.

[14] ¿Qué es Android? Disponible en: http://www.xatakandroid.com/sistemaoperativo/que-es-

android.

[15] Desarrolle aplicaciones Andorid en Eclipse. Disponible en: http://www.ibm.com/developerworks/ssa/opensource/tutorials/os-eclipse-android/.

[16] Teléfonos inteligentes Disponible en: http://www.gcfaprendelibre.org/tecnologia/curso/informatica_basica/empezando_a_usar _un_computador/6.do

[17] "Modelo Vista Controlador" disponible en: http://www.neleste.com/modelo-vista-controlador/

[18] “Patrón Modelo-Vista-Controlador” disponible en:
http://s3.amazonaws.com/academia.edu.documents/38870522/15-42-2-PB.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1494015745&Signature=tUUgsXi5jujv%2F4fSLgPHCM8d3xw%3D&response-content-disposition=inline%3B%20filename%3DPatron_Modelo-Vista-Controlador.pdf
[19] “Metodología para el desarrollo de aplicaciones móviles” Disponible en:
http://www.scielo.org.co/pdf/tecn/v18n40/v18n40a03.pdf

[20] Audio Spectrum Analizer Disponible en: https://github.com/bewantbe/audio-analyzer-for-android

ANEXOS

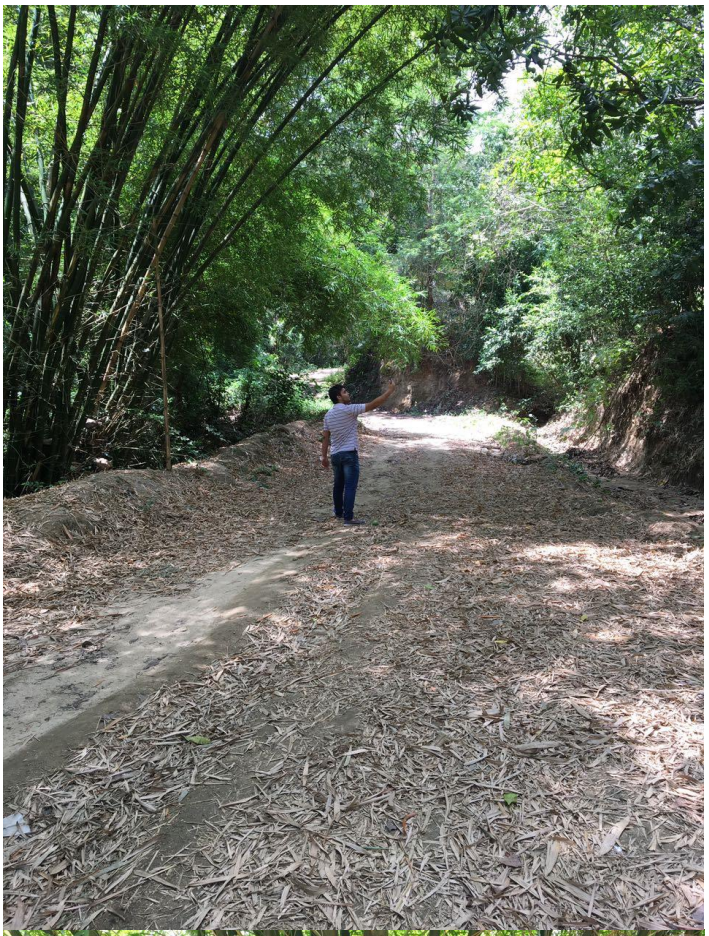
Anexo1.ANDROIDMANIFEST

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="github.bewantbe.audio_analyzer_for_android"
    android:versionCode="20170218"
    android:versionName="1.4"
    android:installLocation="auto" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="23" />

    <uses-permission android:name="android.permission.RECORD_AUDIO" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.VIBRATE" />
    <uses-permission-sdk-23 android:name="android.permission.RECORD_AUDIO" />
    <uses-permission-sdk-23 android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission-sdk-23 android:name="android.permission.VIBRATE" />

    <application
        android:allowBackup="true"
        android:fullBackupContent="@xml/backup_descriptor"
        android:icon="@drawable/icon"
        android:label="@string/app_name"
        android:theme="@style/DarkThemeSelector" >
        <activity
            android:name="github.bewantbe.audio_analyzer_for_android.AnalyzerActivity"
            android:label="@string/app_name" >

        </activity>
        <activity
            android:name=".MyPreferences"
            android:label="Preferences" >
        </activity>
        <activity
            android:name=".InfoRecActivity"
            android:label="@string/title_activity_info_rec"
            android:parentActivityName="github.bewantbe.audio_analyzer_for_android.AnalyzerActivity" >
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value="github.bewantbe.audio_analyzer_for_android.AnalyzerActivity" />
        </activity>

        <activity
        android:name=".Descri_ave"
        android:label="@string/title_activity_descri_ave">

         </activity>
        <activity
        android:name=".Lis_aves"
        android:label="@string/title_activity_lista_aves">

         </activity>
        <activity
            android:name="github.bewantbe.audio_analyzer_for_android.Descripcion"
            android:label="@string/title_activity_Descripcion">

        </activity>
        <activity
            android:name=".Aves_vistas"
            android:label="@string/title_activity_Aves_vistas">

        </activity>


        <activity
            android:name=".Principal"
            android:label="@string/title_activity_ave1">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>

</manifest>
```

Anexo2.PRUEBAS DE FUNCIONAMIENTO DE LA APLICACION EN CAMPO (SECTOR MINCA-POZO AZUL)

Pruebas de la aplicacion sin Microfono especializado

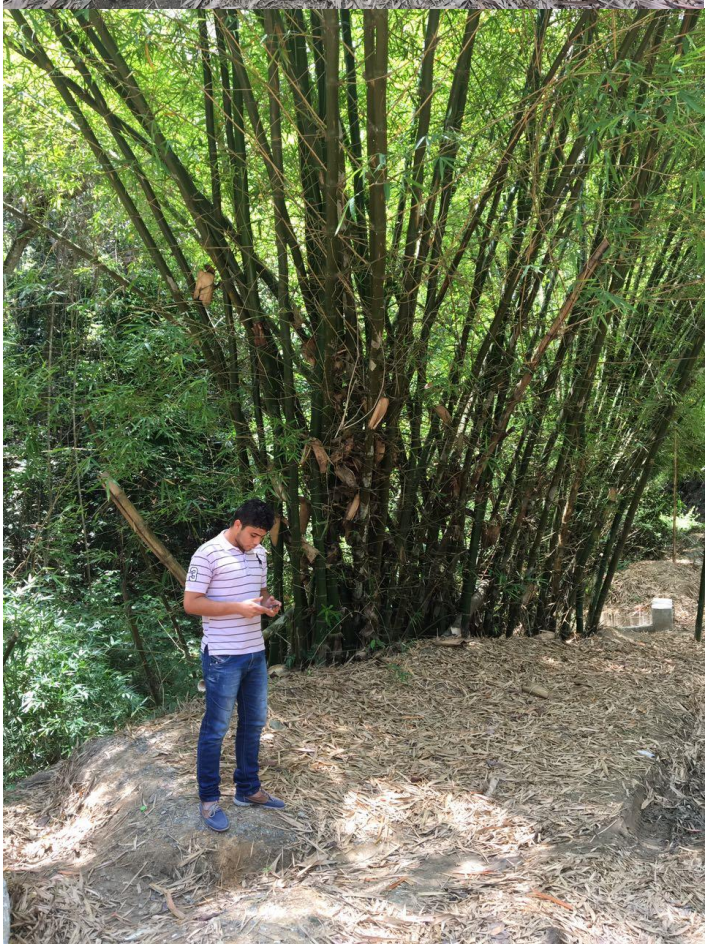
Cucarachero x
buho carinegro ✓
Tucan Pico verde ✓
buho carinegro x
Halcon Guaco ✓
Colibri Jacamar x
Barrado Batara ✓
Bolsero Dorso Dorado ✓
Halcon Guaco x
Grilton bichofue ✓
Copeton Batara ✓
Saltador esmedallado x
Copeton Batara x
Bienteveo Mediano ✓
Bolsero Dorso Dorado x
Bienteveo Mediano ✓
Turpial Amarillo ✓
Grilton bichofue x
Barrado Batara ✓
Trubon enligado x
Bolsero Dorso Dorado ✓



Pruebas de aplicacion con microfono especializado

Grilton bichofue ✓
Barrado Batara ✓
Tragon enligado x
Bolsero Dorso Dorado ✓
Bienteveo Mediano ✓
Copeton Batara ✓
Turpial Amarillo ✓
Halcon Guaco ✓
buho carinegro ✓
Tucan pico verde ✓
buho carinegro x
Halcon Guaco ✓
Colibri Jacamar x
Barrado Batara ✓
Bolsero Dorso Dorado ✓
Halcon Guaco x
Grilton Bichofue ✓
Copeton Batara ✓
Bolsero Dorso Dorado x
Saltador esmedallado x
Bienteveo Mediano ✓
Bolsero Dorso Dorado x
Bienteveo pitagua ✓
Frutero Negro x
Turpial Amarillo ✓
Copeton Batara x

Halcon Guaco ✓
Grilton bichofue ✓
Bienteveo Mediano ✓



Anexo3. Encuesta

Califique de uno (1) a cinco (5), siendo uno completamente insatisfecho y cinco completamente satisfecho a las siguientes preguntas:

1. ¿Es entendible el lenguaje utilizado por la aplicación?
2. ¿la secuencia de pantallas que se muestran llevan un orden lógico para que la aplicación cumpla con el objetivo?
3. ¿es coherente la información colocada en los diferentes botones con las acciones que realizan?
4. ¿se muestra solo la información necesaria para el correcto funcionamiento de la aplicación?
5. ¿los gráficos son coherentes con el objeto de la aplicación?
6. ¿se encuentran bien distribuidos los diferentes mensajes y botones dentro de la aplicación?
7. ¿cada título es coherente con la información mostrada en pantalla?
8. ¿es sencillo encontrar la información necesaria dentro de la aplicación?
9. ¿el manual de usuario brinda la información adecuada para el correcto manejo de la aplicación?
10. ¿brinda la aplicación en caso de mal funcionamiento una forma de salir de ese error?
11. ¿puede el usuario hacer confirmación en caso de que la aplicación muestre un dato errado?
12. ¿son fácil de aprender los pasos necesarios para que la aplicación cumpla con su objeto?
13. ¿le permite la interfaz de usuario cumplir con el objetivo de la aplicación?
14. ¿se puede mover de un lugar a otro sin comprometer la funcionalidad de la aplicación?
15. ¿se puede utilizar la aplicación en cualquier momento que el usuario lo desee?
16. ¿posibilita la aplicación la interacción entre diferentes usuarios?
17. ¿disminuye el costo la utilización de tecnologías de software libre?
18. ¿es posible que la aplicación funcione en la mayoría de dispositivos android?

Anexo4. ANALYZERACTIVITY

```
// TODO: Require the permission for android >=6.
// https://developer.android.com/training/permissions/requesting.html
// https://developer.android.com/guide/topics/permissions/requesting.html

public class AnalyzerActivity extends Activity
    implements OnLongClickListener, OnClickListener,
            OnItemClickListener, AnalyzerGraphic.Ready
{
  static final String TAG="AnalyzerActivity:";
  private Button deten1,lista;
  AnalyzerViews analyzerViews;
  SamplingLoop samplingThread;
  private RangeViewDialogC rangeViewDialogC;
  private GestureDetectorCompat mDetector;

  AnalyzerParameters analyzerParam = null;

  double dtRMS = 0;
  double dtRMSFromFT = 0;
  double maxAmpDB;
  double maxAmpFreq;



  private boolean isLinearFreq = true;
  private boolean isMeasure = false;
  volatile boolean bSaveWav = false;

  @Override
  public void onCreate(Bundle savedInstanceState) {
//  Debug.startMethodTracing("calc");
    final int maxMemory = (int) (Runtime.getRuntime().maxMemory() / 1024);
    Log.i(TAG, " max runtime mem = " + maxMemory + "k");

    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    Resources res = getResources();
    analyzerParam = new AnalyzerParameters(res);

    // Set and get preferences in PreferenceActivity
    PreferenceManager.setDefaultValues(this, R.xml.preferences, false);
    // Set variable according to the preferences
    loadPreferenceForView();

    analyzerViews = new AnalyzerViews(this);

    // travel Views, and attach ClickListener to the views that contain android:tag="select"
    visit((ViewGroup) analyzerViews.graphView.getRootView(), new Visit() {
      @Override
      public void exec(View view) {
        view.setOnLongClickListener(AnalyzerActivity.this);
        view.setOnClickListener(AnalyzerActivity.this);
        ((TextView) view).setFreezesText(true);
      }
    }, "select");

    rangeViewDialogC = new RangeViewDialogC(this, analyzerViews.graphView);

    mDetector = new GestureDetectorCompat(this, new AnalyzerGestureListener());
   deten1=(Button)findViewById(R.id.detener) ;
    lista=(Button)findViewById(R.id.base) ;


 deten1.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View v) {
     Intent mainIntent = new Intent().setClass(AnalyzerActivity.this, Principal.class);
      startActivity(mainIntent);
   }


 });
```

```
    lista.setOnClickListener(new View.OnClickListener() {
     @Override
     public void onClick(View v) {
       Intent ini = new Intent().setClass(AnalyzerActivity.this,Lis_aves.class);
       startActivity(ini);
     }
    });



  }
  public void abrirprue( String nrave){
    samplingThread.setPause(true);
    Intent mainIntent = new Intent().setClass(AnalyzerActivity.this, Descri_ave.class);

    mainIntent.putExtra("nr_ave",nrave);
    startActivity(mainIntent);

  }
}
```

Anexo5. ANALYZERVIEWS

```
class AnalyzerViews {
    final String TAG = "AnalyzerViews";
    private AnalyzerActivity activity;
    AnalyzerGraphic graphView;

    private float DPRatio;
    private float listItemTextSize = 20;        // see R.dimen.button_text_fontsize
    private float listItemTitleTextSize = 12;   // see R.dimen.button_text_small_fontsize

    private StringBuilder textCur = new StringBuilder("");  // for textCurChar
    private StringBuilder textRMS  = new StringBuilder("");
    private StringBuilder textPeak = new StringBuilder("");
    private StringBuilder textRec = new StringBuilder("");  // for textCurChar
    private char[] textRMSChar;   // for text in R.id.textview_RMS
    private char[] textCurChar;   // for text in R.id.textview_cur
    private char[] textPeakChar;  // for text in R.id.textview_peak
    private char[] textRecChar;   // for text in R.id.textview_rec

    PopupWindow popupMenuSampleRate;
    PopupWindow popupMenuFFTLen;
    PopupWindow popupMenuAverage;

    boolean bWarnOverrun = true;

    AnalyzerViews(AnalyzerActivity _activity) {
        activity = _activity;
        graphView = (AnalyzerGraphic) activity.findViewById(R.id.plot);

        Resources res = activity.getResources();
        listItemTextSize      = res.getDimension(R.dimen.button_text_fontsize);
        listItemTitleTextSize = res.getDimension(R.dimen.button_text_small_fontsize);
        DPRatio = res.getDisplayMetrics().density;

        textRMSChar  = new char[1];
        textCurChar  = new char[1];
        textRecChar  = new char[1];
        textPeakChar = new char[res.getString(R.string.textview_peak_text).length()];

        /// initialize pop up window items list
        // http://www.codeofaninja.com/2013/04/show-listview-as-drop-down-android.html
        popupMenuSampleRate = popupMenuCreate( AnalyzerUtil.validateAudioRates(
                res.getStringArray(R.array.sample_rates)), R.id.button_sample_rate);
        popupMenuFFTLen = popupMenuCreate(
                res.getStringArray(R.array.fft_len), R.id.button_fftlen);
        popupMenuAverage = popupMenuCreate(
                res.getStringArray(R.array.fft_ave_num), R.id.button_average);

        setTextViewFontSize();
    }

    // Set text font size of textview_cur and textview_peak
    // according to space left
    //@SuppressWarnings("deprecation")
    private void setTextViewFontSize() {
        TextView tv = (TextView) activity.findViewById(R.id.textview_cur);
        // At this point tv.getWidth(), tv.getLineCount() will return 0

        Display display = activity.getWindowManager().getDefaultDisplay();
        // pixels left
        float px = display.getWidth() -
activity.getResources().getDimension(R.dimen.textview_RMS_layout_width) - 5;

        float fs = tv.getTextSize();  // size in pixel

        // shrink font size if it can not fit in one line.
        final String text = activity.getString(R.string.textview_peak_text);
        // note: mTestPaint.measureText(text) do not scale like sp.
        Paint mTestPaint = new Paint();
        mTestPaint.setTextSize(fs);
        mTestPaint.setTypeface(Typeface.MONOSPACE);
        while (mTestPaint.measureText(text) > px && fs > 5) {
            fs -= 0.5;
            mTestPaint.setTextSize(fs);
        }

        ((TextView)
activity.findViewById(R.id.textview_cur)).setTextSize(TypedValue.COMPLEX_UNIT_PX, fs);
        ((TextView)
activity.findViewById(R.id.textview_peak)).setTextSize(TypedValue.COMPLEX_UNIT_PX, fs);
```

```
    }

    // Prepare the spectrum and spectrogram plot (from scratch or full reset)
    // Currently called by SamplingLoop::run(), maybe move to main thread?
    void setupView(AnalyzerParameters analyzerParam) {
        graphView.setupPlot(analyzerParam.sampleRate, analyzerParam.fftLen,
analyzerParam.timeDurationPref, analyzerParam.nFFTAverage);
    }

    // Will be called by SamplingLoop (in another thread)
    void update(final double[] spectrumDBcopy) {
        graphView.saveSpectrum(spectrumDBcopy);
        activity.runOnUiThread(new Runnable() {
            @Override
            public void run() {
                // data will get out of synchronize here
                invalidateGraphView();
            }
        });
    }

    private double wavSecOld = 0;       // used to reduce frame rate
    void updateRec(double wavSec) {
        if (wavSecOld > wavSec) {
            wavSecOld = wavSec;
        }
        if (wavSec - wavSecOld < 0.1) {
            return;
        }
        wavSecOld = wavSec;
        activity.runOnUiThread(new Runnable() {
            @Override
            public void run() {
                // data will get out of synchronize here
                invalidateGraphView(AnalyzerViews.VIEW_MASK_RecTimeLable);
            }
        });
    }

    void notifyWAVSaved(final String path) {
        activity.runOnUiThread(new Runnable() {
            @Override
            public void run() {
                Context context = activity.getApplicationContext();
                String text = "WAV saved to " + path;
                Toast toast = Toast.makeText(context, text, Toast.LENGTH_SHORT);
                toast.show();
            }
        });
    }

    private long lastTimeNotifyOverrun = 0;
    void notifyOverrun() {
        if (!bWarnOverrun) {
            return;
        }
        long t = SystemClock.uptimeMillis();
        if (t - lastTimeNotifyOverrun > 6000) {
            lastTimeNotifyOverrun = t;
            activity.runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    Context context = activity.getApplicationContext();
                    String text = "Recorder buffer overrun!\nYour cell phone is too slow.\nTry lower
sampling rate or higher average number.";
                    Toast toast = Toast.makeText(context, text, Toast.LENGTH_LONG);
                    toast.show();
                }
            });
        }
    }

    void showInstructions() {
        TextView tv = new TextView(activity);
        tv.setMovementMethod(new ScrollingMovementMethod());
        tv.setText(Html.fromHtml(activity.getString(R.string.instructions_text)));
        new AlertDialog.Builder(activity)
                .setTitle(R.string.instructions_title)
                .setView(tv)
                .setNegativeButton(R.string.dismiss, null)
                .create().show();
    }
```

```
    void enableSaveWavView(boolean bSaveWav) {
        if (bSaveWav) {
            ((TextView) activity.findViewById(R.id.textview_rec)).setHeight((int)(19*DPRatio));
        } else {
            ((TextView) activity.findViewById(R.id.textview_rec)).setHeight((int)(0*DPRatio));
        }
    }

    @SuppressWarnings("deprecation")
    void showPopupMenu(View view) {
        // popup menu position
        // In API 19, we can use showAsDropDown(View anchor, int xoff, int yoff, int gravity)
        // The problem in showAsDropDown (View anchor, int xoff, int yoff) is
        // it may show the window in wrong direction (so that we can't see it)
        int[] wl = new int[2];
        view.getLocationInWindow(wl);
        int x_left = wl[0];
        int y_bottom = activity.getWindowManager().getDefaultDisplay().getHeight() - wl[1];
        int gravity = android.view.Gravity.START | android.view.Gravity.BOTTOM;

        switch (view.getId()) {
            case R.id.button_sample_rate:
                popupMenuSampleRate.showAtLocation(view, gravity, x_left, y_bottom);
//                popupMenuSampleRate.showAsDropDown(view, 0, 0);
                break;
            case R.id.button_fftlen:
                popupMenuFFTLen.showAtLocation(view, gravity, x_left, y_bottom);
//                popupMenuFFTLen.showAsDropDown(view, 0, 0);
                break;
            case R.id.button_average:
                popupMenuAverage.showAtLocation(view, gravity, x_left, y_bottom);
//                popupMenuAverage.showAsDropDown(view, 0, 0);
                break;
        }
    }

    // Maybe put this PopupWindow into a class
    private PopupWindow popupMenuCreate(String[] popUpContents, int resId) {

        // initialize a pop up window type
        PopupWindow popupWindow = new PopupWindow(activity);

        // the drop down list is a list view
        ListView listView = new ListView(activity);

        // set our adapter and pass our pop up window contents
        ArrayAdapter<String> aa = popupMenuAdapter(popUpContents);
        listView.setAdapter(aa);

        // set the item click listener
        listView.setOnItemClickListener(activity);

        listView.setTag(resId);  // button res ID, so we can trace back which button is pressed

        // get max text width
        Paint mTestPaint = new Paint();
        mTestPaint.setTextSize(listItemTextSize);
        float w = 0;
        float wi;       // max text width in pixel
        for (String popUpContent : popUpContents) {
            String sts[] = popUpContent.split("::");
            String st = sts[0];
            if (sts.length == 2 && sts[1].equals("0")) {
                mTestPaint.setTextSize(listItemTitleTextSize);
                wi = mTestPaint.measureText(st);
                mTestPaint.setTextSize(listItemTextSize);
            } else {
                wi = mTestPaint.measureText(st);
            }
            if (w < wi) {
                w = wi;
            }
        }

        // left and right padding, at least +7, or the whole app will stop respond, don't know why
        w = w + 20 * DPRatio;
        if (w < 60) {
            w = 60;
        }

        // some other visual settings
```

```
        popupWindow.setFocusable(true);
        popupWindow.setHeight(WindowManager.LayoutParams.WRAP_CONTENT);
        // Set window width according to max text width
        popupWindow.setWidth((int)w);
        // also set button width
        ((Button) activity.findViewById(resId)).setWidth((int)(w + 4 * DPRatio));
        // Set the text on button in loadPreferenceForView()

        // set the list view as pop up window content
        popupWindow.setContentView(listView);

        return popupWindow;
    }

    /*
     * adapter where the list values will be set
     */
    private ArrayAdapter<String> popupMenuAdapter(String itemTagArray[]) {
        return new ArrayAdapter<String>(activity, android.R.layout.simple_list_item_1, itemTagArray)
{
            @NonNull
            @Override
            public View getView(int position, View convertView, @NonNull ViewGroup parent) {
                // setting the ID and text for every items in the list
                String item = getItem(position);
                String[] itemArr = item.split("::");
                String text = itemArr[0];
                String id = itemArr[1];

                // visual settings for the list item
                TextView listItem = new TextView(activity);

                if (id.equals("0")) {
                    listItem.setText(text);
                    listItem.setTag(id);
                    listItem.setTextSize(listItemTitleTextSize / DPRatio);
                    listItem.setPadding(5, 5, 5, 5);
                    listItem.setTextColor(Color.GREEN);
                    listItem.setGravity(android.view.Gravity.CENTER);
                } else {
                    listItem.setText(text);
                    listItem.setTag(id);
                    listItem.setTextSize(listItemTextSize / DPRatio);
                    listItem.setPadding(5, 5, 5, 5);
                    listItem.setTextColor(Color.WHITE);
                    listItem.setGravity(android.view.Gravity.CENTER);
                }

                return listItem;
            }
        };
    }

    private void refreshCursorLabel() {
        double f1 = graphView.getCursorFreq();

        textCur.setLength(0);
        textCur.append("Cur :");
        SBNumFormat.fillInNumFixedWidthPositive(textCur, f1, 5, 1);
        textCur.append("Hz(");
        AnalyzerUtil.freq2Cent(textCur, f1, " ");
        textCur.append(") ");
        SBNumFormat.fillInNumFixedWidth(textCur, graphView.getCursorDB(), 3, 1);
        textCur.append("dB");
        textCur.getChars(0, Math.min(textCur.length(), textCurChar.length), textCurChar, 0);

        ((TextView) activity.findViewById(R.id.textview_cur))
                .setText(textCurChar, 0, Math.min(textCur.length(), textCurChar.length));
    }

    private void refreshRMSLabel(double dtRMSFromFT) {
        textRMS.setLength(0);
        textRMS.append("RMS:dB \n");
        SBNumFormat.fillInNumFixedWidth(textRMS, 20*Math.log10(dtRMSFromFT), 3, 1);
        textRMS.getChars(0, Math.min(textRMS.length(), textRMSChar.length), textRMSChar, 0);

        TextView tv = (TextView) activity.findViewById(R.id.textview_RMS);
        tv.setText(textRMSChar, 0, textRMSChar.length);
        tv.invalidate();
    }

    private void refreshPeakLabel(double maxAmpFreq, double maxAmpDB, String mensaje) {
```

```
        textPeak.setLength(0);
        textPeak.append(mensaje);
        SBNumFormat.fillInNumFixedWidthPositive(textPeak, maxAmpFreq, 5, 1);
        textPeak.append("Hz(");
        AnalyzerUtil.freq2Cent(textPeak, maxAmpFreq, " ");
        textPeak.append(") ");
        SBNumFormat.fillInNumFixedWidth(textPeak, maxAmpDB, 3, 1);
        textPeak.append("dB");
        textPeak.getChars(0, Math.min(textPeak.length(), textPeakChar.length), textPeakChar, 0);

        TextView tv = (TextView) activity.findViewById(R.id.textview_peak);
        tv.setText(textPeakChar, 0, textPeakChar.length);
        tv.invalidate();
    }

    private void refreshRecTimeLable(double wavSec, double wavSecRemain) {
        // consist with @string/textview_rec_text
        textRec.setLength(0);
        textRec.append("Rec: ");
        SBNumFormat.fillTime(textRec, wavSec, 1);
        textRec.append(", Remain: ");
        SBNumFormat.fillTime(textRec, wavSecRemain, 0);
        textRec.getChars(0, Math.min(textRec.length(), textRecChar.length), textRecChar, 0);
        ((TextView) activity.findViewById(R.id.textview_rec))
                .setText(textRecChar, 0, Math.min(textRec.length(), textRecChar.length));
    }

    private long timeToUpdate = SystemClock.uptimeMillis();
    private volatile boolean isInvalidating = false;

    // Invalidate graphView in a limited frame rate
    void invalidateGraphView() {
        invalidateGraphView(-1);
    }

    private static final int VIEW_MASK_graphView     = 1<<0;
    private static final int VIEW_MASK_textview_RMS  = 1<<1;
    private static final int VIEW_MASK_textview_peak = 1<<2;
    private static final int VIEW_MASK_CursorLabel   = 1<<3;
    private static final int VIEW_MASK_RecTimeLable  = 1<<4;

    private void invalidateGraphView(int viewMask) {
        if (isInvalidating) {
            return ;
        }
        isInvalidating = true;
        long frameTime;                         // time delay for next frame
        if (graphView.getShowMode() != AnalyzerGraphic.PlotMode.SPECTRUM) {
            frameTime = 1000/8;  // use a much lower frame rate for spectrogram
        } else {
            frameTime = 1000/60;
        }
        long t = SystemClock.uptimeMillis();
        //  && !graphView.isBusy()
        if (t >= timeToUpdate) {    // limit frame rate
            timeToUpdate += frameTime;
            if (timeToUpdate < t) {             // catch up current time
                timeToUpdate = t+frameTime;
            }
            idPaddingInvalidate = false;
            // Take care of synchronization of graphView.spectrogramColors and spectrogramColorsPt,
            // and then just do invalidate() here.
            if ((viewMask & VIEW_MASK_graphView) != 0)
                graphView.invalidate();
            // RMS
            if ((viewMask & VIEW_MASK_textview_RMS) != 0)
                refreshRMSLabel(activity.dtRMSFromFT);
            // peak frequency
            if ((viewMask & VIEW_MASK_textview_peak) != 0) {
                String mensaje1 = "F_Pico";



                if(activity.maxAmpFreq>=533 && activity.maxAmpFreq<=645)
                {
                    mensaje1 = "1_Black-and-White Owl ";
                    refreshPeakLabel(activity.maxAmpFreq, activity.maxAmpDB, mensaje1);

                    activity.abrirprue("1");
```

```
}
else
if(activity.maxAmpFreq>=1330&& activity.maxAmpFreq<=1350)
{
    mensaje1 = "2_Gartered Trogon";
    refreshPeakLabel(activity.maxAmpFreq, activity.maxAmpDB, mensaje1);
    activity.abrirprue("2");

}
else
if(activity.maxAmpFreq>=1420 && activity.maxAmpFreq<=1485 )
{
    mensaje1 = "3_Laughing Falcon";
    refreshPeakLabel(activity.maxAmpFreq, activity.maxAmpDB, mensaje1);
    activity.abrirprue("3");


}
else
if(activity.maxAmpFreq>=1630 && activity.maxAmpFreq<=1680 )
{
    mensaje1 = "4_Keel-billed Toucan";
    refreshPeakLabel(activity.maxAmpFreq, activity.maxAmpDB, mensaje1);
    activity.abrirprue("4");


}
else
if(activity.maxAmpFreq>=1830 && activity.maxAmpFreq<=1890 )
{
    mensaje1 = "5_Barred Antshrike ";
    refreshPeakLabel(activity.maxAmpFreq, activity.maxAmpDB, mensaje1);
    activity.abrirprue("5");


}
else
if(activity.maxAmpFreq>=1950 && activity.maxAmpFreq<=2000 )
{
    mensaje1 = "6_Black-crested Antshrike ";
    refreshPeakLabel(activity.maxAmpFreq, activity.maxAmpDB, mensaje1);
    activity.abrirprue("6");


}
else
if(activity.maxAmpFreq>=2256 && activity.maxAmpFreq<=2262)
{
    mensaje1 = "7_Yellow-backed Oriole ";
    refreshPeakLabel(activity.maxAmpFreq, activity.maxAmpDB, mensaje1);
    activity.abrirprue("7");


}
else
if(activity.maxAmpFreq>=2417 && activity.maxAmpFreq<=2423)
{
    mensaje1 = "8_Gray-breasted Wood-Wren ";
    refreshPeakLabel(activity.maxAmpFreq, activity.maxAmpDB, mensaje1);
    activity.abrirprue("8");


}
else
if(activity.maxAmpFreq>=2724 && activity.maxAmpFreq<=2730 )
{
    mensaje1 = "9_White-bearded Manakin ";
    refreshPeakLabel(activity.maxAmpFreq, activity.maxAmpDB, mensaje1);
    activity.abrirprue("9");


}
else
if(activity.maxAmpFreq>=2795 && activity.maxAmpFreq<=2801 )
{
    mensaje1 = "10_Buff-throated Saltator  ";
    refreshPeakLabel(activity.maxAmpFreq, activity.maxAmpDB, mensaje1);
    activity.abrirprue("10");


}
else
```

```
if(activity.maxAmpFreq>=2829 && activity.maxAmpFreq<=2835 )
{
    mensaje1 = "11_Rufous-tailed Jacamar";
    refreshPeakLabel(activity.maxAmpFreq, activity.maxAmpDB, mensaje1);
    activity.abrirprue("11");


}
else
if(activity.maxAmpFreq>=2870 && activity.maxAmpFreq<=2876 )
{
    mensaje1 = "12_White-lined Tanager ";
    refreshPeakLabel(activity.maxAmpFreq, activity.maxAmpDB, mensaje1);
    activity.abrirprue("12");


}
else
if(activity.maxAmpFreq>=3017 && activity.maxAmpFreq<=3023 )
{
    mensaje1 = "13_Streaked Saltator  ";
    refreshPeakLabel(activity.maxAmpFreq, activity.maxAmpDB, mensaje1);
    activity.abrirprue("13");


}
else
if(activity.maxAmpFreq>=3033&& activity.maxAmpFreq<=3039 )
{
    mensaje1 = "14_Great Kiskadee ";
    refreshPeakLabel(activity.maxAmpFreq, activity.maxAmpDB, mensaje1);
    activity.abrirprue("14");


}
else
//
if(activity.maxAmpFreq>=3093 && activity.maxAmpFreq<=3099 )
{
    mensaje1 = "15_Yellow Oriole ";
    refreshPeakLabel(activity.maxAmpFreq, activity.maxAmpDB, mensaje1);
    activity.abrirprue("15");


}
else

if(activity.maxAmpFreq>=3292 && activity.maxAmpFreq<=3298 )
{
    mensaje1 = "16_Cinnamon Becard  ";
    refreshPeakLabel(activity.maxAmpFreq, activity.maxAmpDB, mensaje1);
    activity.abrirprue("16");


}
else
if(activity.maxAmpFreq>=3440 && activity.maxAmpFreq<=3446 )
{
    mensaje1 = "17_Boat-billed Flycatcher  ";
    refreshPeakLabel(activity.maxAmpFreq, activity.maxAmpDB, mensaje1);
    activity.abrirprue("17");


}
else

if(activity.maxAmpFreq>=3762 && activity.maxAmpFreq<=3768 )
{
    mensaje1 = "18_Crimson-backed Tanager  ";
    refreshPeakLabel(activity.maxAmpFreq, activity.maxAmpDB, mensaje1);
    activity.abrirprue("18");


}
else
if(activity.maxAmpFreq>=3875 && activity.maxAmpFreq<=3881 )
{
    mensaje1 = "19_Yellow-bellied Seedeater";
    refreshPeakLabel(activity.maxAmpFreq, activity.maxAmpDB, mensaje1);
    activity.abrirprue("19");
```

```
}
else
if(activity.maxAmpFreq>=4090 && activity.maxAmpFreq<=4096 )
{
    mensaje1 = "20_Collared Aracari";
    refreshPeakLabel(activity.maxAmpFreq, activity.maxAmpDB, mensaje1);
    activity.abrirprue("20");


}
else
if(activity.maxAmpFreq>=4143 && activity.maxAmpFreq<=4149 )
{
    mensaje1 = "21_Streaked Flycatcher";
    refreshPeakLabel(activity.maxAmpFreq, activity.maxAmpDB, mensaje1);
    activity.abrirprue("21");


}
else
if(activity.maxAmpFreq>=4604 && activity.maxAmpFreq<=4610 )
{
    mensaje1 = "22_Ochre-bellied Flycatcher";
    refreshPeakLabel(activity.maxAmpFreq, activity.maxAmpDB, mensaje1);
    activity.abrirprue("22");


}
else
if(activity.maxAmpFreq>=4718 && activity.maxAmpFreq<=4724 )
{
    mensaje1 = "23_Blue-black Grassquit";
    refreshPeakLabel(activity.maxAmpFreq, activity.maxAmpDB, mensaje1);
    activity.abrirprue("23");


}
else
if(activity.maxAmpFreq>=4770 && activity.maxAmpFreq<=4776)
{
    mensaje1 = "24_Orange-chinned Parakeet ";
    refreshPeakLabel(activity.maxAmpFreq, activity.maxAmpDB, mensaje1);
    activity.abrirprue("24");


}
else
if(activity.maxAmpFreq>=5447 && activity.maxAmpFreq<=5453)
{
    mensaje1 = "25_American Redstart ";
    refreshPeakLabel(activity.maxAmpFreq, activity.maxAmpDB, mensaje1);
    activity.abrirprue("25");


}
else
if(activity.maxAmpFreq>=5896 && activity.maxAmpFreq<=5902)
{
    mensaje1 = "26_Palm Tanager ";
    refreshPeakLabel(activity.maxAmpFreq, activity.maxAmpDB, mensaje1);
    activity.abrirprue("26");


}
 else
if(activity.maxAmpFreq>=6040 && activity.maxAmpFreq<=6055)
{
    mensaje1 = "27_Social Flycatcher ";
    refreshPeakLabel(activity.maxAmpFreq, activity.maxAmpDB, mensaje1);
    activity.samplingThread.setPause(true);


}
else
if(activity.maxAmpFreq>=6720&& activity.maxAmpFreq<=6726)
{
    mensaje1 = "28_Bay-headed Tanager";
    refreshPeakLabel(activity.maxAmpFreq, activity.maxAmpDB, mensaje1);
    activity.abrirprue("28");


}
```

```
                else
                if(activity.maxAmpFreq>=9190 && activity.maxAmpFreq<=9230)
                {
                    mensaje1 = "29_Swallow Tanager";
                    refreshPeakLabel(activity.maxAmpFreq, activity.maxAmpDB, mensaje1);
                    activity.abrirprue("29");


                }
                else
                if(activity.maxAmpFreq>=8200 && activity.maxAmpFreq<=8250)
                {
                    mensaje1 = "30_ Scaled Piculet";
                    refreshPeakLabel(activity.maxAmpFreq, activity.maxAmpDB, mensaje1);
                    activity.abrirprue("30");


                }
                else
                {

                   refreshPeakLabel(activity.maxAmpFreq, activity.maxAmpDB, mensaje1);
                   Log.i(TAG, "frecuencia" + activity.maxAmpDB );

                }




            }
            if ((viewMask & VIEW_MASK_CursorLabel) != 0)
                refreshCursorLabel();
            if ((viewMask & VIEW_MASK_RecTimeLable) != 0)
                refreshRecTimeLable(activity.samplingThread.wavSec,
activity.samplingThread.wavSecRemain);
        } else {
            if (! idPaddingInvalidate) {
                idPaddingInvalidate = true;
                paddingViewMask = viewMask;
                paddingInvalidateHandler.postDelayed(paddingInvalidateRunnable, timeToUpdate - t +
1);
            } else {
                paddingViewMask |= viewMask;
            }
        }
        isInvalidating = false;
    }

    private volatile boolean idPaddingInvalidate = false;
    private volatile int paddingViewMask = -1;
    private Handler paddingInvalidateHandler = new Handler();

    // Am I need to use runOnUiThread() ?
    private Runnable paddingInvalidateRunnable = new Runnable() {
        @Override
        public void run() {
            if (idPaddingInvalidate) {
                // It is possible that t-timeToUpdate <= 0 here, don't know why
                invalidateGraphView(paddingViewMask);
            }
        }
    };
}
```

Anexo6. AVESVISTAS

```
public class Aves_vistas extends Activity {
    Button borrar1,seguir,v_lis;
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.aves_vistas);
        borrar1=(Button)findViewById(R.id.borrar) ;
        seguir = (Button) findViewById(R.id.s_buscando);
        v_lis = (Button) findViewById(R.id.lista);



        seguir.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                Intent mainIntent1 = new Intent().setClass(Aves_vistas.this,
AnalyzerActivity.class);
                startActivity(mainIntent1);
            }
        });
        v_lis.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                Intent mainIntent2 = new Intent().setClass(Aves_vistas.this,Lis_aves.class);
                startActivity(mainIntent2);
            }
        });


        borrar1.setOnClickListener(new View.OnClickListener() {
            @TargetApi(Build.VERSION_CODES.GINGERBREAD)
            @Override
            public void onClick(View v) {
               SharedPreferences avevista= getSharedPreferences("Avevista",Context.MODE_PRIVATE);
                SharedPreferences.Editor  editor= avevista.edit();
                editor.clear();
                editor.apply();
                editor.remove("avecanto");
                editor.remove("cuando");
                editor.commit();
                Intent Intent3 = new Intent().setClass(Aves_vistas.this, AnalyzerActivity.class);
                startActivity(Intent3);

            }


        });
        final LinearLayout layoutdinamico = (LinearLayout) findViewById(dinamico);
        final SharedPreferences avevista = getSharedPreferences("Avevista", Context.MODE_PRIVATE);
        String nom1 = avevista.getString("avecanto", "no ");
        String[] nomV = nom1.split(" ");
        String fec1 = avevista.getString("cuando", "no ");
        String[] fecV = fec1.split(" ");
        String imagen1 = avevista.getString("ima1", "no ");
        String[] imaV = imagen1.split(" ");


        for (int f = 1; f < nomV.length; f++) {
            String nombre = "Nombre del Ave: ";
            String fecha = "Fecha de Avistamiento:";
            TextView nomT=new TextView(this);
            TextView fecT=new TextView(this);


            nomT.setTextSize(15);
            nomT.setTextColor(Color.YELLOW);
            nomT.setText(nombre+""+ nomV[f]);
            layoutdinamico.addView(nomT);

            fecT.setTextColor(Color.YELLOW);
            fecT.setTextSize(15);
            fecT.setText( fecha+"" +fecV[f]);
            layoutdinamico.addView(fecT);


        }

    }
}
```

Anexo7. DESCRI_AVE

```
public class Descri_ave extends Activity implements View.OnClickListener {
    TextView titulo, descripcion;
    ImageView imagen;
    ImageButton correcto, compartir, incorrecto;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.descri_ave);
        titulo = (TextView) findViewById(R.id.texttitulo);
        descripcion = (TextView) findViewById(R.id.textDescri);
        imagen = (ImageView) findViewById(R.id.imageView5);
        incorrecto = (ImageButton) findViewById(R.id.incorr);
        correcto = (ImageButton) findViewById(R.id.confir);
        compartir = (ImageButton) findViewById(R.id.compa);

        incorrecto.setOnClickListener(this);
        correcto.setOnClickListener(this);
        compartir.setOnClickListener(this);


        Intent uno = getIntent();
        Bundle llego = uno.getExtras();
        if (llego != null) {


            String cadena3 = (String) llego.get("nr_ave");

            switch (cadena3) {
                case "1":
                    titulo.setText(R.string.T_A_1);
                    imagen.setImageResource(R.drawable.pajaro1);
                    descripcion.setText(R.string.D_A_1);

                    break;

                case "2":
                    titulo.setText(R.string.T_A_2);
                    imagen.setImageResource(R.drawable.pajaro2);
                    descripcion.setText(R.string.D_A_2);

                    break;
                case "3":
                    titulo.setText(R.string.T_A_3);
                    imagen.setImageResource(R.drawable.pajaro3);
                    descripcion.setText(R.string.D_A_3);


                    break;

                case "4":
                    titulo.setText(R.string.T_A_4);
                    imagen.setImageResource(R.drawable.pajaro4);
                    descripcion.setText(R.string.D_A_4);
                    break;
                case "5":
                    titulo.setText(R.string.T_A_5);
                    imagen.setImageResource(R.drawable.pajaro5);
                    descripcion.setText(R.string.D_A_5);
                    break;

                case "6":
                    titulo.setText(R.string.T_A_6);
                    imagen.setImageResource(R.drawable.pajaro6);
                    descripcion.setText(R.string.D_A_6);
                    break;
                case "7":
                    titulo.setText(R.string.T_A_7);
                    imagen.setImageResource(R.drawable.pajaro7);
                    descripcion.setText(R.string.D_A_7);
                    break;

                case "8":
                    titulo.setText(R.string.T_A_8);
                    imagen.setImageResource(R.drawable.pajaro8);
                    descripcion.setText(R.string.D_A_8);
                    break;
                case "9":
```

```
        titulo.setText(R.string.T_A_9);
        imagen.setImageResource(R.drawable.pajaro9);
        descripcion.setText(R.string.D_A_9);
        break;

    case "10":
        titulo.setText(R.string.T_A_10);
        imagen.setImageResource(R.drawable.pajaro10);
        descripcion.setText(R.string.D_A_10);
        break;
    case "11":
        titulo.setText(R.string.T_A_11);
        imagen.setImageResource(R.drawable.pajaro11);
        descripcion.setText(R.string.D_A_11);
        break;

    case "12":
        titulo.setText(R.string.T_A_12);
        imagen.setImageResource(R.drawable.pajaro12);
        descripcion.setText(R.string.D_A_12);
        break;
    case "13":
        titulo.setText(R.string.T_A_13);
        imagen.setImageResource(R.drawable.pajaro13);
        descripcion.setText(R.string.D_A_13);
        break;

    case "14":
        titulo.setText(R.string.T_A_14);
        imagen.setImageResource(R.drawable.pajaro14);
        descripcion.setText(R.string.D_A_14);
        break;
    case "15":
        titulo.setText(R.string.T_A_15);
        imagen.setImageResource(R.drawable.pajaro15);
        descripcion.setText(R.string.D_A_15);
        break;

    case "16":
        titulo.setText(R.string.T_A_16);
        imagen.setImageResource(R.drawable.pajaro16);
        descripcion.setText(R.string.D_A_16);
        break;
    case "17":
        titulo.setText(R.string.T_A_17);
        imagen.setImageResource(R.drawable.pajaro17);
        descripcion.setText(R.string.D_A_17);
        break;

    case "18":
        titulo.setText(R.string.T_A_18);
        imagen.setImageResource(R.drawable.pajaro18);
        descripcion.setText(R.string.D_A_18);
        break;
    case "19":
        titulo.setText(R.string.T_A_19);
        imagen.setImageResource(R.drawable.pajaro19);
        descripcion.setText(R.string.D_A_19);
        break;

    case "20":
        titulo.setText(R.string.T_A_20);
        imagen.setImageResource(R.drawable.pajaro20);
        descripcion.setText(R.string.D_A_20);
        break;

    case "21":
        titulo.setText(R.string.T_A_21);
        imagen.setImageResource(R.drawable.pajaro21);
        descripcion.setText(R.string.D_A_21);
        break;

    case "22":
        titulo.setText(R.string.T_A_22);
        imagen.setImageResource(R.drawable.pajaro22);
        descripcion.setText(R.string.D_A_22);
        break;

    case "23":
        titulo.setText(R.string.T_A_23);
        imagen.setImageResource(R.drawable.pajaro23);
        descripcion.setText(R.string.D_A_23);
```

```
                break;

            case "24":
                titulo.setText(R.string.T_A_24);
                imagen.setImageResource(R.drawable.pajaro24);
                descripcion.setText(R.string.D_A_24);
                break;

            case "25":
                titulo.setText(R.string.T_A_25);
                imagen.setImageResource(R.drawable.pajaro25);
                descripcion.setText(R.string.D_A_25);
                break;

            case "26":
                titulo.setText(R.string.T_A_26);
                imagen.setImageResource(R.drawable.pajaro26);
                descripcion.setText(R.string.D_A_26);
                break;

            case "27":
                titulo.setText(R.string.T_A_27);
                imagen.setImageResource(R.drawable.pajaro27);
                descripcion.setText(R.string.D_A_27);
                break;

            case "28":
                titulo.setText(R.string.T_A_28);
                imagen.setImageResource(R.drawable.pajaro28);
                descripcion.setText(R.string.D_A_28);
                break;

            case "29":
                titulo.setText(R.string.T_A_29);
                imagen.setImageResource(R.drawable.pajaro29);
                descripcion.setText(R.string.D_A_29);
                break;

            case "30":
                titulo.setText(R.string.T_A_30);
                imagen.setImageResource(R.drawable.pajaro30);
                descripcion.setText(R.string.D_A_30);
                break;
            default:
                break;

        }


    }
}

@TargetApi(Build.VERSION_CODES.GINGERBREAD)
@Override
public void onClick(View v) {
    String m_titulo = titulo.getText().toString();

    switch (v.getId()) {

        case R.id.incorr:
            Intent mainIntent = new Intent().setClass(Descri_ave.this, AnalyzerActivity.class);
            startActivity(mainIntent);
            break;
        case R.id.confir:
            Calendar fecha =Calendar.getInstance();
            int dia =fecha.get(Calendar.DAY_OF_MONTH);
            int mes =fecha.get(Calendar.MONTH);
            int ano =fecha.get(Calendar.YEAR);
            String dia1 =String.valueOf(dia);
            String mes1 =String.valueOf(mes+1);
            String ano1 =String.valueOf(ano);
            String dma = (dia1+"/"+mes1+"/"+ano1+".");
            SharedPreferences avevista =getSharedPreferences("Avevista", Context.MODE_PRIVATE);
            String avecanto= avevista.getString("avecanto","");
            String cuando =avevista.getString("cuando","");
            String ima=avevista.getString("ima1","");
            SharedPreferences.Editor  editor= avevista.edit();
            editor.clear();
            editor.apply();
            editor.putString("avecanto",avecanto+" "+m_titulo);
            editor.putString("cuando",cuando+" "+dma);
```

```
                editor.commit();
                  Toast.makeText(getBaseContext(),(m_titulo+" "+dma),Toast.LENGTH_LONG).show();
                Intent intent1 = new Intent(Descri_ave.this,Aves_vistas.class);
                startActivity(intent1);

                break;
            case R.id.compa:
                Intent intent = new Intent(Intent.ACTION_SEND);
                intent.setType("text/plain");
                intent.putExtra(Intent.EXTRA_TEXT, "ENCONTRE EL AVE:"+" "+m_titulo+ " "
+"#UNIMAGDALENA #BIRDSTEN");
                intent.setPackage("com.twitter.android");
                startActivity(intent);
                break;
            default:
                break;
        }
    }
}
```

Anexo8. LIS_AVES

```
public class Lis_aves extends Activity implements View.OnClickListener {

    ImageButton
bot1,bot2,bot3,bot4,bot5,bot6,bot7,bot8,bot9,bot10,bot11,bot12,bot13,bot14,bot15,bot16,bot17,bot18,b
ot19,bot20,bot21,bot22,bot23,bot24,bot25,bot26,bot27,bot28,bot29,bot30;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.lista_aves);

       bot1=(ImageButton) findViewById(R.id.imageView100);
        bot2=(ImageButton)findViewById(R.id.imageView);
       bot3=(ImageButton)findViewById(R.id.imageView3);
        bot4=(ImageButton) findViewById(R.id.imageView8);
        bot5=(ImageButton)findViewById(R.id.imageView5);
        bot6=(ImageButton)findViewById(R.id.imageView50);
        bot7=(ImageButton) findViewById(R.id.imageView7);
        bot8=(ImageButton)findViewById(R.id.imageView80);
        bot9=(ImageButton)findViewById(R.id.imageView9);
        bot10=(ImageButton) findViewById(R.id.imageView90);
        bot11=(ImageButton)findViewById(R.id.imageView11);
        bot12=(ImageButton)findViewById(R.id.imageView2);
        bot13=(ImageButton) findViewById(R.id.imageView33);
        bot14=(ImageButton)findViewById(R.id.imageView14);
        bot15=(ImageButton)findViewById(R.id.imageView55);
        bot16=(ImageButton) findViewById(R.id.imageView6);
        bot17=(ImageButton)findViewById(R.id.imageView77);
        bot18=(ImageButton)findViewById(R.id.imageView88);
        bot19=(ImageButton) findViewById(R.id.imageView99);
        bot20=(ImageButton)findViewById(R.id.imageView20);
        bot21=(ImageButton)findViewById(R.id.imageView222);
        bot22=(ImageButton) findViewById(R.id.imageView22);
        bot23=(ImageButton)findViewById(R.id.imageView23);
        bot24=(ImageButton)findViewById(R.id.imageView24);
        bot25=(ImageButton) findViewById(R.id.imageView25);
        bot26=(ImageButton)findViewById(R.id.imageView26);
        bot27=(ImageButton)findViewById(R.id.imageView827);
        bot28=(ImageButton) findViewById(R.id.imageView27);
        bot29=(ImageButton)findViewById(R.id.imageView29);
        bot30=(ImageButton)findViewById(R.id.imageView30);

        bot1.setOnClickListener(this);
        bot2.setOnClickListener(this);
        bot3.setOnClickListener(this);
        bot4.setOnClickListener(this);
        bot5.setOnClickListener(this);
        bot6.setOnClickListener(this);
        bot7.setOnClickListener(this);
        bot8.setOnClickListener(this);
        bot9.setOnClickListener(this);
        bot10.setOnClickListener(this);
        bot11.setOnClickListener(this);
        bot12.setOnClickListener(this);
        bot13.setOnClickListener(this);
        bot14.setOnClickListener(this);
        bot15.setOnClickListener(this);
        bot16.setOnClickListener(this);
        bot17.setOnClickListener(this);
        bot18.setOnClickListener(this);
        bot19.setOnClickListener(this);
        bot20.setOnClickListener(this);
        bot21.setOnClickListener(this);
        bot22.setOnClickListener(this);
        bot23.setOnClickListener(this);
        bot24.setOnClickListener(this);
        bot25.setOnClickListener(this);
        bot26.setOnClickListener(this);
        bot27.setOnClickListener(this);
        bot28.setOnClickListener(this);
        bot29.setOnClickListener(this);
        bot30.setOnClickListener(this);

    }
```

```
public void descri( String nrave){

    Intent mainIntent = new Intent().setClass(Lis_aves.this, Descripcion.class);

    mainIntent.putExtra("nr_ave",nrave);
    startActivity(mainIntent);

}

@Override
public void onClick(View v) {
    switch (v.getId()){

        case R.id.imageView100:
           descri("1");
            break;
        case R.id.imageView:
            descri("2");
            break;
        case R.id.imageView3:
            descri("3");
            break;
        case R.id.imageView8:
            descri("4");
            break;
        case R.id.imageView5:
            descri("5");
            break;
        case R.id.imageView50:
            descri("6");
            break;
        case R.id.imageView7:
            descri("7");
            break;
        case R.id.imageView80:
            descri("8");
            break;
        case R.id.imageView9:
            descri("9");
            break;
        case R.id.imageView90:
            descri("10");
            break;
        case R.id.imageView11:
            descri("11");
            break;
        case R.id.imageView2:
            descri("12");
            break;
        case R.id.imageView33:
            descri("13");
            break;
        case R.id.imageView14:
            descri("14");
            break;
        case R.id.imageView55:
            descri("15");
            break;
        case R.id.imageView6:
            descri("16");
            break;
        case R.id.imageView77:
            descri("17");
            break;
        case R.id.imageView88:
            descri("18");
            break;
        case R.id.imageView99:
            descri("19");
            break;
        case R.id.imageView20:
            descri("20");
            break;
        case R.id.imageView222:
            descri("21");
            break;
        case R.id.imageView22:
            descri("22");
            break;
        case R.id.imageView23:
            descri("23");
```

```
                break;
            case R.id.imageView24:
                descri("24");
                break;
            case R.id.imageView25:
                descri("25");
                break;
            case R.id.imageView26:
                descri("26");
                break;
            case R.id.imageView827:
                descri("27");
                break;
            case R.id.imageView27:
                descri("28");
                break;
            case R.id.imageView29:
                descri("29");
                break;
            case R.id.imageView30:
                descri("30");
                break;
            default:
                break;

        }
    }
}
```

Anexo9. PRINCIPAL

```
public class Principal extends Activity {
    private Button inicio;
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.principal);

        inicio=(Button)findViewById(R.id.Boton_ini) ;

        inicio.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {



                Intent mainIntent = new Intent().setClass(Principal.this,
AnalyzerActivity.class);
                startActivity(mainIntent);


            }


        });

    }
}
```

Anexo10. AVESVISTAS

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
    <LinearLayout
        android:id="@+id/dinamico"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">



    </LinearLayout>
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_marginTop="10dp">

    <Button
        android:id="@+id/s_buscando"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/boton_Regr"
        android:layout_gravity="center"
        />
    <Button
        android:id="@+id/lista"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/boton_lista"
        android:layout_gravity="center"
        />


</LinearLayout>
        <Button
            android:id="@+id/borrar"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/boton_Borrar_H"
            android:layout_gravity="center"
            />
    </LinearLayout>

</ScrollView>
```

Anexo11. DESCRI_AVE

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background ="#99CC33"
    android:weightSum="1">

    <TextView
        android:id="@+id/texttitulo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text=""
        android:textColor="#660099"
        android:textSize="30sp" />

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:src="@drawable/pajaro1"
        android:layout_marginTop="15dp"
        android:id="@+id/imageView5"
        android:contentDescription="@string/title_activity_Descripcion"/>

    <ScrollView
        android:id="@+id/scroll1"
        android:layout_width="match_parent"
        android:layout_height="100sp"
        android:layout_marginTop="15dp">

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

            <TextView
                android:id="@+id/textDescri"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:gravity="center"
                android:text="@string/D_A_10"
                android:textColor="#003300"
                android:textSize="20sp" />


        </LinearLayout>
    </ScrollView>

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_gravity="center">

        <ImageButton
            android:id="@+id/confir"
            android:layout_width="57dp"
            android:layout_height="59dp"
            android:text="@string/corr"
            android:background="@drawable/chulo22"
            android:contentDescription="@string/corr"/>
        <TextView

            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:text="@string/corr"
            android:textColor="#003300"
            android:textSize="20sp" android:visibility="invisible" />
        <ImageButton
            android:id="@+id/incorr"
            android:layout_width="59dp"
            android:layout_height="match_parent"
            android:text="@string/incorr"
            android:background="@drawable/equis22"
            android:contentDescription="@string/incorr"/>

    </LinearLayout>
```

```
    <ImageButton
        android:id="@+id/compa"
        android:layout_width="61dp"
        android:layout_height="wrap_content"
        android:text="@string/comp"
        android:layout_marginTop="10dp"

        android:layout_gravity="center"
        android:background="@drawable/btwitter22"
        android:contentDescription="@string/comp"
        android:layout_weight="0.24" />


</LinearLayout>
```

Anexo12. DESCRIPCION

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background ="#EEFF41">

    <TextView
        android:text=""
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="24dp"
        android:id="@+id/texttitulo"
        android:textSize="30sp"
        android:textColor="#660099"/>

    <ScrollView
        android:id="@+id/scroll1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/imageView5"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical" >
            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text=""
                android:textColor="#003300"
                android:textSize="20sp"
                android:gravity="center"
                android:id="@+id/textDescri"
                />



        </LinearLayout>
    </ScrollView>

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:src="@drawable/blanco"
        android:layout_marginTop="15dp"
        android:id="@+id/imageView5"
        android:layout_below="@+id/texttitulo"
        android:layout_centerHorizontal="true" />


</RelativeLayout>
```

Anexo13.LISTA_AVES

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:custom="http://schemas.android.com/apk/res-auto"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:background="@drawable/imagen_fondo">

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="50sp"
        android:gravity="center"
        android:orientation="horizontal"
        android:visibility="invisible">

        <github.bewantbe.audio_analyzer_for_android.SelectorText
            android:id="@+id/button_recording"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_margin="4dp"
            android:tag="select"
            android:text="Mon"
            android:textSize="20sp"
            custom:items="Mon Rec"
            android:visibility="invisible"/>

        <Button
            android:id="@+id/button_sample_rate"
            android:layout_width="wrap_content"
            android:layout_height="@dimen/button_button_height"
            android:layout_gravity="center"
            android:onClick="showPopupMenu"
            android:text="@string/sample_s"
            android:textSize="@dimen/button_text_fontsize"
            android:visibility="invisible"/>

        <Button
            android:id="@+id/button_fftlen"
            android:layout_width="wrap_content"
            android:layout_height="@dimen/button_button_height"
            android:layout_gravity="center"
            android:onClick="showPopupMenu"
            android:text="@string/fftlen"
            android:textSize="@dimen/button_text_fontsize"
            android:visibility="invisible"/>

        <Button
            android:id="@+id/button_average"
            android:layout_width="wrap_content"
            android:layout_height="@dimen/button_button_height"
            android:layout_gravity="center"
            android:onClick="showPopupMenu"
            android:text="@string/ave"
            android:textSize="@dimen/button_text_fontsize"
            android:visibility="invisible"/>


    </LinearLayout>

    <TextView
        android:text="@string/mnsaje"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/mensaje_pantalla"
        android:textColor="@android:color/white"
        android:textAlignment="inherit" />

    <TextView
        android:id="@+id/textview_rec"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/textview_rec_text"
        android:typeface="monospace"
        android:visibility="invisible"/>

    <view
        android:id="@+id/plot"
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="0.26"
```

```
    class="github.bewantbe.audio_analyzer_for_android.AnalyzerGraphic"
    custom:cutoffDb="-25"
    custom:sampleRate="16000" />

<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >

    <TextView
        android:id="@+id/textview_RMS"
        android:layout_width="@dimen/textview_RMS_layout_width"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:lines="2"
        android:text="@string/textview_RMS_text"
        android:typeface="monospace"
        android:visibility="invisible"/>

    <TextView
        android:id="@+id/textview_cur"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_toRightOf="@id/textview_RMS"
        android:layout_toEndOf="@id/textview_RMS"
        android:text="@string/textview_cur_text"
        android:typeface="monospace"
        android:visibility="invisible"/>

    <TextView
        android:id="@+id/textview_peak"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/textview_cur"
        android:layout_toRightOf="@id/textview_RMS"
        android:layout_toEndOf="@id/textview_RMS"
        android:text="@string/textview_peak_text"
        android:typeface="monospace"
        android:textColor="@android:color/white"/>
</RelativeLayout>

<LinearLayout
    android:orientation="horizontal"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:id="@+id/botones">


    <Button
        android:id="@+id/base"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/B_datos" />

    <Button
        android:id="@+id/detener"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/deten" />

</LinearLayout>

<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:gravity="center"
   >

    <github.bewantbe.audio_analyzer_for_android.SelectorText
        android:id="@+id/spectrum_spectrogram_mode"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="4dp"
        android:tag="select"
        android:text="spum"
        android:textSize="20sp"
        custom:items="spum spam"
        android:visibility="invisible"/>

    <github.bewantbe.audio_analyzer_for_android.SelectorText
```

```
            android:id="@+id/dbA"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_margin="4dp"
            android:tag="select"
            android:text="dB"
            android:textSize="20sp"
            custom:items="dB dBA"
            android:visibility="invisible"/>

        <!--<github.bewantbe.audio_analyzer_for_android.SelectorText-->
        <!--android:id="@+id/graph_view_mode"-->
        <!--android:layout_width="wrap_content"-->
        <!--android:layout_height="wrap_content"-->
        <!--android:layout_margin="4dp"-->
        <!--android:paddingLeft="15dp"-->
        <!--android:tag="select"-->
        <!--android:text="scale"-->
        <!--android:textSize="20sp"-->
        <!--custom:items="cursor scale" />-->



        <github.bewantbe.audio_analyzer_for_android.SelectorText
            android:id="@+id/freq_scaling_mode"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_margin="4dp"
            android:tag="select"
            android:text="linear"
            android:textSize="20sp"
            custom:items="linear log"
            android:visibility="invisible"/>

        <github.bewantbe.audio_analyzer_for_android.SelectorText
            android:id="@+id/run"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_margin="4dp"
            android:tag="select"
            android:text="run"
            android:textSize="20sp"
            custom:items="run stop"
            android:visibility="invisible"/>
    </LinearLayout>

</LinearLayout>
```

Anexo14. PINCIPAL

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/bienvenidos2">

    <Button
        android:text="@string/boton_ini_A"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="24dp"

        android:id="@+id/Boton_ini" />
</RelativeLayout>
```