UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

TÍTULO: Samplers: Framework para construir aplicaciones Android para recolectar muestras en proyectos de Ciencia Ciudadana

AUTORES: Laura Lus y Javier Ramírez

DIRECTOR: Diego Torres

CODIRECTOR: Alejandro Fernandez

ASESOR PROFESIONAL:

CARRERA: Licenciatura en Informática

Resumen

La Ciencia Ciudadana involucra al público en proyectos de investigación científica. Las tareas de los participantes, denominados científicos ciudadanos, pueden ser simples y no necesitar ningún conocimiento especial, como por ejemplo contar elementos que aparecen en una fotografía o bien responder una serie de preguntas sobre un ambiente que se está observando. El presente trabajo presenta un framework Android para construir aplicaciones móviles que permitan la recolección de muestras utilizando las herramientas que brindan los dispositivos móviles, como puede ser geolocalización o toma de fotografías. Está orientado a los proyectos de recolección, que son los que requieren recolectar muestras del medio físico, y más específicamente a los que requieren recolectar muestras haciendo uso de dispositivos móviles.

Palabras Clave

Ciencia Ciudadana, Android, Framework, Método científico, Protocolo de recolección de muestras.

Conclusiones

Se propuso desarrollar un framework para instanciar aplicaciones móviles Android de ciencia ciudadana, que reciba un archivo con la configuración requerida en formato JSON y genere una aplicación para ejecutarse en un dispositivo Android.

Esto se vio alcanzado con el desarrollo del framework Samplers el cual recibe un archivo de configuración dónde se especifican los pasos necesarios para recolectar una muestra y con ello produce el código de una aplicación móvil para Android. El código queda listo para compilar y ejecutar en un dispositivo móvil con Android o en un emulador virtual de los que provee Android Studio.

Trabajos Realizados

Se desarrolló un framework para instanciar aplicaciones móviles Android de ciencia ciudadana. El framework recibe un archivo con la configuración requerida en formato JSON y genera una aplicación para ejecutarse en un dispositivo Android.

Se instanció una aplicación tomando como ejemplo una app de ciencia ciudadana que ya se encuentra en funcionamiento, que es AppEAR, y se generó una versión usando Samplers, haciendo una breve comparación entre ambas.

Trabajos Futuros

Poder compilar para iOS ampliaría la base de usuarios.

Mantener el código fuente actualizado para ajustarse a los cambios del sistema operativo Android que se actualiza regularmente e introduce cambios, mejoras y nuevos requerimientos para las apps.

Una mejora ya realizada por los participantes del trabajo de tesina Samplers2 es Muestre.AR, una interfaz web que permite a los investigadores definir el protocolo de recolección de una muestra utilizando un sitio web y descargar la aplicación Android resultante.

Fecha de la presentación: JULIO/2022

Samplers: Framework para construir aplicaciones Android para recolectar muestras en proyectos de Ciencia Ciudadana

Laura Lus y Javier Ramírez

6 de julio de 2022

Resumen

La Ciencia Ciudadana involucra al público en proyectos de investigación científica. Las tareas de un voluntario o ciudadano científico pueden ser simples y no necesitar ningún conocimiento especial, o pueden ser más complejas y requerir capacitación previa. Ejemplos de tareas en proyectos de ciencia ciudadana pueden ser contar elementos que aparecen en una fotografía (si aparece un determinado animal o si puede reconocer una galaxia) o bien responder una serie de preguntas para recolectar información sobre un ambiente que el voluntario está observando (podría ser el ecosistema que rodea una laguna o un estuario).

Hacer partícipes a los ciudadanos de proyectos de investigación científica persigue varios fines, entre de ellos poder realizar investigaciones a gran escala temporal y espacial, brindar la oportunidad de participar en proyectos reales e interactuar con científicos, o bien perseguir fines educativos. Los proyectos de investigación que incluyen Ciencia Ciudadana pueden clasificarse en acción, conservación, recolección, virtual y educativos.

El presente trabajo presenta un framework Android para construir aplicaciones que permitan la recolección de muestras utilizando las herramientas que brindan los dispositivos móviles, como puede ser geolocalización o toma de fotografías. Está orientado a los proyectos de recolección, que son los que requieren recolectar muestras del medio físico, y más específicamente a los que requieren recolectar muestras haciendo uso de dispositivos móviles.

Agradecimientos

Quiero darle las gracias a mi madre y a mi padre por su gigantesca y necesaria presencia.

A mi hijo Camilo, a mis hermanes y amigues, no puedo pedir una hinchada mejor que ustedes.

A Damián, que el camino nos encuentre siempre compartiendo mates y estudio.

Laura

Quiero agradecer a mis viejos, por los valores inculcados, por haberme dado siempre lo mejor de ellos y por haber hecho el gran esfuerzo para que pudiera estudiar en La Plata.

A mis hermanos y cuñadas, por el apoyo y acompañamiento cuando más lo necesitaba.

A mi novia, por apoyarme, acompañarme y estimularme para que terminara esta tesina.

A Sura, donde quiera que estés, por haberme hecho compañía todo el tiempo mientras desarrollaba este trabajo.

Javier

Queremos agradecer a nuestros directores, Diego y Alejandro, por habernos apoyado para terminar esta tesina.

También queremos agradecer a la educación pública porque si no fuera gratuita esto no habría sido posible, y en especial a la Facultad de Informática de la UNLP por la calidad académica que brinda a sus alumnos.

Laura y Javier

Índice general

1. Introducción **6**
1.1. Motivación 6
1.2. Objetivos 7
1.3. Estructura de la Tesina 8

2. Marco Teórico **10**
2.1. Introducción a la Ciencia Ciudadana 10
2.2. Clasificación de los Proyectos de Ciencia Ciudadana 12
2.3. Proyectos de Recolección 15
2.4. El método científico 16
2.5. Protocolo de recolección de muestras en proyectos de ciencia ciudadana 17
2.6. Ciencia ciudadana y dispositivos móviles 18

3. Frameworks para proyectos de ciencia ciudadana **20**
3.1. Frameworks 20
3.1.1. Tipos de Frameworks 22
3.1.1.1. Frameworks de caja blanca 22
3.1.1.2. Frameworks de caja negra 23
3.2. Estado del arte 23
3.2.1. Sensr 24
3.2.2. Project Noah 24
3.2.3. EpiCollect 25
3.2.4. CitSci 25
3.2.5. Spotteron 27
3.2.6. Zooniverse 27
3.2.7. Conclusión 29

4. El entorno de Android **31**
4.1. Activities . . . 33
4.2. Fragments . . . 36
4.3. Services . . . 38
4.4. BroadcastReceivers . . . 39

5. Samplers: Framework Android **41**
5.1. Propuesta general . . . 41
5.1.1. Descripción del problema . . . 41
5.1.2. Alcance de la solución propuesta . . . 43
5.1.3. Descripción de la solución propuesta: Samplers . . . 44
5.2. Estructura e implementación del framework . . . 49
5.2.1. Workflow, Step, StepFragment, StepResult, Sample . . . 49
5.2.1.1. Workflow: el protocolo de recolección de las muestras . . . 51
5.2.1.2. Step: el paso . . . 52
5.2.1.3. StepFragment: la vista y controlador del Step 53
5.2.1.4. StepResult: el resultado de la ejecución de un Step . . . 53
5.2.1.5. Sample: la Muestra . . . 54
5.2.2. TakeSampleActivity . . . 54
5.2.3. Persistencia local . . . 54
5.2.4. Envío de muestras a servidor web . . . 55
5.2.5. Identificación . . . 56
5.2.6. Otras Activities de la app generada por Samplers . . . 56
5.2.6.1. SamplersMainActivity . . . 56
5.2.6.2. SamplesListActivity . . . 57
5.2.6.3. HelpActivity . . . 58
5.3. Configuración del framework . . . 59
5.3.1. Archivo de configuración . . . 59
5.3.2. Usando las clases . . . 62

6. Instanciación y uso del framework **64**
6.1. Instanciación manual . . . 64
6.2. Instanciación usando el generador de clases de Gradle . . . 66
6.3. Secciones del archivo . . . 67
6.3.1. El objeto project . . . 67
6.3.2. El objeto application . . . 68
6.3.3. El objeto workflow . . . 69
6.4. Mostrar Ayuda . . . 71

6.4.1. Ayuda general 72
6.4.2. Ayuda puntual para cada Step 73
6.5. Usando identificación 73
6.5.1. Configurar identificación con el generador de clases de Gradle 74
6.5.2. Configurar identificación manualmente 75
6.5.3. Usando un método de identificación propio 76
6.5.3.1. Definiendo un Login Fragment propio 76
6.5.3.2. Definiendo una clase User propia 77
6.6. Los diferentes Steps y sus resultados (StepResult) 78
6.6.1. PhotoStep: Tomar una foto 78
6.6.1.1. PhotoStepResult: El resultado de Tomar una foto 79
6.6.2. SoundRecordStep: Grabar sonido 79
6.6.2.1. SoundRecordStepResult: El resultado de Grabar sonido 81
6.6.3. InformationStep: Mostrar información 81
6.6.3.1. InformationStepResult: El resultado de Mostrar información 82
6.6.4. SelectOneStep: Seleccionar una opción de un grupo de opciones 82
6.6.4.1. SelectOneStepResult: El resultado de Seleccionar una opción de un grupo de opciones . 84
6.6.5. MultipleSelectStep: Seleccionar varias opciones de un grupo de opciones 84
6.6.5.1. MultipleSelectStepResult: El resultado de Seleccionar varias opciones de un grupo de opciones 86
6.6.6. LocationStep: Posicionar la muestra en el mapa con el GPS 86
6.6.6.1. LocationStepResult: El resultado de Posicionar la muestra en el mapa con el GPS 88
6.6.7. RouteStep: Grabar un recorrido en el mapa usando el GPS 88
6.6.7.1. RouteStepResult: El resultado de Grabar un recorrido en el mapa usando el GPS 89
6.6.8. InsertTextStep: Ingresar texto 89
6.6.8.1. InsertTextStepResult: El resultado de Ingresar texto 91
6.6.9. InsertDateStep e InsertTimeStep: Ingresar fecha y hora 91

6.6.9.1. InsertDateStepResult e InsertTimeStep: El resultado de Ingresar fecha y hora 93
6.6.10. Definir un nuevo Step, StepFragment y StepResult . . 93
6.6.10.1. Definir un nuevo Step 94
6.6.10.2. Definir un nuevo StepFragment 94
6.6.10.3. Definir un nuevo StepResult 95

7. Caso de uso **96**
7.1. AppEAR 96
7.2. AppEar usando Samplers 100
7.3. Comparación y conclusión 107

8. Conclusiones y Trabajo Futuro **112**
8.1. Conclusiones 112
8.2. Trabajo Futuro 114

Anexos **116**

A. Instalación del framework **117**
A.1. Requerimientos mínimos: 117
A.2. Pasos para la Instalación: 117

Capítulo 1

Introducción

1.1. Motivación

Los proyectos de investigación científica a menudo requieren la realización de gran número de actividades que son difíciles de automatizar como puede ser la clasificación de fotos, anotaciones, observaciones y todo tipo de actividades que en esencia son simples, pero consumen mucho tiempo. Muchas veces estas actividades son sencillas y no se necesita de ninguna preparación académica o escolarizada previa para realizarlas, por ejemplo indicar si en una foto se observa o no un animal. La ciencia ciudadana es una forma de investigación en colaboración que involucra a los ciudadanos resolviendo este tipo de tareas simples en proyectos de investigación científica que buscan resolver problemas del mundo real [1].

Un científico ciudadano es un voluntario que recoge y/o procesa información como parte de una investigación científica [2]. Para que los voluntarios puedan participar en estos proyectos es necesario brindarles herramientas que los ayuden a contribuir. Nuestro interés está enfocado en los proyectos de recolección. Estos proyectos de investigación científica requieren la recopilación de datos del medio físico. Una forma de asistir a estos proyectos es por medio de sistemas informáticos que posibiliten la recolección de datos usando móviles. Un ejemplo de este tipo de proyectos es AppEAR[3] un sistema de ciencia ciudadana para cuidar y aprender de los ambientes acuáticos en Argentina, realizado por Joaquín Cochero, investigador del CONICET en el Instituto Platense de Limnología. El objetivo final de AppEAR es tener un relevamiento completo y detallado de aguas continentales de todo el territorio nacional para conocer los lugares en riesgo en los que urge trabajar. Los voluntarios de este proyecto descargan una aplicación para su dispositivo

móvil y toman muestras para el proyecto. La aplicación guía a los usuarios a través de los pasos necesarios para tomar una muestra.

La mayoría de los proyectos de ciencia ciudadana de recolección cuentan con aplicaciones desarrolladas específicamente para cada proyecto, en donde el principal problema a resolver es la secuencia de pasos que conforman el protocolo para la toma de la muestra y la combinación de este protocolo y de las herramientas del dispositivo móvil que se desean utilizar cómo puede ser la cámara, el GPS, el micrófono para grabar un audio. Consideramos que proveer un framework que resuelva esta problemática, la de la aplicación específica de cada proyecto, sería útil para la creciente comunidad de científicos que quieren incluir ciencia ciudadana en sus proyectos.

Este proyecto se enmarca dentro de Cientópolis[4], una plataforma para la promoción y el estudio de la Ciencia Ciudadana. Cientópolis se nuclea como un proyecto de investigación desde la Facultad de Informática de la UNLP pero articula su funcionamiento con investigadores de las facultades de Ciencias Astronómicas y Geofísicas, Humanidades y Ciencias de la Educación, Bellas Artes y Ciencias Naturales y Museo.

1.2. Objetivos

Se propone desarrollar un framework para instanciar aplicaciones móviles Android de ciencia ciudadana. El framework recibirá un archivo con la configuración requerida en formato JSON y generará una aplicación para ejecutarse en un dispositivo Android. En este archivo estará el conjunto de pasos que especifican el protocolo de recolección de muestras. Estos pasos pueden ser:

- captura de una foto, un video, un audio, una ubicación o un recorrido hecho con el dispositivo móvil.
- contestar una pregunta con respecto a la muestra. Esta pregunta puede tener una o múltiples respuestas posibles.
- introducir anotaciones de texto.
- indicar una fecha y hora.
- mostrar información de orientación y ayuda para la toma de la muestra.

La aplicación generada servirá para tomar muestras siguiendo el protocolo de recolección especificado y las almacenará y empaquetará en el dispositivo móvil hasta que pueda ser enviada a un servidor web.

Definir el formato del archivo de configuración de la aplicación y la información adicional necesaria, como pueden ser credenciales para acceder a los servicios de Google Services o el posicionamiento por GPS.

Instanciar una aplicación básica de ejemplo con el framework en base a un archivo de configuración, que permita tomar algunas muestras y enviarlas a un servidor web que estará configurado para dicho propósito.

1.3. Estructura de la Tesina

Este trabajo de tesina se organiza de la siguiente manera:

- Capítulo 1 En este capítulo se plantea la motivación de este trabajo y se definen los objetivos a alcanzar. Se detalla la estructura de esta tesina.

- Capítulo 2

  Este capítulo comienza explicando qué es la ciencia ciudadana y cómo es que su popularidad está en aumento. Detalla una clasificación de proyectos de ciencia ciudadana y luego ahonda en uno de los tipos de la clasificación, que son aquellos proyectos donde la inclusión de los científicos ciudadanos se realiza en la recolección de información o muestras. Se describe el método científico ya que los proyectos de investigación que utilizan ciencia ciudadana o cuyo diseño gira en torno a incluir a científicos ciudadanos son proyectos que utilizan las bases de la investigación científica, es decir, son proyectos que utilizan el método científico. A diferencia de los proyectos de investigación donde todos sus participantes son científicos o personas con conocimiento o experticia en el área de estudio, los proyectos de ciencia ciudadana deben tener especial cuidado definiendo los protocolos de recolección de la información para que sus participantes puedan seguirlos. Por último, teniendo en cuenta que los dispositivos móviles están al alcance de muchas personas, presentamos datos de uso de dispositivos móviles y sistemas operativos en el país.

- Capítulo 3

  Se introduce el concepto de framework y se describe una clasificación en base a su diseño y tipo de especialización. Luego se describe el

estado de seis herramientas, plataformas y frameworks, que asisten a los investigadores en la creación y la administración de proyectos de recolección que utilizan ciencia ciudadana. Se analizan las ventajas y las desventajas de las herramientas descritas en la sección.

- Capítulo 4

  Se describe el sistema operativo para dispositivos móviles Android y se detallan sus principales componentes de aplicación. La Activity, el componente principal de las aplicaciones en Android. Características, ciclo de vida y posibles estados. Relación entre Fragment y Activity. Ciclo de vida y estados del Fragment. Propósito de los Services y tipos soportados. Método de suscripción de eventos del sistema y de otras aplicaciones, BroadcastReceiver.

- Capítulo 5

  Se presenta el framework propuesto, Samplers, para construir aplicaciones Android para proyectos de recolección que utilizan ciencia ciudadana. Se describe el alcance de la solución propuesta y los detalles del framework.

- Capítulo 6

  Se presenta un caso de uso del framework Samplers, instanciando una aplicación que imita a AppEar. Se comparan las aplicaciones en cuanto a apariencia y funcionalidad, y se analizan los resultados obtenidos.

- Capítulo 7

  En este capítulo se sumariza el desarrollo de la tesina y se evalúan nuevamente los objetivos con los resultados obtenidos y el grado de cumplimiento. También se detallan posibles trabajos a futuro que permitirían que el proyecto crezca y se mantenga en el tiempo.

Capítulo 2

Marco Teórico

2.1. Introducción a la Ciencia Ciudadana

El término Ciencia Ciudadana se puede definir como la participación del público en esfuerzos organizados de investigación científica. Alrededor del mundo, cientos de miles de personas son "científicos ciudadanos", personas que eligen involucrarse en investigaciones científicas en su tiempo libre. [5] Los proyectos de Ciencia Ciudadana son proyectos de investigación diseñados o adaptados para que los voluntarios cumplan un rol en él; para que lleven adelante una parte. Esta parte puede representar un beneficio para el voluntario o científico ciudadano desde el punto de vista educativo, beneficiar al proyecto de investigación o bien beneficiar a ambos.[2] Dependiendo del tipo de proyecto de ciencia ciudadana, explicado en la sección 2.2, la parte desarrollada por los científicos ciudadanos puede variar e incluir actividades como clasificación de información, recolección de muestras, enviar información desde aplicaciones instaladas en sus dispositivos móviles acerca de especies invasivas, informar sobre el comportamiento de las aves o la cantidad de mariposas presentes al comienzo de la primavera o cualquier otro tema generalmente relacionado a la ecología y la conservación; como así también participar en debates locales de políticas que afectan directamente el ecosistema de una determinada ciudad o zona, como puede ser el fracking, la minería y los pesticidas en la agricultura y de esta manera influir en las políticas locales respecto de cómo regular esas actividades.[6]

Algunos de estos proyectos cuentan con una larga historia de investigación como el CBC (Christmas Bird Count)[7], el censo de aves realizado por voluntarios vigente desde el año 1900, cuya ultima edición, la 122 concluyó el 5 de enero de 2022. Otros con gran cantidad de participantes, quienes

interactúan a través de foros y aplicaciones web desde la comodidad de sus hogares. Proyectos como Zooniverse[8] o el local Galaxy Conqueror[9] donde los participantes señalan en fotografías que ven en una aplicación web, la ocurrencia o no de determinado animal o si pueden identificar una galaxia en la imagen. Otros abarcan grandes extensiones geográficas, como puede ser en The Big Butterfly Count[10], donde los participantes cuentan la cantidad de mariposas que ven en su propio patio o en un parque y participan cientificos ciudadanos de todo Reino Unido.

Aunque la ciencia ciudadana no es un concepto nuevo, su notoriedad es relativamente reciente. Los científicos ciudadanos o voluntarios ahora participan de proyectos relacionados con el cambio climático, las especies invasivas, la conservación de ecosistemas biológicos, el monitoreo de la calidad del agua y varios tópicos más. Esta popularidad se vio impulsada principalmente por tres factores:

- Disponibilidad Tecnológica

  La disponibilidad de herramientas tecnológicas que permiten distribuir información acerca de los proyectos y también recolectar la información generada por los voluntarios. De estas herramientas, internet es la más representativa, pero la tecnología móvil está jugando un rol fundamental con la popularización de smartphones. [2]

- Reconocimiento del Aporte de los Voluntarios por parte de los Profesionales

  La participación de voluntarios en un proyecto de investigación aporta diferentes atributos que pueden ser trabajo, poder de cómputo o habilidad. [11]

- Inclusión del Público General en Proyectos Científicos

  La mejor manera para que el común de la gente entienda y se involucre en proyectos científicos es siendo parte de ellos. [2]

En ciencias como pueden ser la arqueología, la astronomía o las ciencias naturales, la capacidad de observación es a veces más importante que el equipamiento costoso. Los voluntarios o científicos ciudadanos realizan actividades como parte de un proyecto científico que está especialmente diseñado o bien fue adaptado para que cumplan un rol, ya sea para fines educativos de los mismos voluntarios o para beneficio del proyecto. En los mejores ejemplos de proyectos de ciencia ciudadana, se benefician ambos: los voluntarios y el proyecto.[2]

2.2. Clasificación de los Proyectos de Ciencia Ciudadana

Tomando como referencia el artículo "From Conservation to Crowdsourcing: A Typology of Citizen Science"los proyectos de Ciencia Ciudadana pueden clasificarse en cinco tipos diferentes según sus características particulares:

- Acción

  Los proyectos clasificados en esta categoría no son planificados o iniciados por científicos, sino más bien por los ciudadanos, y generalmente requieren un compromiso a lo largo del tiempo en los problemas ambientales locales por lo cual las actividades científicas están orientadas al ambiente físico.

  Estos proyectos solicitan la colaboración de científicos como asesores o consultores, y no como organizadores. Los datos o resultados obtenidos no persiguen un fin académico, sino más bien buscan fundamentar con evidencia la toma de acciones sobre alguna situación.

- Conservación

  Al igual que los proyectos de Acción, los proyectos de Conservación son fuertemente regionales, y las actividades de los voluntarios están enfocadas mayormente en la recolección de información. La mayoría de los proyectos de investigación tienen contenido o fines educativos. También tienden a ser de alcance regional, como lo reflejan sus desafíos y metas.

  Estos proyectos buscan generar información principalmente como fuente para la toma de decisiones relacionadas al manejo de recursos, y la promoción de la administración y reconocimiento del voluntariado. También prestan especial atención a la generación de información científicamente válida. Son esfuerzos de monitoreo a largo plazo y generalmente no tienen problemas de sustentabilidad, ya que son subvencionadas por fondos públicos o reciben ingresos de agencias que son las que nuclean los proyectos o son las interesadas en sus resultados.

- Investigación o Recolección

  Los proyectos de investigación concentran su atención en investigaciones científicas cuyos objetivos requieren la recolección de información del medio físico. Este tipo de proyectos es el que mejor encaja en la

definición de Ciencia Ciudadana. Y aunque los objetivos de educación no son los principales de estos proyectos, forman parte de ellos como material de capacitación o incluyendo protocolos de recolección que alientan el aprendizaje al aire libre. El alcance varía de regional a internacional, y puede lograr participación masiva de hasta decenas de miles de voluntarios y obtener de ellos millones de observaciones anuales. La mayoría de estos proyectos están enfocados (pero no limitados a) la investigación biológica, medioambiental o meteorológica, por dar algunos ejemplos.

Una de las principales preocupaciones de este tipo de proyectos es generar resultados científicamente válidos, ya que son concebidos para generar conocimiento formal y son mayormente organizados por científicos. El cuidadoso diseño del proyecto y de las tareas son los que permiten lograr resultados válidos. También utilizan toda una serie de metodologías que permiten la validación de la información generada. Los voluntarios pueden estar dispersos geográficamente y esto es un recurso valioso ya que este tipo de proyectos intenta muchas veces registrar la distribución geográfica de determinadas especies o la ocurrencia de fenómenos naturales.

- Virtual

  En los proyectos de ciencia ciudadana denominados Virtuales todas las actividades son mediante tecnologías de la información y la comunicación, sin la intervención de elementos del entorno físico.

  Los proyectos que cumplen las condiciones para ser clasificados como virtuales provienen de la astronomía, la paleontología y la proteómica, que es una rama de la microbiología que estudia la estructura de las proteínas. Algunos proyectos de psicología podrían clasificar, pero no lo hacen porque los voluntarios colaboran como sujetos de pruebas, y esto no es formalmente considerado como colaboración en la investigación.

  Galaxy Zoo es un ejemplo de proyecto virtual de ciencia ciudadana. Desde hace más de diez años los voluntarios que colaboran con el proyecto clasifican galaxias en fotografías. Responden una serie de preguntas respecto de la foto que están observando, y de esta manera los científicos encargados del proyecto obtienen una primera clasificación, que se construye en base a las observaciones de varios voluntarios de manera independiente. [12]

  Al igual que en los proyectos de recolección anteriormente mencionados, los proyectos virtuales encuentran dificultades a la hora de con-

seguir resultados válidos en términos científicos. Estos resultados se obtienen mediante el desarrollo cuidadoso de las actividades. Como la participación es mayormente virtual, poder mantener a los voluntarios comprometidos con el proyecto es un desafío. Por eso muchas veces estos proyectos incluyen técnicas de gamificación, de competencia amigable entre participantes o de valoración de la contribución hecha por el voluntario (feedback).

- Educación

  Los proyectos de ciencia ciudadana pertenecientes a esta categoría son aquellos cuyo principal objetivo es educar a los participantes. Los participantes de estos proyectos aportan recursos educativos informales; mientras los proyectos ofrecen material educativo formal. También, las actividades están pensadas para que el participante vaya acumulando conocimientos.

  Un ejemplo de este tipo de proyectos es Fossil Finders, que centra la investigación en el análisis de fósiles del Devónico (período de la era Paleozóica) proveyendo de materiales de estudio a estudiantes y profesores de escuelas secundarias. Los estudiantes van a identificar y medir fósiles en muestras de rocas enviadas a sus aulas. Luego ingresarán los datos obtenidos en una base de datos online, y podrán comparar sus datos con los de otras escuelas participantes. Con ello van a tener la oportunidad de involucrarse con métodos de investigación reales y de asistir a los investigadores del Instituto de Investigación Paleontológica a reconstruir el pasado geológico de Nueva York. [13]

  La mayoría de estos proyectos persiguen un fin educativo, el aprendizaje y el desarrollo de habilidades científicas. Por ello incluyen actividades de análisis de datos o muestras, brindando la posibilidad de desarrollar pensamiento crítico.

Los proyectos de Ciencia Ciudadana buscan resultados que generalmente caen en tres grandes categorías: resultados que sirven a la investigación; resultados que le sirven a los participantes como pueden ser adquirir nuevas habilidades o conocimientos y/o resultados que tienen que ver con sistemas socio-ecológicos, como es influenciar políticas, construir bases para la toma de decisiones en una comunidad o participar de acciones para la conservación del medio ambiente. [14]

2.3. Proyectos de Recolección

Los proyectos de Ciencia Ciudadana englobados en esta sección de la clasificación propuesta en la seccion 2.2 se caracterizan por necesitar la recolección de muestras del medio físico.

Hay tres puntos en los que se debe enfatizar para que los voluntarios pueden recolectar y enviar información confiable: proveer información clara acerca de los protocolos de recolección, proveer formularios para el ingreso de datos que sean lo más lógicos y simples posibles y por último brindar soporte para que los participantes entiendan cómo seguir los protocolos y cómo enviar la información.[15]

- Protocolos

  Los datos que se obtienen en proyectos de Ciencia Ciudadana son recolectados mediante protocolos que especifican dónde, cuándo y cómo esos datos deben ser recolectados. Los protocolos deben definir un diseño formal o un plan de acción que permita combinar las muestras que fueron tomadas por múltiples participantes en diferentes ubicaciones para su posterior análisis. Los protocolos utilizados en proyectos de ciencia ciudadana deben ser fáciles de ejecutar, deben poder ser explicados de manera simple y directa, y deben ser desafiantes para los voluntarios.

- Formularios de ingreso de Datos

  En conjunto con un protocolo de recolección bien diseñado están los formularios de ingreso de datos. Los formularios en los que los usuarios ingresan sus observaciones deben ser fáciles de entender y completar. Es recomendable definir límites en el rango de datos que los formularios recogen para simplificar su posterior análisis. Por ejemplo, si la respuesta esperada a la pregunta 'Cuántos árboles cuenta en una cuadra' debería ser un número entre cero y 15 una respuesta como 150 podría indicar un error de tipeo. Entonces, es aconsejable pedirle al usuario que reporta que chequee si la información ingresada es correcta. De esta manera, los usuarios pueden revisar sus respuestas cvuando están fuera de los rangos establecidos. Aun así, suponiendo que el usuario indique un respuesta de esas características, es decir, una respuesta que se salga de los límites esperados; ese formulario debería guardarse con alguna marca que permita identificarlo para que los responsables del proyecto puedan analizarlo con más detalle y ver si se debe a un

cambio en el entorno que está siendo observado o si es realmente un error de ingreso de datos del usuario.

- Material Educativo

  Los participantes deben ser provistos de material educativo para entender y seguir de manera satisfactoria los protocolos del proyecto. El material educativo puede incluir guías de identificación, posters, manuales, videos, podcasts, listas de correo y FAQ para que los participantes puedan consultar e incluso participar en foros y discusiones acerca del relevamiento que están haciendo o de cómo se espera que completen los formularios provistos. [15]

2.4. El método científico

La ciencia investiga fenómenos de todo tipo, desde fenómenos físicos o biológicos hasta psicológicos o emocionales. Los científicos estudian desde la formación del universo hasta los estadios del desarrollo del intelecto y las emociones humanas o los patrones migratorios de las mariposas. La ciencia es el estudio de casi todo; el estudio de por qué ocurre determinada cosa que es el objeto de estudio. En su nivel más general, la ciencia puede definirse como el estudio de las razones por las cuales las cosas suceden de la manera en que suceden.

El método científico es un simple proceso de tres pasos mediante el cual los científicos investigan el por que de la ocurrencia de un evento o suceso. Empieza con la observación en detalle de algún aspecto del objeto o cosa que es tema de estudio. En cuanto se observa algo que no puede entenderse claramente se elabora una especulación que puede ayudar a explicarlo y luego se busca una manera de probar que esa especulación es acertada.

- Observación

  Antes de que se pueda pensar en una explicación para algo que se investiga, se debe tener conocimientos acerca de los hechos que rodean el fenómeno investigado. Atenerse a los hechos ayuda a revelar la necesidad de encontrar una explicación y aporta indicios acerca de qué es lo que puede estar sucediendo.

- Especulación

  Especular es buscar una posible explicación que explique el cómo o cuándo sucede la cosa en estudio. Cuando algo todavía no tiene una explicación probada, la especulación puede no resultar clara.

- Prueba - Para probar una especulación, es decir, una posible explicación para el objeto o cosa en estudio lo que se hace en primer lugar es buscar algo que ocurra si las circunstancias son apropiadas y la especulación está en lo correcto. Luego se lleva adelante un experimento diseñado para probar que el resultado es el esperado bajo las circunstancias propuestas. Si los resultados del experimento son los previsibles bajo la especulación propuesta, entonces hay buenos motivos para creer que la especulación es la explicación correcta. En cambio si falla se tiene una razón posible para que la explicación no sea la correcta, es decir, que la explicación puede ser equivocada y puede haber buenas razones para modificar la explicación propuesta inicialmente y volver a probar [16]

2.5. Protocolo de recolección de muestras en proyectos de ciencia ciudadana

Los científicos ciudadanos son capaces de ayudar a monitorear plantas y animales en su estado natural, así como otros indicadores como pueden ser alcalinidad del agua o contaminación del aire. La mayoría de los científicos ciudadanos son voluntarios que se involucran en proyectos de investigación, generalmente relacionados con la ecología, porque disfrutan estar al aire libre o porque les preocupan los problemas ambientales y quieren hacer algo al respecto. Generalmente no participan del análisis de la información recolectada, pero son imprescindibles a la hora de la recolección de información en la que los estudios luego se basan. La inclusión de científicos ciudadanos también habilita que la recolección de la información sea en un espacio geográfico amplio o durante períodos de tiempo más extensos que los que permitiría un proyecto de investigación que no incluya ciencia ciudadana.

Si se explica claramente a los científicos ciudadanos qué es lo que deben hacer y cómo deben hacerlo, pueden aprender a usar equipo, leer resultados y recolectar información (o muestras) precisas, confiables y utilizables iguales a las que recolectan los asistentes de campo especializados. Nada de lo hace un científico es tan difícil que no pueda ser hecho por un científico ciudadano si está bien explicado y entrenado. Los científicos que diseñan proyectos de investigación que incluyen ciencia ciudadana deben escribir protocolos que tengan en cuenta a los científicos ciudadanos. Estos protocolos deben limitar lo que se le solicita hacer a los científicos ciudadanos. Es más sencillo solicitar que identifique la ocurrencia o no de 5 o 10 tipos de plantas fácilmente identificables, que solicitar que identifiquen todas las especies de

plantas en un área determinada. El material bibliográfico y la ayuda en el sitio también ayudan a los científicos ciudadanos a la hora de recolectar las muestras.[11] Cuando se diseña un protocolo de recolección puede probarse con un grupo más reducido de científicos ciudadanos o con participantes de otros proyectos, y detectar si la información brindada es clara y sencilla, o si es demasiado compleja o confusa. En ese caso los protocolos de recolección se pueden simplificar, clarificar o incluso modificar hasta que los participantes puedan seguirlos sin inconvenientes. Si se detecta que predomina el informe de los datos positivos (ocurrencia del evento o cosa observada) y los datos negativos también son importantes para el proyecto (no-ocurrencia del evento o cosa observada), este debe estar en material educativo o en el protocolo de recolección de la muestra para que los participantes entiendan que también debe ser informado.[15]

2.6. Ciencia ciudadana y dispositivos móviles

Los dispositivos móviles, en particular los teléfonos celulares se convirtieron en elementos cotidianos. Todos los días vemos usuarios de diferentes ámbitos y edades utilizando sus dispositivos para realizar todo tipo de actividades que van desde participar en redes sociales, organizar sus actividades laborales, realizar trámites y compras, consultar saldos bancarios, etc. Los teléfonos celulares cada vez tienen más presencia entre los usuarios.

Según el INDEC, hasta fines del 2018, 84 de cada 100 argentinos emplea un teléfono celular [17] y según el reporte Mercado Celular Argentino 2019 elaborado por Carrier y Asociados hay 34 millones de celulares en uso en el país. De esta cantidad se observa que un 6 % de los smartphones que hay en funcionamiento en este momento es algún modelo de iPhone, el 93 % pertenece a las marcas que utilizan Android y un 1 % de otras marcas que utilizan otros sistemas operativos (como Windows Phone, Symbian, BlackBerry, etc). En números absolutos, de los 34 millones de celulares en uso que hay en el país, 2.100.000 son iPhone y 31.620.000 tienen sistema operativo Android. [18]

Esta proliferación de tecnologías móviles está enriqueciendo los ambientes urbanos en lo relacionado a sensing, proveyendo herramientas para recolectar datos y creando oportunidades para que cualquier persona interesada pueda involucrarse en actividades científicas. Entre otras cosas, los dispositivos móviles son ideales como soporte para permitir la recolección espontánea de información. [19]

Las nuevas tecnologías influencian el proceso de investigación científica

permitiendo el envío de la información recolectada, optimizando su manejo, automatizando el control de la calidad de la misma y acelerando los tiempos de comunicación. De esta manera, las nuevas tecnologías (entre ellas los dispositivos móviles) y las habilidades relacionadas con las mismas resultan atractivas para un conjunto diverso de participantes, teniendo como desventaja que pueden llegar a marginar a aquellos que no pueden o no se sienten a gusto utilizandolas. [20]

Capítulo 3

Frameworks para proyectos de ciencia ciudadana

3.1. Frameworks

En esta sección se explorarán conceptos básicos de programación orientada a objetos como clases y herencia que son fundamentales para comprender qué es un framework.

Las aplicaciones desarrolladas utilizando programación orientada a objetos están compuestas por una colección de objetos. Estos objetos poseen información sobre sí mismos y comportamiento, denominados atributos y métodos respectivamente. Para construir los objetos que componen la aplicación se utiliza un "molde" que se denomina clase [21]. Todos los objetos son instancias de una clase [22].

A modo de ejemplo se describe un sistema de gestión de pacientes para un consultorio. Se puede modelar una clase Paciente que tenga los atributos nombre, teléfono, correo electrónico, fecha de nacimiento, obra social y un método para agendar un turno. También se puede modelar una clase Médico que tenga los atributos nombre, teléfono, correo electrónico, fecha de nacimiento, matrícula y un método para emitir un turno. Las clases Médico y Paciente tienen atributos en común. Esto permite diseñar una clase Persona que contenga los atributos comunes (nombre, teléfono, fecha de nacimiento) y que las clases Paciente y Médico hereden de la clase Persona y definan sus atributos particulares (obra social en el caso de la clase Paciente y matrícula en la clase Médico).

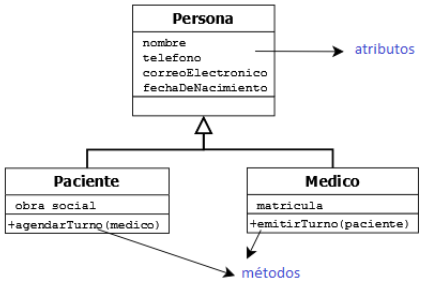

Figura 3.1: Atributos y Métodos de la clase Persona

Ampliando el ejemplo, se requiere que los médicos y los pacientes puedan ser contactados. Pero a los pacientes se los puede contactar por correo electrónico ya que la comunicación nunca es urgente. En cambio a los médicos se los debe contactar por teléfono. Para cumplir con este requerimiento, se agrega en la clase Persona un método abstracto 'contactar()' que no tiene implementación ya que el comportamiento debe ser implementado en las clases que heredan de ella.

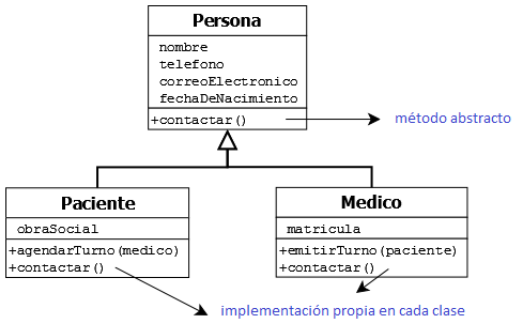

Figura 3.2: Método abstracto en clase Persona

La clase Persona es ahora una clase abstracta ya que contienen uno o más métodos para los cuáles no provee una implementación [21].

Las clases pueden agruparse en librerías para ser reutilizadas en otro proyecto. De esta manera, los componentes se van sumando para crear librerías más complejas que en conjunto con otros componentes reutilizables, diseños, patrones de diseño y estructuras que definen comportamiento logran ser una aplicación a la que sólo le falta la definición propia del dominio en el que se está implementando.

Un framework es una aplicación reutilizable, semi-completa que puede ser especializada para producir aplicaciones a medida [23]. También puede verse como un esqueleto que puede ser configurado por un desarrollador para construir una aplicación [24].

Los frameworks se desarrollan teniendo como objetivo un área en particular y un dominio de aplicación específico [23]. Un framework diseñado para gestionar la agenda de un consultorio médico probablemente permita ser adaptada para gestionar la agenda de otro tipo de consultorio de salud (un consultorio odontológico o psicológico) ya que ambos ámbitos de aplicación comparten la gestión de pacientes y médicos. Pero en el caso de querer desarrollar una aplicación para gestionar muestras de un relevamiento para una investigación científica debe investigarse un framework que se adapte mejor.

Diseñar un framework requiere experiencia y experimentación al igual que lo requiere el diseño de las clases abstractas de sus componentes.[25]

3.1.1. Tipos de Frameworks

Como se mencionó en la sección anterior 3.1, los frameworks son específicos de un dominio de aplicación. Pueden clasificarse de varias maneras dependiendo de si son para construir servicios, interfaces de usuario o aplicaciones. También pueden clasificarse por la manera en que permiten la especialización. Los frameworks de caja blanca son aquellos que utilizan la herencia y la extensión de métodos. Estos frameworks permiten conocer los estados internos de las clases que lo componen. Los frameworks denominados de caja negra utilizan la composición para permitir la especialización. El desarrollador no tiene acceso al estado interno de las clases del framework ni puede extender o agregar métodos. La especialización se realiza proveyendo un componente que respeta un protocolo (o API) de interacción con el framework para lograr el comportamiento específico.

3.1.1.1. Frameworks de caja blanca

Una de las características importantes de un framework es que los métodos definidos por el usuario para extender el comportamiento van a ser invocados desde el interior del framework más que del código de la aplicación del usuario. A menudo hace las veces de programa principal coordinando y secuenciando las actividades de la aplicación. Esa inversión de control le permite al framework servir como esqueleto extensible. El código brindado por los usuarios en los métodos extienden el algoritmo genérico del framework

para una aplicación en particular.

Si el comportamiento específico de una aplicación que utiliza un framework se define agregando métodos a las subclases o a una o más de sus clases, el framework es denominado de Caja Blanca (white-box). Cada método que se agrega a una subclase debe continuar con las convenciones internas que adoptan las superclases.

El principal problema de los frameworks de caja blanca es que cada aplicación requiere la creación de una numerosa cantidad de subclases. Y aunque muchas de estas subclases creadas son simples, es su número lo que para un desarrollador con poca experiencia puede volver difícil comprender el diseño de una aplicación lo suficiente como para modificarla.

Un segundo problema es que un framework de caja blanca puede ser difícil de aprender a utilizar, ya que entender cómo se utliza es lo mismo que entender cómo está construido.[25]

3.1.1.2. Frameworks de caja negra

Otra manera de especializar un framework es incluir en él un conjunto de componentes que sean los que proveen el comportamiento específico de la aplicación. Cada uno de estos componentes debe entender un protocolo en particular. Todos o la mayoría de los componentes pueden tomarse de una librería de componentes. La interfaz entre componentes pueden ser definidas con un protocolo, de esta manera el usuario sólo necesita entender la interfaz externa de los mismos. Este tipo de framework se denomina de caja negra.

Los frameworks de caja negra son más fáciles de aprender a utilizar que los de caja blanca, pero son menos flexibles.

Una manera de caracterizar la diferencia entre un framework de caja blanca y uno de caja negra es observar que en el de caja blanca el estado de cada instancia está disponible de manera implícita en todos los métodos del framework, casi como las variables globales de Pascal. En un framework de caja negra, cualquier información que se pase a las partes constituyentes del framework debe pasarse de manera explícita. Un framework de caja blanca utiliza las reglas de alcance intra-objeto para evolucionar sin forzarlo a subscribirse a un protocolo explícito que podría restringir de manera prematura el proceso de diseño. [25]

3.2. Estado del arte

Luego de detallar los aspectos conceptuales de los frameworks con respecto a su arquitectura y con respecto a la manera en la que permiten la

especialización, en esta sección se exploran seis herramientas, entre plataformas y frameworks, que permiten generar aplicaciones que sirvan de soporte a los proyectos de recolección de ciencia ciudadana.

3.2.1. Sensr

Sensr [19] es una herramienta que permite que interesados sin conocimientos de programación puedan crear aplicaciones para dispositivos móviles que faciliten la recolección de información para ciencia ciudadana.

La documentación que acompaña la herramienta explica las ventajas que los dispositivos móviles poseen para recolectar de manera espontánea información del entorno, pero también advierte que bajo esta simplicidad existe una complejidad técnica y de infraestructura a la hora de desarrollar este tipo de aplicaciones. Contrasta la facilidad de utilización de un dispositivo móvil con la complejidad que el desarrollo de aplicaciones para estos dispositivos representa. Los desarrolladores de aplicaciones móviles tienen la posibilidad de usar directamente el hardware y los sensores que brindan estos dispositivos. Sensr es un framework que consiste de dos partes: Una parte es un sitio web alojado en Amazon Web Services donde los interesados pueden crear y administrar campañas y donde el público puede acceder a la lista de campañas activas junto con visualización de información. La otra parte es una aplicación móvil con la que los voluntarios pueden explorar, suscribir y/o participar en campañas.

3.2.2. Project Noah

Project Noah [26] es una comunidad global para compartir fotos de avistajes de fauna en su hábitat natural. A medida que se comparten fotos, el usuario miembro de la comunidad, empieza a plasmar sus registros de la naturaleza igual que lo haría en un diario o jornal de viaje. Permite el registro particular generando un usuario con correo electrónico y contraseña, y también permite el inicio de sesión por integración a servicios de autenticación como los brindados por Google o con redes sociales. Una vez que se accede con un usuario se pueden ingresar avistajes completando un formulario y agregando una o varias fotografías del avistaje. Entre la información requerida se encuentran geolocalización, fecha, nombre científico (si se conoce), información adicional que se quiera agregar o que no haya sido capturada en la o las imágenes, una descripción del habitat y otras notas. Project Noah permite participar de 'Misiones' que son lo que en Sensr serían campañas: compartir fotos de avistajes con determinadas características. Una misión,

por ejemplo, tiene por título “Aves del Mundo“. Se puede unir a esa misión y cuando se registre un nuevo avistaje se puede seleccionar la opción de agregarlo a la galería de esa misión. También permite aportar en avistajes de otros miembros de la comunidad observaciones propias o bien sugerir una identificación del ejemplar del avistaje. La comunidad tiene miembros que son diferentes al usuario estándar. Se denominan «Rangers» y son los que se encargan de sugerir identificación de especies de otros avistajes, de dar la bienvenida a los nuevos miembros y de compartir avistajes de manera más regular.

3.2.3. EpiCollect

Epicollect [27] es una aplicación web y aplicación móvil gratuita y fácil de utilizar para recolectar información. Los proyectos se crean utilizando el sitio web y luego se descarga una app a un dispositivo móvil para efectuar la recolección de información. La información (incluyendo GPS y multimedia) puede ser recolectada utilizando múltiples dispositivos y puede ser vista en un servidor central (utilizando mapas, tablas y gráficos). También puede ser exportada en formato cvs y json. La aplicación móvil está disponible para Android (6+) y iOS (12+)

Los usuarios pueden ingresar a la aplicación con su cuenta de Google, su cuenta de Apple o bien creando una cuenta con su correo electrónico. Para crear un proyecto se debe ingresar el nombre, una descripción, seleccionar el tipo de acceso entre público (cualquier usuario puede colaborar) o privado (sólo usuarios seleccionados por el administrador del proyecto pueden colaborar) y definir como mínimo un formulario para recolectar la información. Permite toda una variedad de “preguntas“ que pueden ser de tipo texto, numérico, telefóno, foto, video, código de barras y varios tipos más entre los disponibles. Cada formulario de pregunta tiene una opción para conectar a la siguiente pregunta que permite formato condicional, es decir, en caso de que la respuesta cumpla determinadas condiciones se ejecuta un “salto“ a un formulario determinado. En la edición del proyecto, permite modificar el tipo de acceso nuevamente, el estado, la visibilidad y permite asignarle una categoría entre las predefinidas. La información de las entradas de los usuarios se puede visualizar en una tabla, en un mapa o descargarse.

3.2.4. CitSci

CitSci.org [28] se define como una plataforma global para soporte para la Ciencia Ciudadana. La aplicación web CitSci permite que un usuario regis-

trado cree un proyecto de Ciencia Ciudadana definiendolo con un título, una descripción breve de qué consiste el proyecto, sus objetivos y tareas. Esta información la utiliza como portada de presentación del proyecto. También permite definir si la colaboración va a ser pública o privada. Si la colaboración es pública, con que un usuario registrado quiera unirse y registrar observaciones alcanza. En el caso de la colaboración privada, los usuarios deben solicitarle al administrador del proyecto su permiso para unirse y aportar la información que recolectan. La visibilidad del proyecto también puede ser pública y salir en la lista de proyectos de la plataforma, o puede ser privada y no salir en el listado. Permite integrar con Zooniverse para el procesamiento de imágenes y permite agregar material relacionado como pueden ser documentos que amplíen la información de los objetivos, o la recolección de muestras, así como recursos gráficos y cualquier material extra que quiera anexarse. Una vez creado un proyecto de ciencia ciudadana, para que los colaboradores pueden ingresar información debe definirse una plantilla con las opciones de recolección disponibles. La plantilla debe tener un nombre, instrucciones de la actividad que debe realizar el ciudadano científico, el formato de fecha y el formato de geolocalización que puede ser latitud y longitud, recorrido o ambas. Una vez determinada esta información básica y obligatoria, se pueden agregar otras opciones al formulario de recolección de información que pueden ser:

- Imagen
- Fecha/Hora
- Número
- Título
- Opción Simple - selección de una opción (radio button)
- Lista Desplegable - Selección de una opción mediante lista desplegable
- Texto
- Organismo

Cada una de estas opciones permiten su configuración particular. Por ejemplo, la opción Lista Desplegable permite definir un título, una ayuda en forma de pista que aparece, los items que componen la lista y si es opcional u obligatoria. Por cada una de las opciones que se agreguen a la recolección de la muestra la aplicación móvil genera un paso para que el ciudadano

científico complete. Las plantillas definidas pueden modificarse y también pueden definirse otras plantillas de recolección de muestras. Una vez definidas las opciones requeridas para la recolectar una muestra, la información puede ingresarse por la aplicación web, o puede descargarse la aplicación móvil que provee CitSci para Android o Ios. La aplicación móvil permite que el científico ciudadano se una a los proyectos publicados o recolecte muestras en los proyectos que creó o a los que se unió. Las muestras que recolecta quedan en un listado de muestras donde por cada una tiene disponibles las acciones de editar, eliminar o subir la información a la plataforma. La plataforma brinda una herramienta que permite administrar las muestras recolectadas, listarlas y por cada una de ellas visualizarlas, editarlas o eliminarlas. También provee un foro asociado al proyecto.

3.2.5. Spotteron

Spotteron [29] provee Aplicaciones y Servicios de Ciencia Ciudadana para proyectos científicos e instituciones. Su equipo de desarrollo se enfoca en el diseño, el profesionalismo técnico, la confiabilidad y la interacción con el usuario. Este equipo se especializa en crear soluciones a medida para Ciencia Ciudadana que incluyen aplicaciones móviles, sitio web y herramientas para el Análisis y la Visualización de la información.

Ofrecen, en conjunto con la plataforma Spotteron para la Ciencia Ciudadana un sistema de aplicaciones personalizables y económicamente accesibles, aplicables en áreas de la Ciencia Ciudadana, de protección ambiental y monitoreo de voluntarios. Desarrollan aplicaciones móviles independientes en Ios y Android, aplicaciones web interactivas como mapas o juegos para ciencia ciudadana y proveen una plataforma estable y confiable para los mismos.

El objetivo es proveer un sistema con mantenimiento y mejoras constantes y un servicio profesional y confiable para la Ciencia Ciudadana, para la Conservación del Medio Ambiente y Proyectos Sociales. También considerar a los usuarios y albergar el crecimiento de la comunidad y fomentar la interacción entre ciudadanos y científicos.

3.2.6. Zooniverse

Zooniverse [8] es la más grande y popular plataforma de ciencia ciudadana para investigaciones científicas. El objetivo de Zooniverse es llevar a cabo investigaciones que no serían posibles o prácticas de otra manera. Aunque Zooniverse opera sobre todo en su aplicación web [30] también provee una

aplicación móvil para que los científicos ciudadanos puedan hacer su colaboración desde sus dispositivos móviles. La plataforma provee una herramienta de construcción de proyectos, Project Builder, que permite configurar tres secciones de los proyectos de investigación

- Proyecto (Project)

  Galaxy Zoo, un proyecto que vive en la plataforma desde hace más de diez años, es un buen ejemplo de proyecto que puede construirse utilizando la herramienta Project Builder. En este proyecto los científicos ciudadanos clasifican galaxias de acuerdo a su forma observando fotografías del espacio y dibujando digitalmente sobre ellas. Un proyecto tiene nombre, descripción, un avatar (una imagen que lo identifica), colaboradores y otros atributos configurables más.

- Secuencia de Tareas (Workflows) Workflow es la secuencia de tareas que deben realizar los científicos ciudadanos. Un workflow tiene un título que sirve para identificarlo, un número de versión que se utiliza para trazar los cambios y tareas que pueden ser de tres tipos: responder preguntas, transcribir o dibujar.

  - Pregunta Las tareas de tipo pregunta pueden ser preguntas de única respuesta o de respuesta múltiple. Las preguntas pueden marcarse como 'requeridas' y en este caso el científico ciudadano no puede avanzar hacia la siguiente tarea hasta no haber ingresado una respuesta.
  - Dibujar Las tareas de este tipo requieren que el científico ciudadano marque o dibuje de manera digital sobre una muestra del conjunto de muestras definido. La marca o dibujo que realice va a terminar siendo de un tipo entre los predefinidos (forma libre, rectángulo, elipse, círculo y otros disponibles) y será para señalar alguna cosa en la muestra que se está analizando. La herramienta dibujo permite ingresar también etiquetas o color aparte del marcado en sí.
  - Transcripción Las tareas de transcripción requieren que los científicos ciudadanos transcriban una porción de texto que perciben en un archivo multimedia. Puede solicitarse una transcripción fiel que respete la ortografía original o bien una que modernice y corrija el texto original. Las herramientas de transcripción también permiten marcar o hacer una lista de palabras clave asociadas al texto o porción de texto que se está analizando.

Una vez definidas las tareas se está en condiciones de definir el workflow. Una de esas tareas se marca como 'Primer Tarea'. Luego, utilizando un selector de opciones se configura que tarea de las definidas será la siguiente. En las tareas de tipo 'pregunta' se puede especificar una tarea distinta para cada respuesta disponible. La secuencia de tareas que cumplen determinadas condiciones se pueden habilitar para estar disponibles desde la aplicación móvil que provee Zooniverse y permitir que los científicos ciudadanos colaboren desde sus dispositivos. Diseñar un workflow que se puede habilitar para la aplicación móvil permite incrementar la cantidad de colaboraciones que recibe el proyecto.

- Conjunto de Muestras (Subject Set) Un conjunto de muestras es un grupo de datos que debe ser clasificado. Consiste generalmente de archivos multimedia como imágenes, archivos de sonido o videos, sobre los cuáles los científicos ciudadanos deben marcar cosas que perciban. Se puede tener un conjunto de muestras sobre el que se pueden seguir agregando muestras a lo largo del tiempo, o varios de diferentes momentos o lugares. Se pueden tener diferentes conjuntos de muestras para diferentes secuencias de tareas aunque no es necesario.

3.2.7. Conclusión

La herramienta Sensr permite desarrollar una campaña de recolección de información más una aplicación móvil para que los científicos ciudadanos que quieran participar de una campaña puedan hacerlo mediante esa aplicación. El autor de la campaña, completando formularios en el sitio web y arrastrando componentes para definir el contenido y las validaciones, diseña la aplicación móvil que se generará. Esta aplicación móvil sólo está disponible para dispositivos con sistema operativo iOS y la administración de la información se puede realizar sólo a través del sitio web. Al no ser iOS el sistema operativo más popular entre los usuarios de dispositivos móviles de Argentina, como se mencionó en la sección 2.6, es una alternativa con menos alcance que la que proponemos. Project Noah podría calificar como herramienta social para colaborar con la Ciencia Ciudadana. La posibilidad de compartir avistajes a través de las redes sociales, puede llegar a más interesados que quieran formar parte de esta comunidad o puede poner en contacto a personas interesadas en la vida al aire libre. Aunque asiste de manera intuitiva, el ingreso de la información de geolocalización del avistaje es en forma de coordenadas, latitud y longitud. Si un usuario quisiera

subir varios avistajes de un viaje, debería recordar dónde hizo cada uno y señalarlos en un mapa. También está limitado al avistaje de fauna. Epicollect es una herramienta simple e intuitiva para construir proyectos de recolección de información mediante cuestionarios. El proyecto está financiado por Wellcome Trust Foundation y está implementado con tecnologías open source. Ofrece una API para desarrolladores que permite consultar la información mediante servicios o integrar los cuestionarios a una app propia, previo registro de la aplicación en el proyecto Epicollect. Epicollect siempre está intermediando entre los científicos ciudadanos y los administradores del proyecto. Las herramientas que ofrece, las ofrece siempre a través de la plataforma. CitSci permite implementar proyectos de recolección de ciencia ciudadana de manera sencilla y se integra con Zooniverse y con SciStarter, lo que permite diversificar las muestras y los colaboradores. La aplicación móvil que ofrece es propia y compartida entre todos los proyectos que conviven en la plataforma. No permite personalizar muchas características del proyecto más allá de la foto de portada. Spotteron es un equipo y una plataforma que está ampliamente utilizada sobre todo en Europa. El equipo que hace el desarrollo y mantenimiento de las aplicaciones está integrado por profesionales con experiencia en áreas como marketing y desarrollo. De esta manera el equipo puede brindar servicios y soporte a precios accesibles. No se integran a otras aplicaciones de Ciencia Ciudadana ni proveen API's de integración. Zooniverse brinda herramientas que permiten realizar una configuración exhaustiva de un proyecto. Aunque posee más riqueza que las otras herramientas exploradas en cuanto a definición de tareas y secuencia de las mismas, su aplicación móvil también es compartida entre los proyectos, de la misma manera que ofrece CitSci.

Capítulo 4

El entorno de Android

Android es un sistema operativo móvil desarrollado por Google, basado en Kernel de Linux. Está pensado para diferentes dispositivos móviles con pantalla táctil como teléfonos inteligentes, tablets, relojes inteligentes (Wear OS), automóviles (Android Auto) y televisores (Android TV).

Inicialmente fue desarrollado por Android Inc., adquirido por Google en 2005 y presentado en 2007 junto con la fundación del Open Handset Alliance (un consorcio de compañías de hardware, software y telecomunicaciones) para avanzar en los estándares abiertos de los dispositivos móviles. El código fuente principal de Android se conoce como Android Open Source Project (AOSP), que se licencia principalmente bajo la Licencia Apache.

Para escribir aplicaciones (app) para Android, se pueden usar los lenguajes Java, Kotlin y C++. Las herramientas de Android SDK compilan el código, junto con los archivos de recursos y datos, en un APK: un paquete de Android, que es un archivo de almacenamiento con el sufijo .apk. Un archivo APK incluye todos los contenidos de una app de Android y es el archivo que usan los dispositivos con tecnología Android para instalar la app.

Cada app de Android reside en su propio entorno aislado de seguridad y está protegida por las siguientes características de seguridad de Android:

- El sistema operativo Android es un sistema Linux multiusuario en el que cada app es un usuario diferente.
- De forma predeterminada, el sistema le asigna a cada app un ID de usuario de Linux único (solo el sistema utiliza el ID y la app lo desconoce). El sistema establece permisos para todos los archivos en una app de modo que solo el ID de usuario asignado a esa app pueda acceder a ellos.

- Cada proceso tiene su propia máquina virtual (VM), por lo que el código de una app se ejecuta de forma independiente de otras apps.
- De forma predeterminada, cada app ejecuta su propio proceso de Linux. El sistema Android inicia el proceso cuando se requiere la ejecución de alguno de los componentes de la app y, luego, lo cierra cuando el proceso ya no es necesario o cuando el sistema debe recuperar memoria para otras apps.

De esta manera, el sistema Android implementa el principio de mínimo privilegio. Es decir, de forma predeterminada, cada app tiene acceso solo a los componentes que necesita para llevar a cabo su trabajo y nada más. Esto crea un entorno muy seguro, en el que una app no puede acceder a partes del sistema para las que no tiene permiso. Sin embargo, hay maneras en las que una app puede compartir datos con otras apps y en las que puede acceder a servicios del sistema solicitando permiso para acceder a datos del dispositivo, como los contactos de un usuario, los mensajes de texto, el dispositivo de almacenamiento (tarjeta SD), la cámara, la conexión Bluetooth, etc. El usuario debe conceder de manera explícita estos permisos.

Una app en Android esta compuesta por uno o varios componentes de app. Los componentes de app son los bloques de compilación fundamentales, y cada uno es un punto de entrada a través del cual el sistema o un usuario puede entrar a la app. Algunos dependen unos de otros y hay 4 tipos diferentes de componentes de app. Cada tipo sirve para diferentes propósitos y tienen diferentes ciclos de vida que definen como se crea y destruye el componente. Los 4 tipos de componentes de app son:

- **Activities:** Una Activity es un punto de entrada para interactuar con el usuario. Representa una simple pantalla con interfaz de usuario.
- **Services:** Un Service es un punto de entrada de propósito general para mantener una app corriendo en segundo plano por múltiples razones. Es un componente que corre en segundo plano para realizar operaciones de larga duración o procesar trabajos para un proceso remoto. Un Service no proporciona una interfaz de usuario.
- **BroadcastReceivers:** Un BroadcastReceiver es un componente que habilita al sistema a entregar eventos a una app fuera del flujo de usuario habitual, permitiendo a la app responder a una gran variedad de anuncios del sistema. Como los BroadcastReceivers son otro punto de entrada a una app, el sistema puede entregar eventos de difusión a apps que incluso no se estén ejecutando en ese momento.

- **ContentProviders:** Un ContentProvider administra un conjunto de datos de la app que se pueden compartir y que se almacenan en el sistema de archivos, en una base de datos SQLite, en la web, o en cualquier otro almacenamiento persistente. A través del ContentProvider, otras apps pueden consultar o modificar los datos si el ContentProvider lo permite.

Más adelante se explican con más detalle los componentes que influyeron en el desarrollo de Samplers.

Un aspecto único del diseño del sistema Android es que cualquier app puede iniciar un componente de otra app (por medio de una petición al sistema). Por ejemplo, si se desea tomar una foto con la cámara del dispositivo, seguramente ya hay otra app que lo hace, y se puede llamar a esa app en lugar de desarrollar una Activity que tome una foto. [31]

4.1. Activities

La clase Activity es un componente clave de una app para Android, y la forma en que se inician y se crean las Activities es una parte fundamental del modelo de aplicación de la plataforma. A diferencia de los paradigmas de programación en los que las apps se inician con un método main(), el sistema Android inicia el código en una instancia de Activity invocando métodos de devolución de llamada específicos que corresponden a etapas específicas de su ciclo de vida.

La experiencia con la app para dispositivos móviles difiere de la versión de escritorio, ya que la interacción del usuario con la app no siempre comienza en el mismo lugar. En este caso, no hay un lugar específico desde donde el usuario comienza su actividad. Por ejemplo, si abres una app de correo electrónico desde la pantalla principal, es posible que veas una lista de correos electrónicos. Por el contrario, si usas una app de redes sociales que luego inicia tu app de correo electrónico, es posible que accedas directamente a la pantalla de la app de correo electrónico para redactar uno.

La clase Activity está diseñada para facilitar este paradigma y se debe heredar de ella para implementar las Activities de una app.

Cuando una app invoca a otra, la app que realiza la llamada invoca una Activity en la otra, en lugar de a la app en sí. De esta manera, la Activity sirve como el punto de entrada para la interacción de una app con el usuario

Una Activity proporciona la ventana en la que la app dibuja su interfaz de usuario (IU). Por lo general, esta ventana llena la pantalla, pero puede ser más pequeña y flotar sobre otras ventanas. Generalmente, una Activity

implementa una pantalla en una app. Por ejemplo, una Activity de una app puede implementar una pantalla Preferencias mientras otra implementa una pantalla Seleccionar foto.

Cada Activity administra su propio diseño, que se carga desde un archivo XML en donde se definen los diferentes elementos de diseño (existen herramientas en las que se puede definir el diseño visualmente y éste se traduce al formato XML). Un diseño define la estructura de una interfaz de usuario que está compuesto por una jerarquía de objetos View y ViewGroup. Una View suele mostrar un elemento que el usuario puede ver y con el que puede interactuar; suelen llamarse 'widgets' y pueden ser una de las muchas subclases, como Button (que muestra un botón) o TextView (que muestra un texto). Por su parte, ViewGroup es un contenedor invisible que define la estructura de diseño de View y otros objetos ViewGroup; se denominan generalmente 'layouts' y pueden ser de muchos tipos que proporcionan una estructura diferente, como LinearLayout (que posiciona los elementos de forma linear, ya sea horizontal o verticalmente) o RelativeLayout (que posiciona los elementos en relación a otros elementos o a los bordes de la pantalla). También se puede crear el diseño de una Activity declarando instancias de elementos de diseño en tiempo de ejecución.

El sistema Android administra las Activities como una pila (stack[1]). Cuando se crea una nueva Activity, esta es puesta en la cima de la pila actual y se convierte en la Activity actual; la Activity anterior siempre permanece abajo en la pila, y no vuelve a estar en primer plano de nuevo hasta que la nueva Activity finalice, y puede haber una o muchas Activities en la pila. Es por esto que a lo largo de su vida útil, una Activity pasa por varios estados. Para administrar las transiciones entre estados, se deben usar una serie de devoluciones de llamadas.

Una Activity tiene básicamente cuatro estados:

- Si una Activity esta en primer plano de la pantalla (en la posición más alta de la pila) está en estado 'activa' o 'running'. Es generalmente la Activity con la que el usuario está interactuando.

- Si una Activity ha perdido el foco pero todavía se muestra al usuario, entonces está en estado 'visible'. Esto sucede por ejemplo cuando hay otra Activity encima pero que no cubre toda la pantalla. La Activity en este estado esta completamente 'viva' (conserva todos sus estados e información y se mantiene vigente para el administrador de ventanas).

[1]Stack en inglés, una lista ordenada con acceso a sus elementos de tipo «último en entrar, primero en salir»

- Si una Activity esta cubierta completamente por otra, entonces esta en estado 'detenida' u 'oculta'. Aún conserva todos sus estados e información, pero ya no es visible al usuario por lo que su ventana esta oculta, y podría ser destruida por el sistema si se necesita memoria para otro propósito.
- El sistema puede bajar de la memoria a la Activity solicitándole que finalice, o simplemente destruyendo su proceso, pasándola al estado 'destruida'. Cuando se necesita mostrarla al usuario de nuevo, la Activity debe ser restaurada completamente y se debe restaurar su estado anterior.

El siguiente diagrama muestra los estados más importantes por los que pasa una Activity. Los rectángulos representan métodos de respuesta a llamada que se pueden implementar para realizar operaciones cuando la Activity se mueve entre un estado y otro. Los óvalos representan los estados más importantes por los que pasa una Activity. [31]

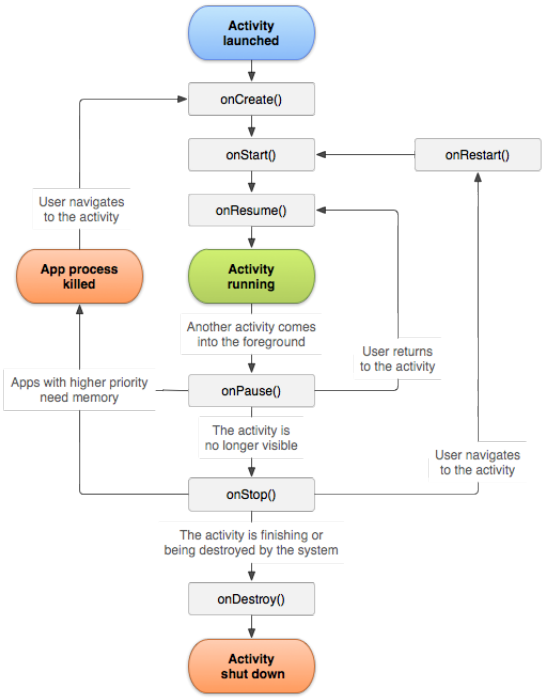

Figura 4.1: Ciclo de vida de una Activity

4.2. Fragments

Un Fragment representa una sección modular y reusable de la IU de una Activity, el cual se puede agregar o quitar mientras la Activity se esté ejecutando (algo así como una "sub-Activity"). Se pueden combinar varios Fragments en una sola Activity para crear una IU multipanel y volver a usar un Fragment en diferentes Activities.

Un Fragment define y administra su propio diseño, su propio ciclo de vida, y recibe sus propios eventos de entrada. Un Fragment no es un componente de app, por lo que no es un punto de entrada a la app como lo es una Activity. Un Fragment no puede "vivir"por si mismo, debe ser hospedado por una Activity, y el ciclo de vida del Fragment se ve afectado directamente por el ciclo de vida de la Activity anfitriona. Por ejemplo, cuando la Activity está pausada, también lo están todos sus Fragments, y cuando la Activity se destruye, lo mismo ocurre con todos los Fragments. Sin embargo, mientras una Activity se está ejecutando (está en el estado 'running' del ciclo de vida), se puede manipular cada Fragment de forma independiente, por ejemplo, para agregarlo o quitarlo. Cuando se realiza una transacción de Fragments como esta, también se pueden agregar a una pila administrada por la Activity; cada entrada de la pila de la Activity es un registro de la transacción de Fragments realizada. La pila le permite al usuario invertir una transacción de Fragments (navegar hacia atrás) al presionar el botón Atrás del dispositivo móvil.

El siguiente diagrama muestra los estados mas importantes por los que pasa un Fragment. Los rectángulos representan métodos de respuesta a llamada que se pueden implementar para realizar operaciones cuando el Fragment se mueve entre un estado y otro. Los óvalos representan los estados más importantes por los que pasa un Fragment.

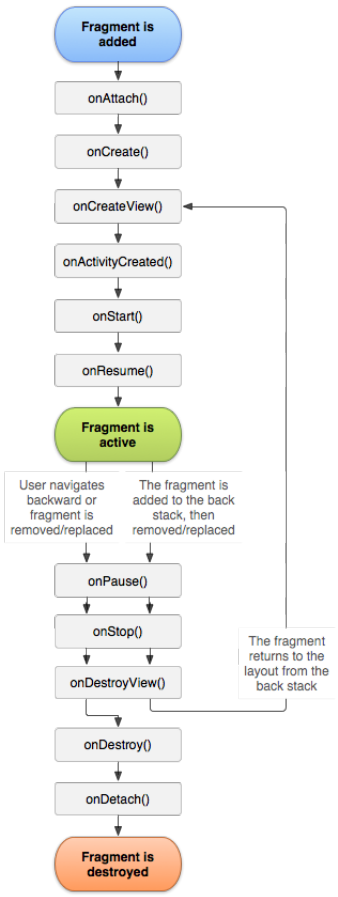

Figura 4.2: Ciclo de vida de un Fragment

Los Fragments introducen modularidad y reusabilidad en la IU de una Activity, permitiendo dividir la IU en módulos más discretos. Se pueden usar múltiples instancias de un mismo Fragment en la misma Activity, en varias Avtivities, o incluso dentro de otro Fragment. Por esta razón se deben diseñar los Fragments solamente con la lógica necesaria para manejar su propia IU, de forma que sean independientes, evitando la dependencia con otros Fragments. Con esto en mente, las Activities son el lugar ideal para poner los elementos globales de la IU, como por ejemplo una barra de navegación. Por el contrario, los Fragments son mejor lugar para definir y manejar la IU de una sola pantalla, o una sección de la misma.

Por ejemplo, en una app que responde a varios tamaños de pantalla, la app debería mostrar en pantallas pequeñas una barra de navegación con

íconos y una lista lineal con el contenido. En cambio, en pantallas más grandes, debería mostrar un área de navegación más detallada y el contenido en forma de grilla para aprovechar mejor el espacio. Administrar todas estas variaciones en la Activity puede resultar engorroso. Separar los elementos de navegación del contenido hace este proceso más manejable. La Activity es responsable de mostrar la IU de navegación correcta mientras que el Fragment muestra el contenido con la disposición adecuada.

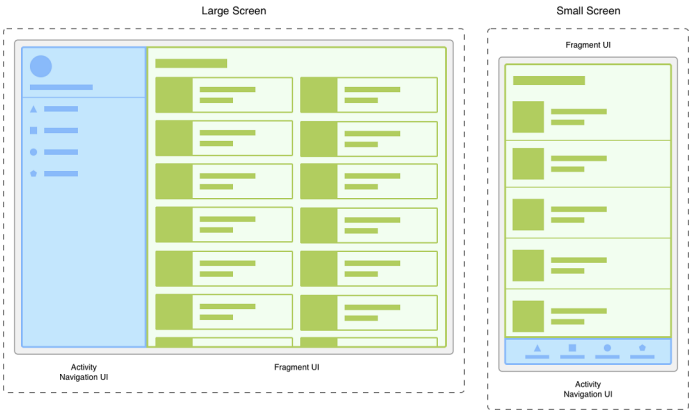

Figura 4.3: Dos versiones de la misma pantalla en diferentes tamaños de pantalla

Para crear un Fragment, se debe crear una subclase Fragment (o una subclase existente de ella). La clase Fragment tiene un código que se asemeja bastante a una Activity. Contiene métodos de devolución de llamada similares a los de una actividad, como onCreate(), onStart(), onPause() y onStop(), aunque agrega otros para controlar cuando se crea y destruye su IU, o cuando se adjunta el Fragment a la Activity anfitriona. [31]

4.3. Services

Un Service es un componente de app que puede realizar operaciones de larga ejecución en segundo plano y que no proporciona una interfaz de usuario. Una vez iniciado, un Service puede continuar ejecutándose en segundo plano aún cuando el usuario cambie a otra app. Además, un componente puede enlazarse con un Service para interactuar con él e incluso realizar una comunicación entre procesos (IPC por sus siglas en inglés). Por ejemplo, un Service puede manejar transacciones de red, reproducir música, realizar I/O

de archivos o interactuar con un ContentProvider, todo en segundo plano. Un Service se ejecuta en el subproceso principal del proceso que lo contiene; el Service no crea su propio subproceso ni se ejecuta en un proceso separado (a menos que se especifique lo contrario), por lo que parar realizar un trabajo que consume más CPU u operaciones de bloqueo (como reproducción MP3 o funciones de red), se debe crear un subproceso nuevo dentro del Service para completar ese trabajo.

Hay tres tipos de Service:

- **Primer plano:** Un Service en primer plano realiza una operación que el usuario puede notar. Por ejemplo, una aplicación de audio usa un Service en primer plano para reproducir una pista de audio. Los Services en primer plano deben mostrar una notificación para que el usuario sea consciente de que el Service se está ejecutando. Estos Services continúan ejecutándose incluso si el usuario deja de interactuar con la aplicación.

- **Segundo plano:** Un Service en segundo plano realiza una operación que el usuario no nota directamente. Por ejemplo, si una aplicación usa un Service para comprimir su almacenamiento, suele tratarse de un Service en segundo plano.

- **Enlace:** Un Service es de enlace cuando un componente de app se vincula a él llamando a bindService(). Un Service de enlace ofrece una interfaz cliente-servidor que permite que los componentes interactúen con el Service, envíen solicitudes, reciban resultados e incluso lo hagan en distintos procesos con la comunicación entre procesos (IPC). Un Service de enlace se ejecuta solamente mientras otro componente de aplicación está enlazado a él. Se pueden enlazar varios componentes con el mismo Service a la vez, pero cuando todos ellos se desenlazan, el Service se destruye. [31]

4.4. BroadcastReceivers

Las apps de Android pueden enviar o recibir mensajes de emisión desde el sistema de Android y otras apps para Android, de forma similar al patrón de diseño de publicación y suscripción. Estas emisiones se envían cuando ocurre un evento de interés. Por ejemplo, el sistema Android envía emisiones cuando ocurren diferentes eventos del sistema, como cuando este se inicia, cuando el dispositivo comienza a cargarse o cuando se conecta a una

red de Internet. Las apps también pueden enviar emisiones personalizadas, por ejemplo, para notificar a otras apps sobre algo que podría interesarles (como cuando se descargaron algunos datos nuevos). Las apps pueden registrarse para recibir emisiones específicas. Cuando se envía una emisión, el sistema redirige automáticamente las emisiones a las apps que se suscribieron para recibir ese tipo de emisión particular. Un BroadcastReceiver es un componente de app que permite recibir esos mensajes de emisión, incluso si la app no se está ejecutando en ese momento, ya que estos mensajes se generan fuera del flujo de usuario habitual. [31]

Capítulo 5

Samplers: Framework Android

5.1. Propuesta general

5.1.1. Descripción del problema

En la descripción del Estado del Arte (Sección 3.2), se describen algunas herramientas que sirven de soporte a los científicos para incluir ciencia ciudadana en sus proyectos, y facilitar la recolección de la información o la recolección de muestras a través de formularios web o de aplicaciones para dispositivos móviles. Estas herramientas de soporte existen porque la simplicidad de utilización de los dispositivos móviles es ideal para que los científicos ciudadanos participen en la recolección de muestras, pero el desarrollo de estas aplicaciones puede ser una limitante para algunos proyectos por su complejidad técnica. [19] Para evitar que el desarrollo de una aplicación móvil sea limitante a la hora de incluir ciencia ciudadana en proyectos de investigación, se propone un framework que permita la creación de una aplicación móvil de manera sencilla para la recolección de muestras realizada por científicos ciudadanos. El objetivo principal es permitir crear una aplicación móvil de ciencia ciudadana sin tener conocimientos de programación. Por ejemplo, desde una página web armar el protocolo de recolección de las muestras y con el mismo poder generar una aplicación móvil que sirva para tomar las muestras siguiendo dicho protocolo.

El protocolo de recolección debe estar compuesto por los diferentes pasos necesarios para tomar la muestra con la aplicación. Estos pasos deben permitir:

- capturar una foto, un video o un audio.
- tomar la posición del GPS o grabar un recorrido con el GPS.
- contestar una pregunta con respecto a la muestra. Esta pregunta puede tener una o múltiples respuestas posibles.
- introducir anotaciones (texto).
- seleccionar una fecha o una hora.
- mostrar información de orientación para la toma de la muestra.

Las muestras recolectadas con la aplicación deben ser enviadas por Internet a un servidor web previamente configurado para dicho propósito.

Se decidió que la aplicación generada debe ser una aplicación nativa, y no una solución web por ejemplo, para poder aprovechar mejor las características de los dispositivos móviles (cámara, GPS, etc.). Además se debe contemplar que al momento de tomar una muestra es posible que no se cuente con conexión a Internet, pero igualmente se debe permitir la toma de la muestra y se la debe almacenar en el dispositivo móvil hasta que haya conexión a Internet y pueda ser enviada al servidor web.

La aplicación generada servirá para tomar muestras siguiendo el protocolo de recolección especificado, almacenarlas y empaquetarlas en el dispositivo móvil hasta que puedan ser enviadas al servidor web.

Además se deberá contemplar algún mecanismo para poder mostrar ayuda para el usuario final de la aplicación (el científico ciudadano[1]) que sirviera de orientación para tomar la muestra.

También se debe contemplar algún mecanismo de identificación del usuario que toma las muestras, con usuario y contraseña, o con alguna red social (Facebook, Twitter, Instagram, Google, etc.) para así poder darle una devolución mostrándole información, o incentivarlo para que siga participando del proyecto a través de gamificación por ejemplo.

[1]Se diferencia al usuario de Samplers, que es el que desea generar una app para recolectar muestras en un proyecto de ciencia ciudadana, del usuario final de la app, que es quien va a usar la app generada, que sería el científico ciudadano.

5.1.2. Alcance de la solución propuesta

En un principio, Samplers se pensó para que un científico pudiera crear su propia aplicación móvil de ciencia ciudadana sin tener conocimientos de programación. La idea inicial era que, mediante una aplicación web, el científico pudiera armar el protocolo de recolección de las muestras de manera visual e intuitiva, y se generara un archivo de configuración para el framework de este trabajo: Samplers. Con el archivo de configuración se pasaría a Samplers y se generaría la app para Android (el archivo APK para instalarla). En este trabajo se presenta una solución que supone que el archivo de configuración ya viene armado, y se plantea la aplicación web que genera dicho archivo como un trabajo a futuro.

Como se mencionó antes se requerían aplicaciones nativas, y de los 3 sistemas operativos móviles más usados en ese momento (Android, iOS y Windows Phone) se decidió optar por Android, que era el sistema operativo móvil más usado en ese momento, y se plantea como trabajo futuro el desarrollo en otras plataformas. Según una estadística elaborada por Gartner Inc., empresa consultora y de investigación de las tecnologías de la información, sobre las ventas de smartphones a nivel mundial en el último trimestre de 2016[32], mas del 80 % de las mismas fueron de celulares con Android, y ese porcentaje fue creciendo hasta obtener una cuota del mercado del 88 % a nivel mundial a mediados de 2018. En la actualidad, en Argentina esa cuota de mercado es mas grande, llegando al 93 % según datos publicados por Carrier y Asociados, estudio profesional dedicado a la información y el análisis de mercado para productos y servicios vinculados a internet, en su reporte Mercado Celular Argentino 2019[18].

Para desarrollar aplicaciones móviles para la plataforma Android, el entorno de desarrollo integrado (IDE por sus siglas en inglés) oficial es Android Studio[33], por lo que Samplers utiliza el mismo para la generación de la app. Android Studio ha sido publicado de forma gratuita bajo Licencia Apache 2.0 y está disponible para las plataformas Microsoft Windows, MacOS y GNU/Linux.

5.1.3. Descripción de la solución propuesta: Samplers

Samplers es un framework que permite construir, de manera sencilla, aplicaciones Android (apps) para recolectar muestras en proyectos de Ciencia Ciudadana. Brinda una solución simple al problema de la recolección de la muestra aprovechando las funcionalidades de los dispositivos móviles.

Configurando el Workflow (que representa el protocolo de recolección en Samplers) y unos parámetros más, con Samplers se puede generar una app lista para ejecutarse en un dispositivo móvil Android. Esta app generada sirve para tomar muestras siguiendo el Workflow especificado, y las almacena en el dispositivo móvil hasta que puedan ser enviadas a un servidor web previamente configurado.

Para configurar el Workflow y los demás parámetros que necesita Samplers para generar la app, se debe completar un archivo de configuración llamado SamplersConfig.json.

Una vez configurado el archivo, Samplers generará el código fuente de la aplicación móvil para Android. El código queda listo para compilar y ejecutar en un dispositivo móvil con Android o en un emulador virtual de los que provee Android Studio.

Samplers está compuesto por dos elementos: una librería con las clases y demás recursos necesarios para crear la app y un script en Gradle[2] para procesar el archivo de configuración.

La librería es un archivo .aar, que es similar a un archivo .jar de Java, pero especifico para Android. En el mismo están todas las clases y los archivos de recursos de Android (como archivos XML con los diseños de IU de los Fragments y Activities, iconos, imágenes, etc).

El script de Gradle es el que procesa el archivo de configuración y genera el código fuente de la app correspondiente usando las clases y recursos de la librería. Este script se ejecuta en Android Studio (como se explica en la sección 6.2) y crea las clases necesarias para la app, como ser entre otras cosas, una Activity principal que muestra un texto de bienvenida con un botón para tomar una muestra. También instancia un Workflow que se usa para llamar a la Activity encargada de tomar las muestras. Todo esto se hace siguiendo lo especificado en el archivo de configuración.

Por ejemplo, si se desea hacer un relevamiento de las zonas con presencia del mosquito transmisor del Dengue (Aedes Aegypti), se puede crear una

[2]Gradle es un sistema de automatización de construcción de código de software que construye sobre los conceptos de Apache Ant y Apache Maven e introduce un lenguaje específico del dominio basado en Groovy. El sistema de compilación de Android Studio está basado en Gradle. https://gradle.org/

aplicación móvil para solicitar a los científicos ciudadanos que tomen fotos de los mosquitos que encuentren con las características de dicho mosquito. Además de la foto, también se puede solicitar la posición del GPS del dispositivo móvil. De esta manera se podría identificar al mosquito con la foto, y con la posición GPS armar un mapa de los avistajes. Para recolectar estas muestras se podría definir el protocolo de recolección de la siguiente manera:

1. Mostrar al científico ciudadano las características del mosquito Aedes Aegypti para que las pueda comparar con el mosquito encontrado (para evitar el envío de fotos innecesarias).

2. Pedir al científico ciudadano que tome una foto desde arriba del mosquito.

3. Pedir al científico ciudadano que tome una foto del costado del mosquito.

4. Pedir al científico ciudadano que tome la posición del GPS

Para representar este protocolo de recolección se debería configurar el Workflow en el archivo de configuración para Samplers como se muestra en el código 5.2

```
{
  "project": {
    "app_path": "app/src/main/java/com/example/myApplication/",
    "package_name": "com.example.myApplication"
  },
  "application": {
    "title": "App de prueba",
    "welcomeMessage": "Bienvenido a la app de prueba!",
    "networkConfiguration": {
      "url": "https://192.168.0.10/samplers/upload.php",
      "paramName": "sample",
    },
    "googleMaps_API_KEY": "YOUR_GOOGLE_API_KEY"
  },
  "workflow": {
    "actionLabel": "Tomar muestra",
    "steps": [
      {
        "id": 1,
        "type": "Information",
        "text": "El mosquito Aedes Aegypti tiene estas caracter
    ísticas: \n 1) Es negro\n 2) Tiene rayas blancas en cuerpo
    y patas",
        "nextStepId": 2,
        "helpFileName": "mosquito_help.html"
      },
      {
        "id": 2,
        "type": "Photo",
        "text": "Tome una foto del mosquito desde arriba",
        "nextStepId": 3
      },
      {
        "id": 3,
        "type": "Photo",
        "text": "Tome una foto del mosquito desde un costado",
        "nextStepId": 4
      },
      {
        "id": 4,
        "type": "Location",
        "text": "Posicione la muestra en el mapa"
      }
    ]
  }
}
```

Código 5.1: Archivo de configuración para Samplers de la app ejemplo.

El código 5.2 muestra el archivo de configuración para Samplers. Si bien el detalle de como se configura este archivo se explica en la sección 5.3.1, en la línea 15 se puede observar como se define el Workflow y los «pasos» que lo conforman (Steps en Samplers) . El primer Step (línea 18) muestra un texto de información al científico ciudadano. Los Steps 2 y 3 (líneas 25 y 31) le solicitan tomar una foto desde arriba y desde un costado respectivamente, y el último Step (línea 37) le solicita la posición de la muestra.

Con este archivo de configuración, Samplers generará el código necesario para la app, que al compilarla y ejecutarla en un dispositivo móvil Android se verá como muestra la imagen a continuación (Figura 5.1).

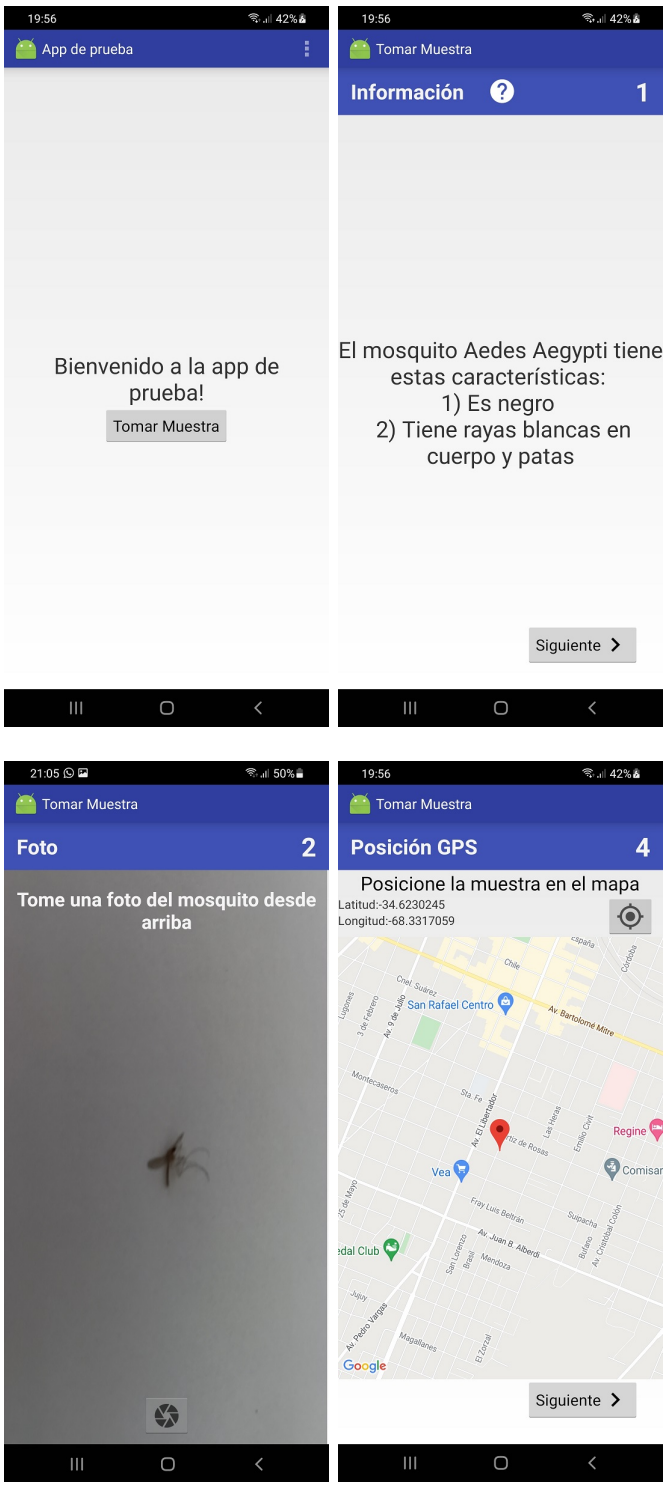

Figura 5.1: Capturas de pantalla de la app de ejemplo generada por Samplers

La app que genera Samplers tendrá una Activity principal que muestra el texto de bienvenida configurado ("Bienvenido a la app de prueba!") y un botón para tomar una muestra. Al presionar dicho botón se llama a la Activity encargada de tomar las muestras siguiendo el Workflow configurado. Esta Activity irá mostrando en pantalla un Fragment por cada Step dentro del Workflow. Cada uno de estos Fragments va interactuando con el usuario final de la app (el científico ciudadano) para completar cada uno de los Steps del Workflow y así tomar la muestra (Sample en Samplers).

Las opciones disponibles y cómo se configura el archivo de configuración de Samplers se explican en la sección 5.3.1.

De esta manera un científico que desee generar una app para su proyecto de ciencia ciudadana, solo tiene que completar el archivo de configuración, ejecutar Samplers y luego compilar en Android Studio y ejecutar la app en un dispositivo móvil o en los dispositivos virtuales que provee Android Studio para debugging.

Como Samplers genera el código fuente de la app, esto permite que un usuario con conocimientos de programación pueda modificarlo y personalizar la app a su gusto, así como también agregarle otras funcionalidades a la app generada. También permite que pueda ser usado en una app ya desarrollada o en desarrollo, como si fuera una librería que se incluye al proyecto y permite usar sus clases (esto se explica con más detalle en la sección 5.3.2).

Como Samplers se pensó como un proyecto dentro de Cientópolis[4] el mismo es open source bajo licencia GNU General Public License v3.0 y se usó un repositorio público en GitHub[34] para dejar el código fuente disponible para todo el mundo.

5.2. Estructura e implementación del framework

A continuación se explican las clases y componentes de Android que permiten que se cree una aplicación móvil con Samplers y su funcionamiento, y como se resolvieron cuestiones como el almacenamiento y envió de las muestras, la identificación de los usuarios que envían las muestras, etc.

5.2.1. Workflow, Step, StepFragment, StepResult, Sample

Estas clases conforman el corazón del framework, y son las necesarias para poder definir el protocolo de recolección de las muestras.

A continuación se muestra el diagrama de clases (Figura 5.2).

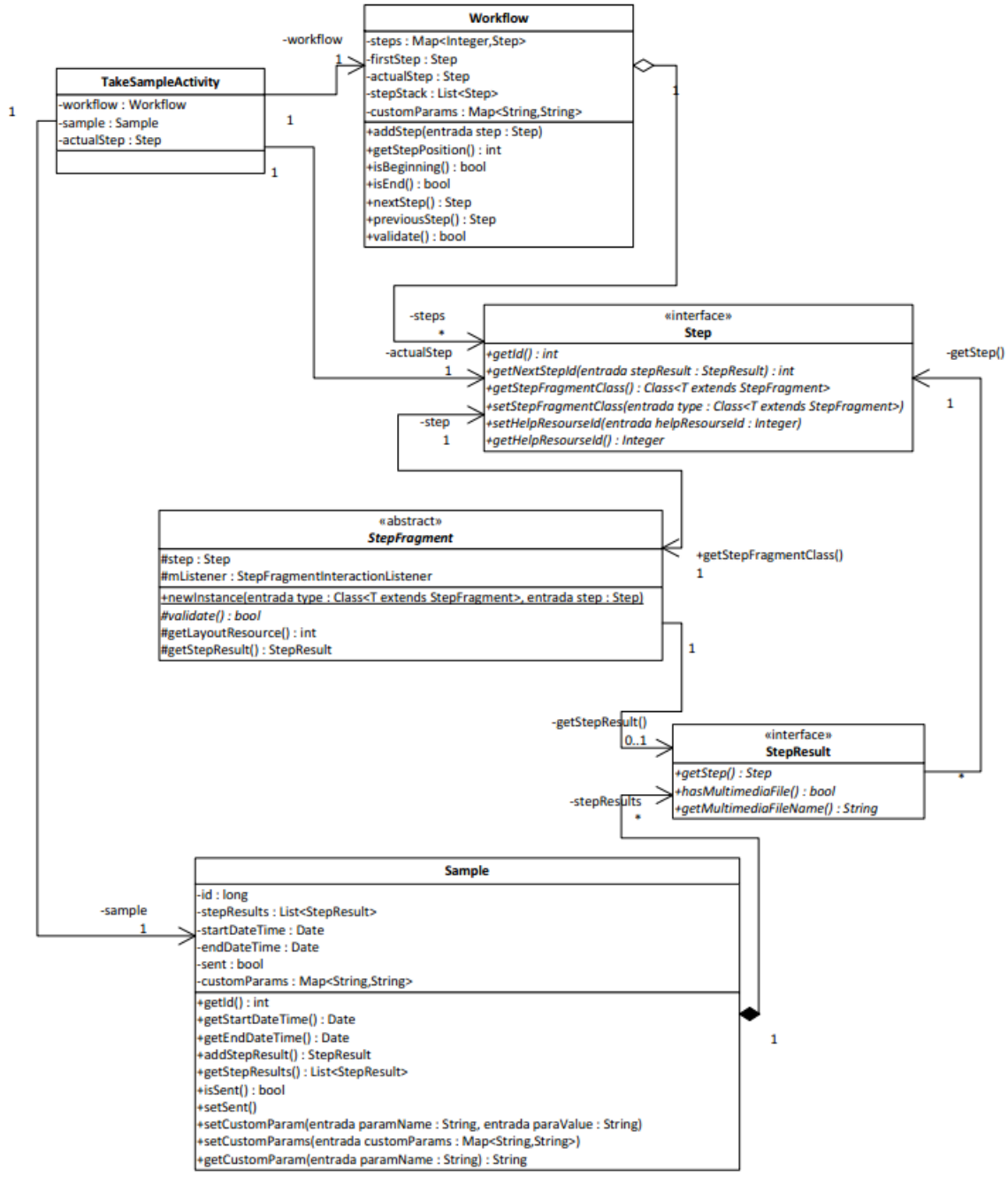

Figura 5.2: Diagrama de clases del core del framework

TakeSampleActivity es una Activity y es la encargada de tomar la muestra. Recibe como parámetro un Workflow, que representa el protocolo de recolección de la muestra. El Workflow esta formado por varias instancias de Step, que representan cada una un paso a seguir dentro del protocolo de recolección para obtener la muestra.

TakeSampleActivity ejecuta el Workflow y por cada Step obtiene y muestra en pantalla el StepFragment asociado, que es el Fragment encargado de mostrar la interfaz de usuario para ejecutar ese Step. El StepFragment a través de la interacción con el usuario final (el científico ciudadano) genera un StepResult, que es el resultado de la ejecución del Step asociado. Por cada tipo de Step, debe haber una clase StepFragment que sepa ejecutar ese Step y generar el StepResult asociado.

El conjunto de todos los StepResults obtenidos después de ejecutar todos los Steps del Workflow forman la muestra, que esta representada por la clase Sample.

5.2.1.1. Workflow: el protocolo de recolección de las muestras

La clase Workflow representa el protocolo de recolección de las muestras. Está formado por una colección de Steps, que representan los pasos a seguir para completar dicho protocolo, y un Step inicial que indica el inicio del mismo (el primer Step a ejecutar).

El Workflow es el encargado de llevar el estado del paso en el que se encuentra, y posee métodos para obtener el siguiente Step (*nextStep()* que depende del Step actual) y el Step anterior (*previuosStep()* para lo que maneja una lista a modo de pila -stack en inglés-).

El Workflow se comporta de manera secuencial, puede ir hacia adelante o volver hacia atrás, pero esto no impide que pueda tener bifurcaciones, y así formar diferentes caminos para tomar la muestra. Por ejemplo, se le podría preguntar al científico ciudadano si observa una característica particular al momento de tomar la muestra, y si la observa solicitarle que tome una foto de la misma, pero en caso contrario se puede omitir el paso de la foto.

5.2.1.2. Step: el paso

Un Step representa un paso dentro del Workflow. Tiene asociado un StepFragment que es el encargado de ejecutar el Step para generar un StepResult (el resultado). El Step básicamente tiene los parámetros o información para que pueda ser ejecutado por el StepFragment. Por ejemplo, para el caso en que el paso sea contestar una pregunta, el Step tendrá la pregunta en sí (un String) y una colección con las posibles respuestas (id y descripción).

El Step conoce cuál es el siguiente Step a ejecutar aunque a veces éste depende del StepResult generado, como es el caso de SelectOneStep en el que el siguiente paso depende de la opción seleccionada, y con esto se pueden crear bifurcaciones en el Workflow.

Samplers provee los siguientes Steps:

- InformationStep: Muestra una información (texto) al usuario.
- PhotoStep: Permite tomar una foto con la cámara del dispositivo móvil.
- SoundRecordStep: Permite grabar un sonido con el micrófono del dispositivo móvil.
- SelectOneStep: Muestra una pregunta con varias opciones como respuesta, de las cuales solo se puede seleccionar una sola.
- MultipleSelectStep: Muestra una pregunta con varias opciones como respuesta, de las cuales solo se pueden seleccionar varias opciones.
- LocationStep: Permite tomar la geo posición del dispositivo móvil.
- RouteStep: Permite grabar un recorrido usando el GPS del dispositivo móvil.
- InsertTextStep: Permite ingresar un texto.
- InsertDateStep: Permite seleccionar una fecha.
- InsertTimeStep: Permite seleccionar una hora.

El detalle de estos Steps y su funcionamiento se explican en la sección 6.6.

Si bien estos son los Steps que provee Samplers de manera predeterminada, un usuario programador puede definir y agregar sus propios Steps, junto con sus StepFragments y StepResults como se explica en la sección 6.6.10.

5.2.1.3. StepFragment: la vista y controlador del Step

El StepFragment es el encargado de ejecutar un Step y generar un StepResult con la interacción del usuario final (el científico ciudadano). Un StepFragment recibe un Step, que generalmente contiene los parámetros necesarios para ejecutarlo, y en base a eso muestra un Fragment para que, interactuando con el científico ciudadano, poder obtener un resultado (StepResult) para la muestra (Sample).

Por ejemplo, para el caso en que el paso sea contestar una pregunta, se mostrará la pregunta y se listarán las posibles respuestas en forma de radio-buttons, si solo se admite seleccionar una única respuesta, o en forma de check-buttons, si se permite seleccionar más de una.

Por cada tipo de Step, debe haber una clase StepFragment que sepa ejecutar ese Step y generar el StepResult asociado.

Es una clase abstracta, ya que su comportamiento depende de la implementación de cada tipo de StepFragment que se defina. Se definió así y no como una interfaz por la necesidad de que heredara de Fragment porque así lo necesita la Activity TakeSampleActivity, que es la encargada de ejecutar el Workflow para tomar la muestra.

Por cada Step que provee Samplers de manera predeterminada, también se provee un StepFragment que ejecuta cada Step. Asimismo, un usuario programador también puede desarrollar un StepFragment propio para alguno de los Steps provistos por Samplers, como se explica en la sección 6.6.10

5.2.1.4. StepResult: el resultado de la ejecución de un Step

El StepResult representa el resultado que se obtiene de ejecutar un Step. Contiene los datos obtenidos de ejecutar el Step asociado. Cada ejecución de un mismo Step, puede generar un StepResult diferente.

Por ejemplo, para el caso en que el Step sea contestar una pregunta, el StepResult contendrá la respuesta seleccionada, si solo se admite seleccionar una única respuesta, o una colección de respuestas, si se permite seleccionar más de una.

Se definió como una interfaz, ya que su comportamiento depende de la implementación de cada tipo de Step que se defina.

Un StepResult tiene asociado el Step en base al cual se generó. También puede tener asociado un archivo multimedia, como por ejemplo en los casos de PhotoStep y SoundRecordStep que guardan una foto y un archivo de sonido respectivamente.

5.2.1.5. Sample: la Muestra

La clase Sample representa una muestra tomada a partir de seguir los pasos (Steps) del protocolo del recolección (Workflow). Contiene los resultados (StepResult) de la ejecución de cada paso. Cada ejecución del Workflow puede generar una colección de StepResults diferente.

También guarda fecha y hora de inicio y finalización. Esto es útil para sacar una estadística de cuanto tarda un científico ciudadano en recolectar una muestra y así poder analizar optimizaciones para la aplicación final.

Una vez recolectadas, las muestras se guardan en el dispositivo móvil y son enviadas a través de Internet a un servidor web previamente configurado.

5.2.2. TakeSampleActivity

La clase TakeSampleActivity es la encargada de ejecutar el Worflow para tomar la muestra. Es una Activity que recibe un objeto Workflow como parámetro y va iterando sobre los Steps del mismo. A cada Step le pide su StepFragment y lo muestra en pantalla para que, interactuando con el científico ciudadano, se genere el StepResult para la muestra (Sample). Una vez finalizado el Workflow, guarda la muestra, controla si se puede enviar la misma y finaliza.

5.2.3. Persistencia local

Las muestras se guardan en el dispositivo móvil en un archivo JSON, junto con los archivos multimedia que pudiera tener, dentro de un directorio por cada muestra. Las mismas se guardan dentro del directorio «samples» en el almacenamiento interno del dispositivo.

Se eligió el almacenamiento interno para guardar las muestras porque no se necesitan permisos especiales para acceder al mismo y, de forma predeterminada, los archivos que se guardan en el mismo son privados para la aplicación y otras aplicaciones no pueden tener acceso a ellos (tampoco el usuario). Cuando el usuario desinstala la aplicación, estos archivos se quitan[35].

Por ejemplo, una muestra con id 123456 y 2 fotos se guarda de la siguiente manera:

```
/samples/                // Directorio de muestras
  sample_123456/         // Directorio de la muestra con id:123456
    sample_123456.json   // Objeto Sample en formato JSON
    1524437599776.jpg    // Archivo de foto 1
    1524441664170.jpg    // Archivo de foto 2
```

Figura 5.3: Ejemplo de cómo se guarda una muestra.

Para pasar la muestra a un archivo JSON se usó Gson[36], una librería de Google distribuida bajo licencia Apache 2.0 que convierte objetos Java a JSON y viceversa de manera muy sencilla, con métodos *toJson()* y *fromJson()* para convertir hacia y desde JSON respectivamente.

5.2.4. Envío de muestras a servidor web

Una vez guardadas localmente, las muestras se envían al servidor web previamente configurado, mediante un mensaje HTTP POST. Para ello se usó la librería OkHttp[37], distribuida por Square Inc. bajo licencia Apache 2.0, que resuelve de manera sencilla y eficiente el envío de mensajes HTTP en Android.

Por cada muestra se envía el objeto Sample en formato JSON junto con los archivos multimedia que pudiera tener, todo comprimido en un solo archivo ZIP. Para comprimir las mismas se usaron las librerías estándares de Java (java.util.zip) que proporciona clases para leer y escribir archivos ZIP y GZIP estándares.

Las muestras se envían automáticamente cuando se detecta conexión wifi o a petición del usuario usando cualquier red disponible.

En el caso de envío automático, luego de guardar la muestra se controla si el dispositivo está conectado a una red wifi y se intenta enviar la muestra; caso contrario queda pendiente de envío y se intenta enviar cuando se detecta conexión wifi. Para detectar la conexión a wifi se usa un BroadcastReceiver que atiende el evento de difusión que envía Android cuando el dispositivo móvil se conecta a wifi, aún cuando la app no se esté ejecutando, lo que permite el envío de la muestra con un Service en segundo plano.

Para el caso que es a petición del usuario, se listan las muestras tomadas en la Activity SamplesListActivity con un botón que permite el envío de las

mismas usando cualquier tipo de conexión a Internet.

5.2.5. Identificación

Samplers provee un sistema de identificación (login), el cual se puede habilitar o no de acuerdo a las necesidades del proyecto. Si se habilita el mismo puede ser opcional, permitiendo al científico ciudadano tomar y enviar muestras habiéndose o no identificado, o puede ser obligatorio, requiriendo que el científico ciudadano se identifique para poder tomar y enviar las muestras.

De manera predeterminada Samplers incluye identificación con la API de Google, es decir que el científico ciudadano puede usar su cuenta de Google que tiene configurada en el dispositivo móvil Android para identificarse en la aplicación.

Además, un usuario programador del framework puede definir su propio método de identificación, ya sea con usuario y contraseña o usando alguna otra API de redes sociales por ejemplo, como son Facebook, Instagram, Twitter, etc. como se explica en la sección 6.5.3.

Cuando se envía la muestra también se incluyen dos parámetros que son el id del usuario identificado y el método usado para identificar (que puede ser google o uno personalizado). Se envían de esta forma porque al momento de tomar la muestra el científico ciudadano puede no estar identificado, e identificarse antes de enviarlas.

5.2.6. Otras Activities de la app generada por Samplers

Como se mencionó antes, Samplers genera una app lista para ejecutar en un dispositivo móvil con Android, por lo que se incluyen otras Activities para completar la app generada. A continuación se describen dichas Activities

5.2.6.1. SamplersMainActivity

SamplersMainActivity es la Activity principal de la app generada, es la que se inicia cuando el usuario hace clic sobre el icono de la app. Esta Activity contiene la pantalla principal de la app, en la que se muestra el mensaje de bienvenida configurado, un menú para acceder a las Activities que se explican mas abajo, y un botón para tomar la muestra que cuando se presiona se llama a TakeSampleActivity para tomar la muestra. También, si se configuró el uso de identificación, se muestra un botón para que el usuario final pueda iniciar sesión.

Figura 5.4: Ejemplo de la Activity principal.

5.2.6.2. SamplesListActivity

SamplesListActivity es una Activity que visualiza un listado de las muestras tomadas por el usuario final de la app, y que se encuentran almacenadas en el dispositivo móvil, con opciones para eliminarlas o enviarlas al servidor en caso de que no hayan sido enviadas automáticamente.

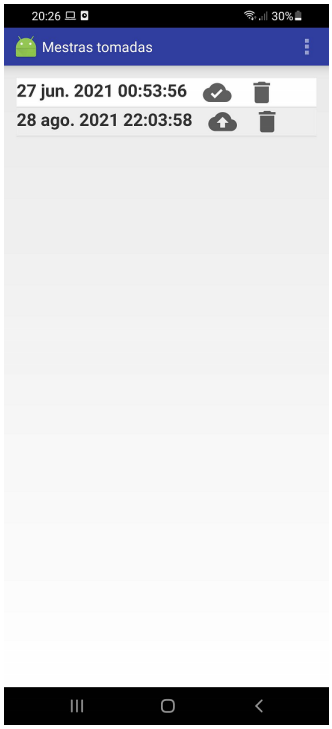

Figura 5.5: Ejemplo de como se visualiza SamplesListActivity.

5.2.6.3. HelpActivity

HelpActivity es una Activity que se usa para mostrar ayuda al usuario final de la app. Para mostrar ayuda se debe configurar la misma en formato HTML como se explica en la sección 6.4. HelpActivity es la Activity que muestra tanto la ayuda general (que se accede desde el menú de Samplers-MainActivity) como la ayuda que se muestra en cada StepFragment (si se configura) al momento de tomar la muestra.

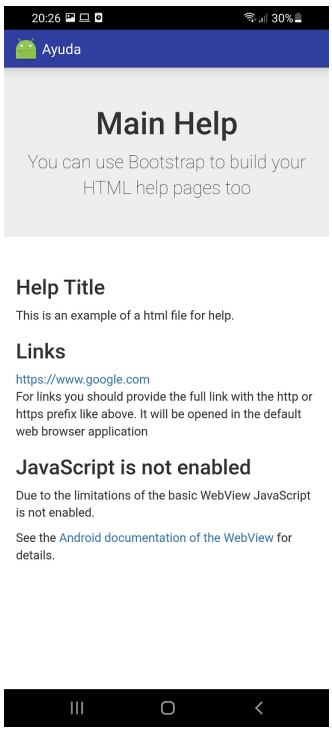

Figura 5.6: Ejemplo de como se visualiza SamplesListActivity.

5.3. Configuración del framework

Como se mencionó antes, para generar aplicaciones móviles con Samplers se debe configurar el Workflow y unos parámetros más, lo que se hace con el archivo de configuración llamado SamplersConfig.json. También se puede hacer mediante código, instanciando las clases como si fuera una librería, por usuario con conocimientos de programación. A continuación se explican ambos métodos.

5.3.1. Archivo de configuración

El archivo de configuración es un archivo en formato JSON que se debe completar con los parámetros necesarios para crear la app con Samplers. Se eligió el formato JSON porque nos pareció más sencillo de manipularlo por una persona (que XML por ejemplo, que también fue evaluado), suponiendo que el científico tuviese que editarlo manualmente.

El archivo está compuesto por tres objetos: project, application y workflow.

En el objeto project, se encuentran las configuraciones del proyecto de Android Studio en el cual se creará la app, como por ejemplo la ruta en donde se crearán los archivos de las Activities y el nombre del package que se usará para las mismas.

En el objeto application, se encuentran los parámetros para configurar la app, como por ejemplo el título de la app, el mensaje de bienvenida o el servidor web al que se deben enviar las muestras recolectadas.

El objeto workflow se usa justamente para configurar el Workflow (el protocolo de recolección) que usará la app para tomar las muestras. En el mismo se deben especificar los Steps que se usarán con sus respectivos parámetros cada uno. Por defecto Samplers provee los siguientes Steps:

- Information: Muestra un texto al científico ciudadano.
- Photo: Solicita tomar una foto.
- Sound: Solicita grabar sonido.
- Location: Solicita tomar la posición del GPS del dispositivo móvil.
- Route: Solicita grabar con el GPS del dispositivo móvil un recorrido (un conjunto de posiciones GPS).
- SelectOne: Solicita seleccionar una única respuesta a una pregunta.
- MultipleSelect: Solicita seleccionar una o varias respuestas a una pregunta.
- InsertText: Solicita ingresar texto.
- InsertDate: Solicita seleccionar una fecha.
- InsertTime: Solicita seleccionar una hora.

En la siguiente imagen se muestra un ejemplo del archivo de configuración para Samplers.

```
{
  "project": {
    "app_path": "app/src/main/java/com/example/myApplication/",
    "package_name": "com.example.myApplication"
  },
  "application": {
    "title": "App de prueba",
    "welcomeMessage": "Bienvenido a la app de prueba!",
    "networkConfiguration": {
      "url": "https://192.168.0.10/samplers/upload.php",
      "paramName": "sample"
    },
    "googleMaps_API_KEY": "YOUR_GOOGLE_API_KEY",
    "mainHelpFileName": "mainhelp.html"
  },
  "workflow": {
    "actionLabel": "Tomar muestra",
    "steps": [
      {
        "id": 1,
        "type": "Information",
        "text": "Por favor siga las instrucciones",
        "nextStepId": 2
      },
      {
        "id": 2,
        "type": "Location",
        "text": "Posicione la muestra en el mapa",
        "nextStepId": 3
      },
      {
        "id": 3,
        "type": "Photo",
        "text": "Tome una foto del lugar",
        "nextStepId": 4,
        "helpFileName": "photohelp.html"
      },
      {
        "id": 4,
        "type": "MultipleSelect",
        "title": "Seleccione las cosas que ve",
        "options": [
           {
            "id": 1,
            "text": "Arboles"
          },
```

```
            {
              "id": 2,
              "text": "Basura"
            },
            {
              "id": 3,
              "text": "Agua"
            }]
        }
    ]
  }
}
```

Código 5.2: Ejemplo de un archivo de configuración para Samplers.

Una vez configurado el archivo, Samplers generará una Activity principal para la app que tendrá todas las configuraciones del objeto application y que usará el Workflow configurado para llamar a la Activity encargada de tomar las muestras (TakeSampleActivity) cuando el científico ciudadano presione el botón para tomar una muestra.

El detalle de todos los parámetros, como configurar el archivo de configuración y los diferentes Steps disponibles para armar el Workflow se explican en la sección 6.3

5.3.2. Usando las clases

Como se mencionó antes, Samplers también puede ser usado por una persona con conocimientos de programación en una app que se encuentre en desarrollo, como si fuera una librería que se incluye al proyecto y permite usar sus clases.

Básicamente se tiene que instanciar la clase Workflow y agregarle instancias de los diferentes Steps que se quieran usar. Luego se debe iniciar la Activity TakeSampleActivity pasándole como parámetro la instancia de Workflow creada. Ésto se puede hacer por ejemplo en el método «onClick» de un botón; de esta forma, cuando se presione dicho botón, Samplers se encargará te tomar la muestra, guardarla en el dispositivo móvil y enviarla al servidor cuando haya conexión wifi.

```
public void takeSampleClick(View view) {
  Intent intent = new Intent(this, TakeSampleActivity.class);

  Workflow workflow = getWorkflow();
  intent.putExtra(TakeSampleActivity.EXTRA_WORKFLOW, workflow);

  startActivity(intent);
}
```

Código 5.3: Ejemplo de como iniciar la activity TakeSampleActivity instanciando las clases.

El detalle de como usar las clases y establecer las configuraciones adicionales se explican en la sección 6.1.

Capítulo 6

Instanciación y uso del framework

En esta sección se explica como instanciar el framework para su uso, brindando algunos ejemplos concretos.

Una vez instalado el framework (como se explica en el anexo A) el siguiente paso es instanciarlo para usarlo. La instanciación puede ser manual o usando el generador de clases de Gradle.

6.1. Instanciación manual

Una vez agregada la librería de Samplers al proyecto, se debe crear un objeto Workflow, agregarle los objetos Steps, y llamar a la activity TakeSampleActivity pasándole el objeto Workflow como parámetro. También es necesario establecer la configuración general en el método onCreate de la activity principal (main activity) para indicar a donde se enviarán las muestras tomadas con la app. Se puede usar una activity principal propia o se puede heredar de SamplersMainActivity. En ambos casos se debe hacer lo siguiente:

- Establecer la configuración general en el método onCreate de la activity principal:

```java
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);

  // Establecer la configuración de red
  NetworkConfiguration.setURL("http://192.168.1.10/samplers/
    upload.php");
  NetworkConfiguration.setPARAM_NAME_SAMPLE("sample");

  // Opcional, si se desea usar identificación (*)
  NetworkConfiguration.setPARAM_NAME_USER_ID("user_id");
  NetworkConfiguration.setPARAM_NAME_AUTHENTICATION_TYPE("
    authentication_type");

  AuthenticationManager.setAuthenticationEnabled(true);
  AuthenticationManager.setAuthenticationOptional(true);
}
```

Código 6.1: Ejemplo de configuración general

(*) Ver la sección 6.5 para más detalles sobre como usar identificación.

- Crear un Workflow. Si se está heredando de SamplersMainActivity se debe hacer sobreescribiendo el método getWorkflow.

```java
@Override
protected Workflow getWorkflow() {
  Workflow workflow = new Workflow();

  Step step = new PhotoStep(2,"Por favor tome una foto de su
    gato", null);
  workflow.add(step);

  step = new InformationStep(1,"Bienvenido a la app de prueba",
     2);
  workflow.add(step);

  return workflow;
}
```

Código 6.2: Ejemplo de un Workflow que tiene dos Steps. El primero muestra un mensaje de bienvenida y el segundo pide para tomar una foto.

Nota: Para ver los distintos Steps que se pueden usar vea la sección 6.6.

- Iniciar la activity TakeSampleActivity. Si se está heredando de SamplersMainActivity esto se hace automáticamente en el método onClick

del botón "tomar muestra". De lo contrario, deberá iniciarla de la siguiente forma, en el método onClick de un botón por ejemplo:

```
public void takeSampleClick(View view) {
  Workflow workflow = getWorkflow();

  Intent intent = new Intent(this, TakeSampleActivity.class);
  intent.putExtra(TakeSampleActivity.EXTRA_WORKFLOW, workflow);
  startActivity(intent);
}
```

Código 6.3: Ejemplo de como iniciar la activity TakeSampleActivity.

6.2. Instanciación usando el generador de clases de Gradle

Básicamente, el generador de clases de Gradle se encarga de hacer una instanciación manual a partir de un archivo de configuración (JSON). Está pensado para desarrolladores que no tienen muchos conocimientos en Android, o para servir de interfaz entre una aplicación que genere apps a través de Samplers.

Los pasos para usar el generador de clases de Gradle son:

1. Crear un archivo JSON con el nombre SamplersConfig.json
   - El formato y las opciones están explicadas sección 6.3.
   - Para validar sintaxis se puede usar por ejemplo el validador online https://jsonformatter.curiousconcept.com que es gratuito.
   - Al final de esta sección se provee un archivo de ejemplo.
2. Copiar el archivo creado en el item anterior al directorio raíz del proyecto Android creado.
3. Descargar la última versión de los archivos **samplers.gradle** y **samplersclassgenerator.jar** del repositorio de Samplers[1] y copiarlos también al directorio raíz del proyecto Android.
4. Enlazar el archivo samplers.gradle en el archivo **build.gradle** de la aplicación:

[1] https://github.com/cientopolis/samplers/releases/

- Android Studio crea por defecto dos archivos build.gradle, uno a nivel de aplicación y otro a nivel de proyecto. Debe usarse el de aplicación.
- Al final del archivo build.gradle de aplicación agregar:

```
apply from: '../samplers.gradle'
```

- Al guardar los cambios, Android Studio sugerirá hacer una sincronización del proyecto, hacerla. Esto generará en la aplicación una activity llamada MyMainSamplersActivity en base a las opciones configuradas en el archivo SamplersConfig.json.
- Si se necesita volver a generar esta activity (si se quieren modificar algunas opciones por ejemplo) se puede eliminar la misma, hacer las modificaciones en el archivo SamplersConfig.json y volver a generar el proyecto (en el menu Build -> Make Project)

5. Eliminar o personalizar el archivo **style.xml** que está en **res/values** en la aplicación

6. Ejecutar la aplicación y listo.

6.3. Secciones del archivo

El archivo SamplersConfig.json es un archivo JSON con 3 objetos: project, application y workflow. A continuación se explican en detalle cada uno de ellos.

6.3.1. El objeto project

El objeto project tiene dos campos

- **app_path**: Un String con la ubicación del directorio de los fuentes de la aplicación, relativo al directorio del proyecto. Es donde están los archivos -java de la aplicación y donde se creará el archivo MyMainSamplersActivity.java

- **package_name**: Un String con el nombre del package usado para las activities de la aplicación. Es el package donde la activity MyMainSamplersActivity será agregada.

```
"project" : {
  "app_path" = "app/src/main/java/com/example/myApplication/"
  "package_name" : "com.example.myApplication"
}
```

Código 6.4: Ejemplo del objeto project.

6.3.2. El objeto application

El objeto application tiene 7 campos, de los cuales 3 son requeridos y los otros 4 opcionales (para habilitar características especiales)

- **title**: Un String con el nombre de la aplicación.
- **welcomeMessage**: Un String con el mensaje de bienvenida que se mostrará en la activity principal (MyMainSamplersActivity)
- **networkConfiguration**: Un objeto con la configuración de red que se usará para enviar las muestras al servidor web. Ver mas abajo la configuración de este objeto.
- **googleMaps_API_KEY**: [Opcional] Un String con la API Key de Google. Este campo es necesario si se van a usar los servicios de ubicación y mapas (Location Step y Route Step). La API Key de google se puede obtener desde la página de google developers [2].
- **mainHelpFileName**: [Opcional] Un String con el nombre del archivo HTML que contiene la ayuda principal de la aplicación. Este archivo debe estar junto con el archivo SamplersConfig.json. Ver la sección 6.4 para mas detalles.
- **authenticationEnabled**: [Opcional] Un boolean que indica si se usará autenticación (true) o no (false). Si se omite este campo se asume false. Ver la sección Usando Autenticación para mas detalles.
- **authenticationOptional**: [Opcional] Un boolean que indica si la autenticación será opcional (true) o requerida (false). Si se omite este campo se asume true (autenticación opcional). Este campo solo tiene sentido si se usa autenticación. Ver la sección Usando Autenticación para mas detalles.

[2] https://developers.google.com/maps/documentation/android-api/signup

El objeto **networkConfiguration**: El objeto networkConfiguration contiene la configuración de red que se usará para enviar las muestras al servidor web. Tiene 4 campos, de los cuales 2 son requeridos y los otros 2 opcionales.

- **url**: Un String con la URL del servidor web al cual se le enviaran las muestra con un mensaje HTTP POST.
- **paramName**: Un String con el nombre del parámetro dentro del mensaje HTTP POST en el que se enviará la muestra.
- **paramNameUserId**: [Opcional] Un String con el nombre del parámetro dentro del mensaje HTTP POST en el que se enviará el id del usuario que envía la muestra. Este campo solo es necesario si se usa autenticación. Ver la sección Usando Autenticación para mas detalles.
- **paramNameAuthenticationType**: [Opcional] Un String con el nombre del parámetro dentro del mensaje HTTP POST en el que se enviará el tipo de autenticación que usó el usuario que envía la muestra. Este campo solo es necesario si se usa autenticación. Ver la sección Usando Autenticación para mas detalles.

```
"application": {
  "title" : "App de prueba",
  "welcomeMessage" : "Bienvenido a la app de prueba!",
  "networkConfiguration" : {
    "url" : "http://192.168.1.10/samplers/upload.php",
    "paramName" : "sample",
    "paramNameUserId" : "user_id",
    "paramNameAuthenticationType" : "authentication_type"
  },
  "authenticationEnabled" : true,
  "authenticationOptional" : true,
  "googleMaps_API_KEY" : "your_google_maps_API_KEY",
  "mainHelpFileName" : "mainhelp.html"
}
```

Código 6.5: Ejemplo del objeto application.

6.3.3. El objeto workflow

El objeto workflow representa el protocolo para la toma de la muestra. Son los pasos que se ejecutarán para tomar la misma. El objeto cuenta con dos campos:

- **actionLabel**: Un String con el título que se usará para el botón que inicia la activity TakeSampleActivity, que es la encargada de tomar la muestra.
- **steps**: Un Array de Objetos Step los cuales forman el workflow. El primer objeto del array se considera como el inicio del mismo. Ver la sección Steps para mas detalles.

```
"workflow": {
  "actionLabel" : "Tomar muestra",
  "steps": [
    {
      "id" : 1,
      "type" : "Information",
      "text" : "Por favor, siga las instrucciones",
      "nextStepId" : 2
    },
    {
      "id" : 2,
      "type" : "Location",
      "text" : "Posicione la muestra en el mapa",
      "nextStepId" : 3
    },
    {
      "id" : 3,
      "type" : "Photo",
      "text" : "Tome una foto del lugar",
      "nextStepId" : 4,
      "helpFileName" : "photohelp.html"
    },
    {
      "id" : 4,
      "type": "MultipleSelect",
      "title" : "Seleecione las cosas que ve",
      "options" : [
        {
          "id":1,
          "text":"Arboles"
        },
        {
          "id":2,
          "text":"Basura"
        },
        {
          "id":3,
          "text":"Agua"
        },
```

```
40          {
41            "id":4,
42            "text":"Animales"
43          }
44        ]
45      }
46    ]
47  }
```

Código 6.6: Ejemplo del objeto workflow.

6.4. Mostrar Ayuda

Samplers permite mostrar ayuda para el usuario final de la app a través de archivos HTML. Estos archivos deben estar ubicados junto con el archivo SamplersConfig.json

Samplers permite dos formas de mostrar ayuda:

- **Ayuda general**: que se accede desde el menú de la Activity principal cuando se configura un archivo de ayuda para la app.
- **Ayuda puntual para cada Step**: que se accede desde un botón en los StepFragments que aparece cuando el Step tiene configurado un archivo de ayuda.

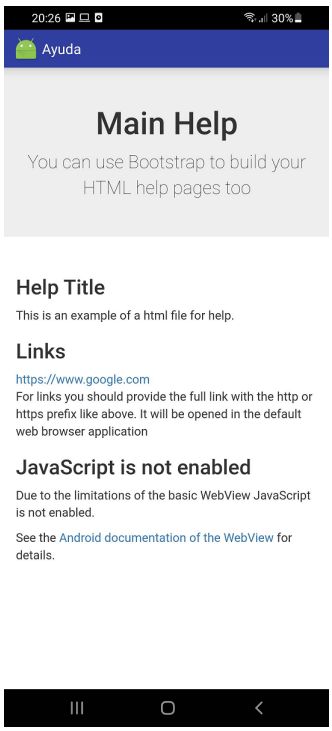

Figura 6.1: Ejemplo de como se muestra la ayuda.

JavaScript no esta habilitado: Debido a las limitaciones del componente WebView de Android no esta habilitado el uso de JavaScript. Para más información ver la documentación oficial de Android sobre WebView.[3]

6.4.1. Ayuda general

Para configurar la ayuda general se debe completar el atributo *mainHelpFileName* del objeto *application* con el nombre del archivo HTML que contiene la ayuda que se quiere mostrar. El archivo HTML debe estar ubicado junto con el archivo SamplersConfig.json.

[3]https://developer.android.com/reference/android/webkit/WebView

```
"application": {
  "title" : "App de prueba",
  "welcomeMessage" : "Bienvenido a la app de prueba!",
  "networkConfiguration" : {
    "url" : "http://192.168.1.10/samplers/upload.php",
    "paramName" : "sample",
    "paramNameUserId" : "user_id",
    "paramNameAuthenticationType" : "authentication_type"
  },
  "authenticationEnabled" : true,
  "authenticationOptional" : true,
  "googleMaps_API_KEY" : "your_google_maps_API_KEY",
  "mainHelpFileName" : "mainhelp.html"
}
```

Código 6.7: Ejemplo de configuración de ayuda general (línea 13).

6.4.2. Ayuda puntual para cada Step

Para configurar la ayuda para un Step se debe completar el atributo *helpFileName* del objeto *step* con el nombre del archivo HTML que contiene la ayuda que se quiere mostrar. El archivo HTML debe estar ubicado junto con el archivo SamplersConfig.json.

```
"workflow": {
  "actionLabel" : "Tomar muestra",
  "steps": [
    {
      "id" : 1,
      "type" : "Photo",
      "text" : "Tome una foto del lugar",
      "nextStepId" : 2,
      "helpFileName" : "photohelp.html"
    },
...
```

Código 6.8: Ejemplo de configuración de ayuda en un Step (línea 9).

6.5. Usando identificación

Por defecto Samplers provee identificación con Google, porque en Android es necesario tener una cuenta de Google para usar el dispositivo móvil, aunque también está preparado para poder integrar identificación con otras plataformas/APIs.

Para usar la identificación con Google, es necesario registrar la aplicación en la pagina de desarrolladores de google[4]. Ahí hay que seguir los pasos para configurar la API en el proyecto dentro de la consola de Google. Es necesario proveer el nombre de la aplicación, package name, y también el SHA-1 hash del certificado con el que se firma la aplicación.

Una vez registrada la aplicación en Google, hay que configurar Samplers para habilitar la identificación como se explican en las secciones 6.5.1 o 6.5.2.

Una vez configurado los valores de los parámetros de identificación, Samplers mostrará un fragment de inicio de sesión la primera vez que el usuario intente tomar una muestra. Si la identificación es opcional, se mostrará un botón para omitir el inicio de sesión y continuar con la toma de la muestra. También se muestra un botón para iniciar sesión en la activity principal (si se está usando la que provee Samplers).

Cuando la muestra es enviada, el id de usuario y el método de identificación (por defecto 'google') se envían junto con esta.

6.5.1. Configurar identificación con el generador de clases de Gradle

Para usar identificación usando el generador de clases de Gradle, es necesario configurar los siguientes parámetros en el objeto **applicaction**:

- **authenticationEnabled**: poner en true para habilitar la identificación
- **authenticationOptional**: poner en true si se desea que la identificación sea opcional, o en false si se desea que la identificación sea obligatoria.
- Dentro del parámetro **networkConfiguration** es necesario establecer los parámetros **paramNameUserId** y **paramNameAuthenticationType** con los nombres de los parámetros con los que irán el id de usuario y el tipo de identificación usada respectivamente dentro del mensaje HTTP POST.

[4]https://developers.google.com/identity/sign-in/android/start-integrating

```
"application": {
  "title" : "App de prueba",
  "welcomeMessage" : "Bienvenido a la app de prueba!",
  "networkConfiguration" : {
    "url" : "http://192.168.1.10/samplers/upload.php",
    "paramName" : "sample",
    "paramNameUserId" : "user_id",
    "paramNameAuthenticationType" : "authentication_type"
  },
  "authenticationEnabled" : true,
  "authenticationOptional" : true,
  "googleMaps_API_KEY" : "your_google_maps_API_KEY",
  "mainHelpFileName" : "mainhelp.html"
}
```

Código 6.9: Ejemplo del objeto application configurado para usar identificación (líneas 7 8 10 y 11).

6.5.2. Configurar identificación manualmente

Para usar identificación cuando se usan las clases directamente, es necesario definir la configuración de red y de identificación en el método **onCreate()** de la activity principal:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);

  // Establecer la configuración de red
  NetworkConfiguration.setURL("http://192.168.1.10/samplers/
    upload.php");
  NetworkConfiguration.setPARAM_NAME_SAMPLE("sample");

  // Establecer la configuración para usar Identificación
  NetworkConfiguration.setPARAM_NAME_USER_ID("user_id");
  NetworkConfiguration.setPARAM_NAME_AUTHENTICATION_TYPE("
    authentication_type");
  AuthenticationManager.setAuthenticationEnabled(true);
  AuthenticationManager.setAuthenticationOptional(true);
}
```

Código 6.10: Ejemplo de configuración de identificación en el método onCreate.

6.5.3. Usando un método de identificación propio

Con Samplers también se puede usar un método de identificación propio, definiendo un LoginFragment y una clase User (o varias clases si se desea proporcionar identificación con diferentes APIs, como Facebook, Instagram, Tweeter, etc.) y Samplers enviará junto con la muestra el id de usuario y el método de identificación usado.

6.5.3.1. Definiendo un Login Fragment propio

Es necesario crear un fragment que herede de LoginFragment, y configurar la clase AuthenticationManager para que lo use, llamando al método setLoginFragmentClass() dentro del método onCreate() de la ativity principal:

```
1  @Override
2  protected void onCreate(Bundle savedInstanceState) {
3    super.onCreate(savedInstanceState);
4
5    // Configurar AuthenticationManager para que use un
       LoginFragment propio
6    AuthenticationManager.setLoginFragmentClass(
       MyCustomLoginFragment.class);
7
8    // resto de las configuraciones
9    // ...
10 }
```

Código 6.11: Ejemplo de como configurar un LoginFragment propio (línea 6).

El proceso de login y la interacción con las APIs responsabilidad del desarrollador, pero después de que el usuario inicia sesión en la API seleccionada, es necesario llamar al método login() en la clase AuthenticationManager y al método onLogin() en el objeto mListener heredado:

```
1  if (loginOK) {
2    AuthenticationManager.login(user, getActivity().
       getApplicationContext());
3    mListener.onLogin(user);
4  }
```

Código 6.12: Ejemplo de como llamar al método onLogin.

6.5.3.2. Definiendo una clase User propia

Es necesario crear una clase usuario propia que implemente la interfaz User por cada método de identificación que se use. Los objetos de dichas clases se usarán para llamar al método login() de la clase AuthenticationManager.

```
public class EMailUser implements User {

    public static final String AUTHENTICATION_TYPE = "email";

    private String userName;
    private String email;

    public EMailUser(String userName, String email) {
        this.userName = userName;
        this.email = email;
    }

    @Override
    public String getAuthenticationType() {
        return AUTHENTICATION_TYPE;
    }

    @Override
    public String getUserName() {
        return userName;
    }

    @Override
    public String getUserId() {
        return email;
    }

}
```

Código 6.13: Ejemplo de una clase usuario propia.

6.6. Los diferentes Steps y sus resultados (StepResult)

De manera predeterminada, Samplers provee los Steps que se explican a continuación. Además de los provistos por Samplers, un usuario programador puede definir y agregar sus propios Steps, junto con sus StepFragments y StepResults como se explica en la sección 6.6.10

Todos los Step tienen los siguientes campos:

- **id:** El identificador del Step, que debe ser único por instancia.
- **type:** El tipo de Step, que debe ser uno de los que se detallan a continuación.
- **nextStepId:** [Opcional] El identificador del siguiente Step que se debe ejecutar al finalizar con el actual. Si no se especifica, se asume que el Step es el final del Workflow.
- **helpFileName:** [Opcional] El nombre del archivo HTML con la ayuda del Step (como se explicó en la sección 6.4. Si no se especifica, no se mostrará el botón de ayuda para el Step.

Además de estos campos, los diferentes tipos de Step agregan campos adicionales como se detalla en la explicación de cada Step a continuación.

6.6.1. PhotoStep: Tomar una foto

PhotoStep se usa para solicitarle al científico ciudadano que tome una foto. Tiene un texto (text) que se muestran a modo de instrucciones o mensaje cuando la cámara esta encendida. Luego de tomada la foto, se muestra una vista preliminar de la misma, con un botón que da la opción de volver a tomar la foto.

```
{
  "id" : 1,
  "type" : "Photo",
  "text" : "Tome una foto de su gato",
  "nextStepId" : 2
}
```

Código 6.14: PhotoStep usando el generador de clases.

```
Step step = new PhotoStep(1,"Tome una foto de su gato",2);
workflow.addStep(step);
```

Código 6.15: PhotoStep en Java.

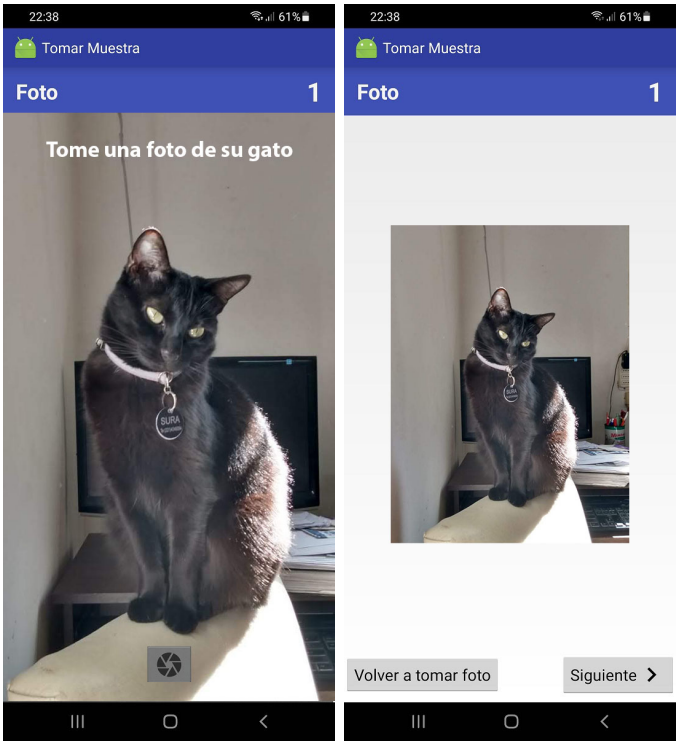

Figura 6.2: Ejemplo de un PhotoStep en ejecución. A la izquierda al momento de tomar la foto, y a la derecha al momento de mostrar la vista preliminar.

6.6.1.1. PhotoStepResult: El resultado de Tomar una foto

El PhotoStepResult contiene el nombre del archivo de la foto tomada. La foto va como archivo jpg en la carpeta de la muestra. Puede haber varias fotos si hay varios PhotoSteps en el Workflow.

6.6.2. SoundRecordStep: Grabar sonido

SoundRecordStep se usa para solicitarle al científico ciudadano que grabe un sonido. Tiene un texto (text) que se muestran a modo de instrucciones

o mensaje. Luego de grabado el sonido, el mismo se puede escuchar, y en todo caso volver a grabar otro sonido.

```
{
  "id" : 1,
  "type" : "Sound",
  "text" : "Grabe el sonido de su auto",
  "nextStepId" : 2
}
```

Código 6.16: SoundRecordStep usando el generador de clases.

```
Step step = new SoundRecordStep(1,"Grabe el sonido de su auto"
    ,2);
workflow.addStep(step);
```

Código 6.17: SoundRecordStep en Java.

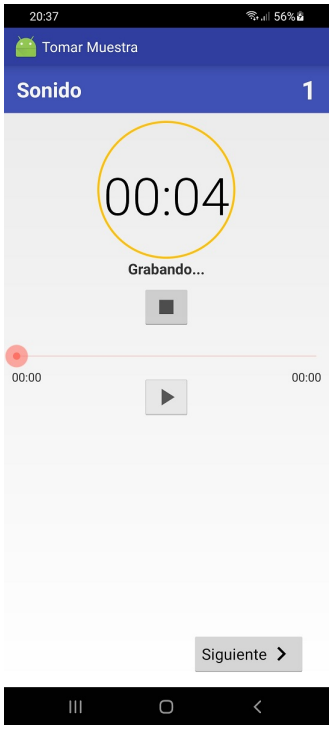

Figura 6.3: SoundRecordStep en ejecución.

6.6.2.1. SoundRecordStepResult: El resultado de Grabar sonido

El SoundRecordStepResult contiene el nombre del archivo del sonido grabado. El sonido va como archivo mp4 en la carpeta de la muestra. Puede haber varios sonidos si hay varios SoundRecordSteps en el Workflow.

6.6.3. InformationStep: Mostrar información

InformationStep se usa para mostrar una información (texto) al científico ciudadano, por ejemplo, para mostrar un mensaje de bienvenida o para explicar brevemente los pasos que seguirán a continuación para tomar la muestra.

```
{
  "id":1,
  "type" : "Information",
  "text" : "Por favor siga las instrucciones para tomar la
    muestra",
  "nextStepId": 2
}
```

Código 6.18: InformationStep usando el generador de clases.

```
Step step = new InformationStep(1,"Por favor siga las
    instrucciones para tomar la muestra",2);
workflow.addStep(step);
```

Código 6.19: InformationStep en Java.

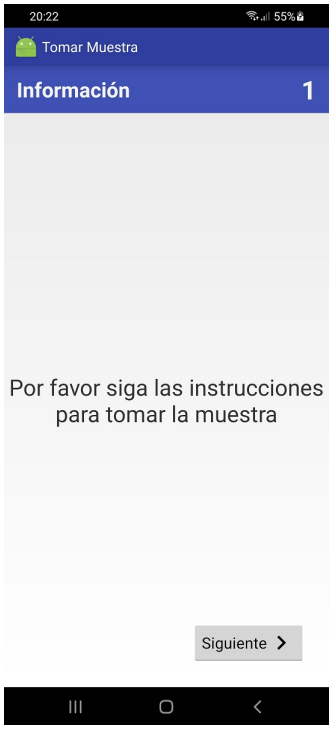

Figura 6.4: InformationStep en ejecución.

6.6.3.1. InformationStepResult: El resultado de Mostrar información

El resultado de mostrar información es nulo, es una clase vacía. Solo está para cerrar el circuito.

6.6.4. SelectOneStep: Seleccionar una opción de un grupo de opciones

SelectOneStep se usa para solicitarle al científico ciudadano que seleccione una sola opción de varias opciones disponibles. Tiene un texto (title) que se muestran a modo de título o instrucciones. Las opciones disponibles para seleccionar son una lista de objetos SelectOneOption. Cada objeto SelectOneOption tiene un id, textToShow y nextStepId, que se muestran en pantalla en forma de radio buttons.

Este caso particular de Step no tiene un identificador del siguiente Step a ejecutar (nextStepId), sino que cada opción seleccionable posee uno. Esto

permite crear bifurcaciones en el Workflow, por ejemplo, se puede preguntar si se observa cierta característica y si la respuesta es si, pasar un PhotoStep para pedir que tome una foto de esa característica, pero si la respuesta es no, se puede saltar el paso de tomar una foto.

```
{
  "id": 1,
  "type": "SelectOne",
  "title": "Indique si observa árboles",
  "options": [{
      "id": 1,
      "text": "Muchos árboles",
      "nextStepId": 2
    }, {
      "id": 2,
      "text": "Pocos árboles",
      "nextStepId": 2
    }, {
      "id": 4,
      "text": "No hay árboles",
      "nextStepId": 3
    }]
}
```

Código 6.20: SelectOneStep usando el generador de clases.

```
ArrayList<SelectOneOption> optionsToSelect = new ArrayList<>();
optionsToSelect.add(new SelectOneOption(1,"Muchos árboles",2));
optionsToSelect.add(new SelectOneOption(2,"Pocos árboles",2));
optionsToSelect.add(new SelectOneOption(3,"No hay árboles",3));

Step step = new SelectOneStep(1, optionsToSelect, "Indique si
    observa árboles");
workflow.addStep(step);
```

Código 6.21: SelectOneStep en Java.

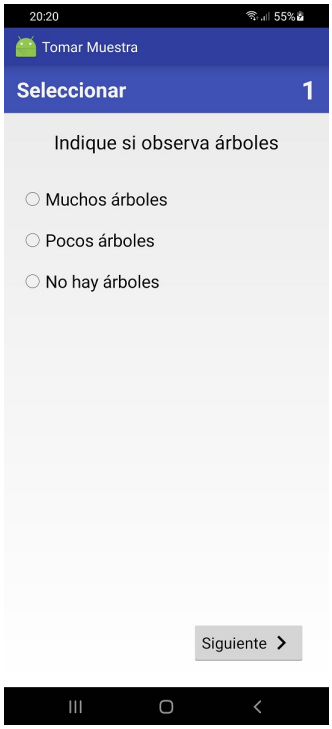

Figura 6.5: SelectOneStep en ejecución.

6.6.4.1. SelectOneStepResult: El resultado de Seleccionar una opción de un grupo de opciones

SelectOneStepResult contiene la opción seleccionada (un objeto SelectOneOption).

6.6.5. MultipleSelectStep: Seleccionar varias opciones de un grupo de opciones

MultipleSelectStep se usa para solicitarle al científico ciudadano que seleccione una o más opciones de varias opciones disponibles. Tiene un texto (title) que se muestran a modo de título o instrucciones. Las opciones disponibles para seleccionar son una lista de objetos MultipleSelectOption. Cada objeto MultipleSelectOption tiene un id, textToShow. Las opciones disponibles para seleccionar se muestran en pantalla en forma de check-boxes.

```
{
  "id": 1,
  "type": "MultipleSelect",
  "title" : "Seleccione lo que observa",
  "options": [{
          "id": 1,
          "text": "Árboles"
        }, {
          "id": 2,
          "text": "Basura"
        }, {
          "id": 3,
          "text": "Agua"
        }, {
          "id": 4,
          "text": "Animales"
        }],
  "nextStepId": 2
}
```

Código 6.22: MultipleSelectStep usando el generador de clases.

```
ArrayList<MultipleSelectOption> optionsToSelect = new ArrayList
    <MultipleSelectOption>();
optionsToSelect.add(new MultipleSelectOption(1,"Árboles"));
optionsToSelect.add(new MultipleSelectOption(2,"Basura"));
optionsToSelect.add(new MultipleSelectOption(3,"Agua"));
optionsToSelect.add(new MultipleSelectOption(4,"Animales"));

Step step = new MultipleSelectStep(1, optionsToSelect,
     "Seleccione lo que observa", 2);
workflow.addStep(step);
```

Código 6.23: MultipleSelectStep en Java.

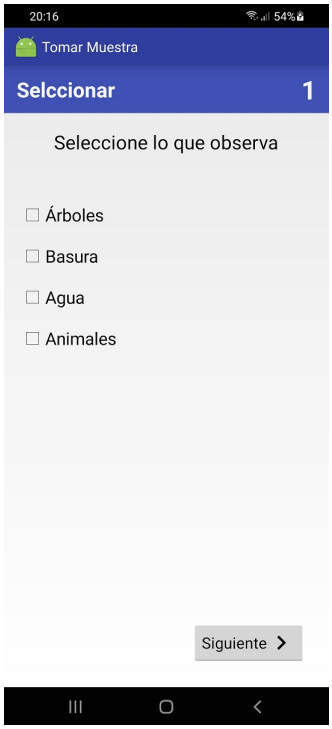

Figura 6.6: MultipleSelectStep en ejecución.

6.6.5.1. MultipleSelectStepResult: El resultado de Seleccionar varias opciones de un grupo de opciones

MultipleSelectStepResult contiene una lista de las opciones seleccionadas (una lista de objetos MultipleSelectOption).

6.6.6. LocationStep: Posicionar la muestra en el mapa con el GPS

LocationStep se usa para solicitarle al científico ciudadano que tome la posición con el GPS del dispositivo móvil. Tiene un texto (text) que se muestran a modo de instrucciones o mensaje.

Permite usar el GPS o posicionar la muestra manualmente en el mapa.

```
{
  "id": 1,
  "type": "Location",
  "text": "Por favor posicione la muestra en el mapa",
  "nextStepId": 2
}
```

Código 6.24: LocationStep usando el generador de clases.

```
Step step = new LocationStep(1,"Por favor posiciones la muestra
    en el mapa",2);
workflow.addStep(step);
```

Código 6.25: LocationStep en Java.

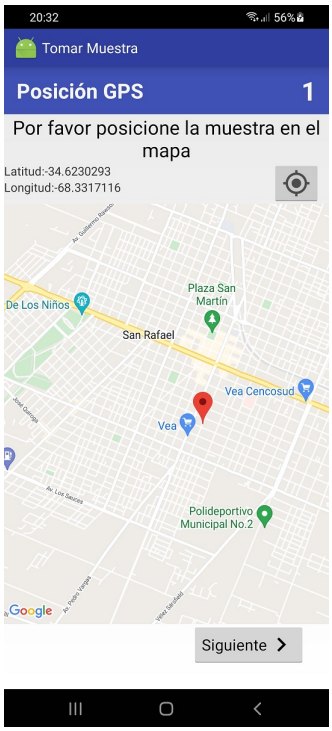

Figura 6.7: LocationStep en ejecución.

6.6.6.1. LocationStepResult: El resultado de Posicionar la muestra en el mapa con el GPS

LocationStepResult contiene la latitud y longitud (ambos de tipo double) de la posición GPS tomada.

6.6.7. RouteStep: Grabar un recorrido en el mapa usando el GPS

LocationStep se usa para solicitarle al científico ciudadano que grabe un recorrido con el GPS del dispositivo móvil. Tiene las siguientes propiedades:

- **text:** un texto que se muestran a modo de instrucciones o mensaje.
- **interval:** [Opcional] el intervalo en milisegundos en el que se pedirán actualizaciones del GPS. Si no se especifica el valor por defecto es 5000.
- **mapZoom:** [Opcional] el zoom con el que se muestra el mapa. Si no se especifica el valor por defecto es 15.

```
{
  "id": 1,
  "type": "Route",
  "text": "Inicie el recorrido cuando este listo",
  "interval": 30000,
  "mapZoom": 20,
  "nextStepId": 2
}
```

Código 6.26: RouteStep usando el generador de clases.

```
RouteStep step = new RouteStep(1,"Inicie el recorrido cuendo
    esté listo",2);
step.setInterval(30000);
step.setMapZoom(20);
workflow.addStep(step);
```

Código 6.27: RouteStep en Java.

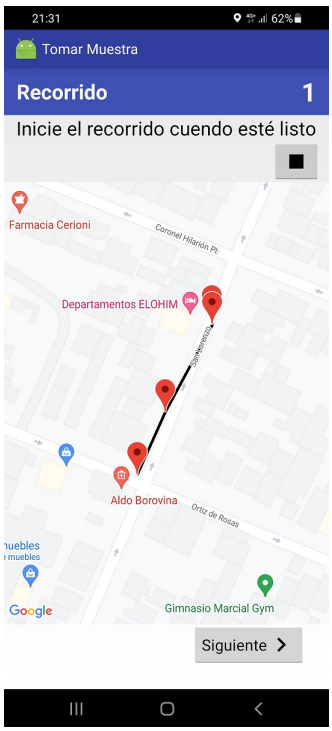

Figura 6.8: RouteStep en ejecución.

6.6.7.1. RouteStepResult: El resultado de Grabar un recorrido en el mapa usando el GPS

RouteStepResult tiene una lista de objetos de la clase Location proporcionada por Android. Para mas información sobre esta clase ver la documentación oficial de Android sobre Location [5].

6.6.8. InsertTextStep: Ingresar texto

InsertTextStep se usa para solicitarle al científico ciudadano que ingrese una anotación, que puede ser texto o números. Tiene las siguientes propiedades:

- **text:** un texto que se muestran a modo de instrucciones o mensaje.
- **sampleText:** un texto, a modo de muestra.

[5] https://developer.android.com/reference/android/location/Location

- **maxLength:** establece la cantidad máxima de caracteres permitidos.
- **inputType:** indica el tipo de caracteres admitidos. Los valores pueden ser **text** para el ingreso de texto, **number** para el ingreso de número enteros, **decimal** para el ingreso de números con decimales.
- **optional:** un booleano que indica si se puede dejar vacío (true) o si es obligatorio que ingrese una anotación (false).

```
{
  "id": 1,
  "type": "InsertText",
  "text": "Por favor, ingrese el nombre del lago",
  "sampleText" : "Nombre del lago",
  "inputType" : "text",
  "maxLength" : 50,
  "optional" : true,
  "nextStepId": 2
}
```

Código 6.28: InsertTextStep usando el generador de clases.

```
Step step = new InsertTextStep(1,"Por favor, ingrese el nombre
    del lago","Nombre del lago",50, InsertTextStep.InputType.
    TYPE_TEXT, true,2);
workflow.addStep(step);
```

Código 6.29: InsertTextStep en Java.

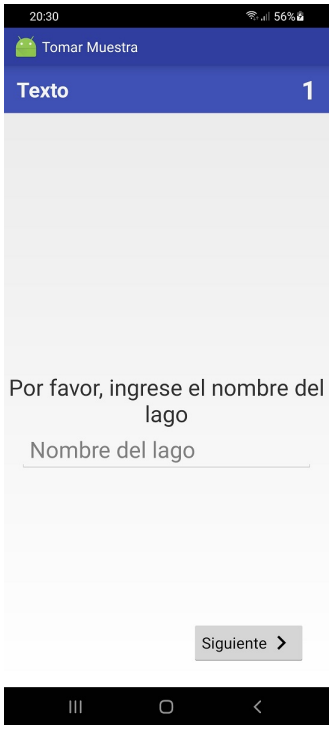

Figura 6.9: InsertTextStep en ejecución.

6.6.8.1. InsertTextStepResult: El resultado de Ingresar texto

InsertTextStepResult contiene el String con la anotación ingresada.

6.6.9. InsertDateStep e InsertTimeStep: Ingresar fecha y hora

InsertDateStep e InsertTimeStep se usan para solicitarle al científico ciudadano que ingrese una fecha y una hora respectivamente. Tienen un texto que se muestran a modo de instrucciones o mensaje.

```
{
{
  "id": 1,
  "type": "InsertDate",
  "text": "Ingrese la fecha que encontró el mosquito",
  "nextStepId": 2
},
{
  "id": 2,
  "type": "InsertTime",
  "text": "Ingrese la hora que encontró el mosquito",
  "nextStepId": 3
}

}
```

Código 6.30: InsertDateStep e InsertTimeStep usando el generador de clases.

```
Step step = new InsertDateStep(1,"Ingrese la fecha que encontró
    el mosquito",2);
workflow.addStep(step);

step = new InsertTimeStep(2,"Ingrese la hora que encontró el
   mosquito",3);
workflow.addStep(step);
```

Código 6.31: InsertDateStep e InsertTimeStep en Java.

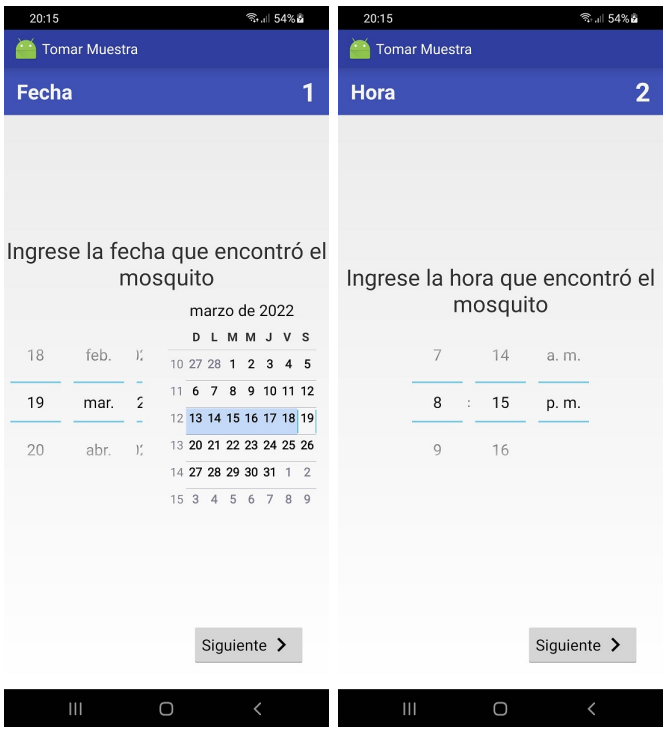

Figura 6.10: InsertDateStep (izquierda) e InsertTimeStep (derecha) en ejecución.

6.6.9.1. InsertDateStepResult e InsertTimeStep: El resultado de Ingresar fecha y hora

InsertDateStepResult e InsertTimeStep tienen un objeto Date de Java que tiene la fecha o la hora según corresponda.

6.6.10. Definir un nuevo Step, StepFragment y StepResult

Como se explicó en la sección 5.2.1 un Step tiene los parámetros necesarios para que se ejecute un StepFragment y genere un StepResult con la interacción del usuario.

Para definir un nuevo Step personalizado, se debe definir también un StepFragment personalizado que sepa ejecutarlo y StepResult personalizado que contendrá el resultado de esa ejecución. A continuación se explica cómo definir cada uno de ellos.

6.6.10.1. Definir un nuevo Step

Para definir un nuevo Step personalizado se debe implementar la interface Step, la cual extiende Serializable y define los siguientes métodos que se deben implementar:

- ***String getStepResourceName()*:** debe devolver el nombre o etiqueta del Step, por ejemplo «Tomar foto».
- ***int getId()*:** debe devolver el id del Step.
- ***<T extends StepFragment>***

  ***Class<T>getStepFragmentClass()*:** debe devolver la clase del StepFragment que lo ejecutará.
- ***<T extends StepFragment>***

  ***void setStepFragmentClass(Class<T>type)*:** se usa para establecer la clase del StepFragment que lo ejecutará.
- ***@Nullable Integer getNextStepId(StepResult stepResult)*:** debe devolver el id del siguiente Step a ejecutar en el Workflow. Puede ser null y en ese caso se considerará el final del Workflow.
- ***@Nullable Integer getHelpResourseId()*:** debe devolver el id del recurso que se usará para mostrar la ayuda del Step. Puede ser null y en ese caso se considerará que el Step no tiene ayuda para mostrar.
- ***void setHelpResourseId(Integer helpResourseId)*:** se usa para establecer el recurso que se usará para mostrar la ayuda del Step.

6.6.10.2. Definir un nuevo StepFragment

Para definir un nuevo StepFragment personalizado se debe extender la clase StepFragment, la cual es una clase abstracta que extiende de Fragment y define los siguientes métodos que se deben implementar:

- ***int getLayoutResource()*:** debe devolver el «Layout Resource» que contiene diseño con la estructura de la interfaz de usuario. Es el que se usará en el método *onCreateView* para cargar dicha estructura de interfaz de usuario.

- ***void onCreateViewStepFragment(View rootView, Bundle savedInstanceState)*:** este método se llama internamente en el método *onCreateView* luego de cargar la estructura de la interfaz de usuario y debe usarse para inicializar y guardar referencia a los elementos de la IU. El parámetro *rootView* contiene la estructura de la interfaz de usuario cargada y el parámetro *savedInstanceState* contiene el estado de instancia guardado que se recibe en el método *onCreateView*.

- ***boolean validate()*:** este método se llama internamente cuando el usuario aprieta el botón «Siguiente» y se debe usar para hacer las validaciones pertinentes antes de finalizar el StepFragment, como por ejemplo validar que el usuario haya seleccionado una opción de las dadas. Debe devolver **true** si la validación es exitosa o **false** en caso contrario.

- ***StepResult getStepResult()*:** debe devolver el StepResult generado.

6.6.10.3. Definir un nuevo StepResult

Para definir un nuevo StepResult personalizado se debe implementar la interface StepResult, la cual extiende Serializable y define los siguientes métodos que se deben implementar:

- ***Step getStep()*:** debe devolver el Step que generó el StepFragment.

- ***boolean hasMultimediaFile()*:** debe devolver **true** si el StepResult tiene un archivo multimedia (foto, video, audio, etc.) asociado o **false** en caso contrario.

- ***String getMultimediaFileName()*:** debe devolver el nombre del archivo multimedia asociado al StepResult.

Capítulo 7

Caso de uso

En esta sección se presenta un caso de uso del framework Samplers tomando como ejemplo una app de ciencia ciudadana que ya se encuentra en funcionamiento, que es AppEAR, y se generará una versión usando Samplers, haciendo una breve comparación entre ambas.

7.1. AppEAR

AppEAR un sistema de ciencia ciudadana para cuidar y aprender de los ambientes acuáticos en Argentina, realizado por Joaquín Cochero, investigador del CONICET en el Instituto Platense de Limnología. Los científicos ciudadanos interactúan y colaboran en el proyecto de varias maneras, mientras que aprenden sobre los ambientes y responden a objetivos científicos. El objetivo final de AppEAR es tener un relevamiento completo y detallado de las aguas continentales de todo el territorio nacional para conocer los lugares en riesgo en los que urge trabajar.

Los voluntarios de este proyecto descargan una aplicación para su dispositivo móvil y toman muestras para el proyecto. La aplicación guía a los usuarios a través de los pasos necesarios para tomar una muestra.[3]

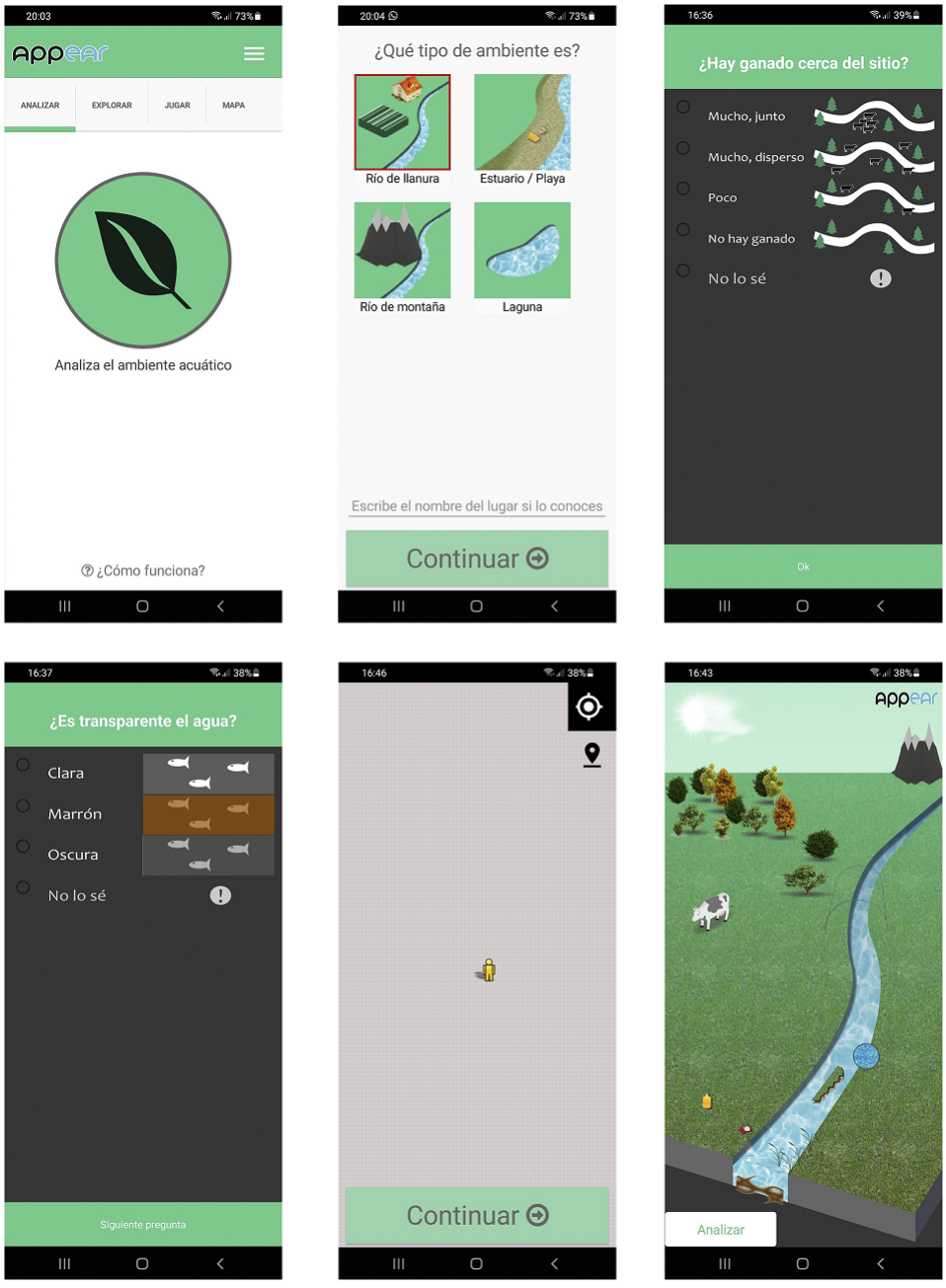

Figura 7.1: Capturas de pantalla de la app AppEAR

Como se mencionó antes, la app va guiando a los científicos ciudadanos a través de los pasos necesarios para tomar una muestra, que son básicamente preguntas acerca del lugar que están relevando y de las cosas que ven alrededor. AppEAR diferencia 4 tipos de ambientes, que son río de llanura, río de montaña, estuario/playa y laguna, y en base a eso realiza algunas preguntas que son específicas de cada tipo y el resto preguntas comunes a los 4 tipos de ambientes. A medida que se van completando las preguntas, la app va armando una especie de dibujo del lugar con figuras que representan los elementos observados por el científico ciudadano.

Analizando todos los pasos propuestos por AppEAR se puede deducir el Workflow para tomar la muestra representado en el siguiente grafo direccional:

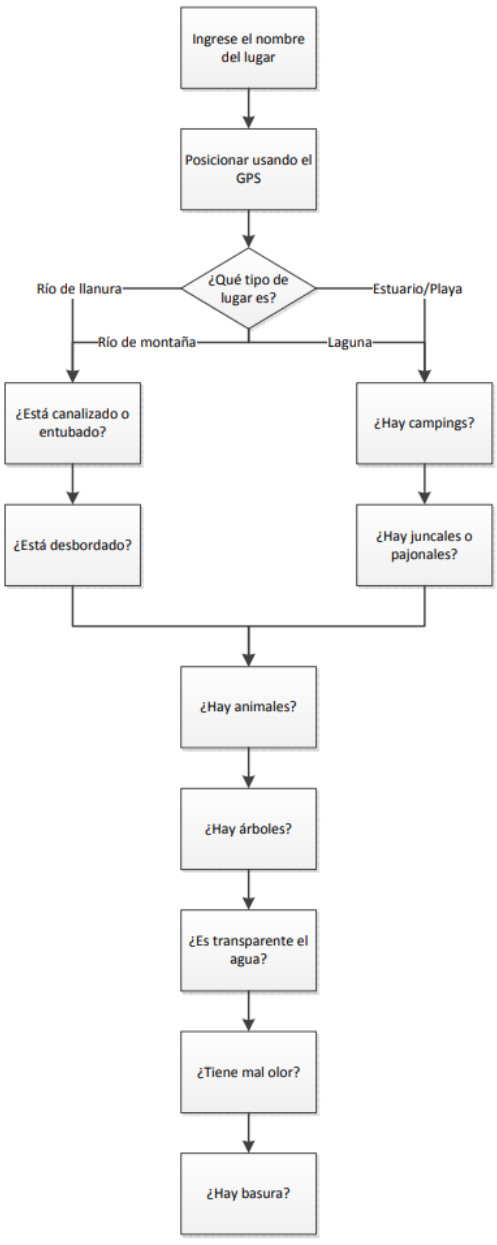

Figura 7.2: Grafo direccional que representa el Workflow de AppEAR

7.2. AppEar usando Samplers

A continuación se muestra como sería AppEAR usando Samplers.

Observando el Workflow planteado en la figura 7.1, se genera el archivo de configuración para Samplers de la siguiente manera:

```
{
  "project": {
    "app_path": "app/src/main/java/com/example/AppEAR/",
    "package_name": "com.example.AppEAR"
  },
  "application": {
    "title": "AppEAR",
    "welcomeMessage": "Analiza el ambiente acuático",
    "networkConfiguration": {
      "url": "https://192.168.0.10/samplers/upload.php",
      "paramName": "sample",

    },
    "googleMaps_API_KEY": "YOUR_GOOGLE_API_KEY",
    "mainHelpFileName": "mainhelp.html"
  },
  "workflow": {
    "actionLabel": "Analizar",
    "steps": [
      {
        "id": 1,
        "type": "InsertText",
        "text": "Escriba el nombre del lugar si lo conoce",
        "sampleText" : "Nombre del lugar",
        "inputType" : "text",
        "maxLength" : 50,
        "optional" : true,
        "nextStepId": 2
      },
      {
        "id": 2,
        "type": "Location",
        "text": "Posicione el lugar en el mapa",
        "nextStepId": 3
      },
```

```
        {
          "id": 3,
          "type": "SelectOne",
          "title": "¿Qué tipo de ambiente es?",
          "options": [{
              "id": 1,
              "text": "Río de llanura",
              "nextStepId": 4
            }, {
              "id": 2,
              "text": "Río de montaña",
              "nextStepId": 4
            }, {
              "id": 3,
              "text": "Estuario/Playa",
              "nextStepId": 6
            },
            {
              "id": 4,
              "text": "Laguna",
              "nextStepId": 6
            }]
        },
        {
          "id": 4,
          "type": "SelectOne",
          "title": "¿Está canalizado o entubado?",
          "options": [{
              "id": 1,
              "text": "Entubado/canalizado",
              "nextStepId": 5
            }, {
              "id": 2,
              "text": "Sin entubar/canalizar",
              "nextStepId": 5
            }, {
              "id": 3,
              "text": "No lo sé",
              "nextStepId": 5
            }]
        },
```

```
        {
          "id": 5,
          "type": "SelectOne",
          "title": "¿Está desbordado el cauce?",
          "options": [{
              "id": 1,
              "text": "Desbordado",
              "nextStepId": 8
            }, {
              "id": 2,
              "text": "Sin desbordar",
              "nextStepId": 8
            }, {
              "id": 3,
              "text": "No lo sé",
              "nextStepId": 8
            }]
        },
        {
          "id": 6,
          "type": "SelectOne",
          "title": "¿Hay campings en la costa cerca tuyo?",
          "options": [{
              "id": 1,
              "text": "Hay campings o casillas",
              "nextStepId": 7
            }, {
              "id": 2,
              "text": "Sin campings ni casillas",
              "nextStepId": 7
            }, {
              "id": 3,
              "text": "No lo sé",
              "nextStepId": 7
            }]
        },
```

```
    {
      "id": 7,
      "type": "SelectOne",
      "title": "¿Hay juncales o pajonales?",
      "options": [{
          "id": 1,
          "text": "Si, muchos",
          "nextStepId": 8
        }, {
          "id": 2,
          "text": "Si, pocos",
          "nextStepId": 8
        }, {
          "id": 3,
          "text": "No, sin juncales ni pajonales",
          "nextStepId": 8
        }, {
          "id": 4,
          "text": "No lo sé",
          "nextStepId": 8
        }]
    },
    {
      "id": 8,
      "type": "SelectOne",
      "title": "¿Hay ganado cerca del sitio?",
      "options": [{
          "id": 1,
          "text": "Mucho, junto",
          "nextStepId": 9
        }, {
          "id": 2,
          "text": "Mucho, disperso",
          "nextStepId": 9
        }, {
          "id": 3,
          "text": "Poco",
          "nextStepId": 9
        }, {
          "id": 4,
          "text": "No hay ganado",
          "nextStepId": 9
        }, {
          "id": 5,
          "text": "No lo sé",
          "nextStepId": 9
        }]
    },
```

```
        {
          "id": 9,
          "type": "SelectOne",
          "title": "¿Cómo son los árboles?",
          "options": [{
              "id": 1,
              "text": "Árboles poco variados",
              "nextStepId": 10
            }, {
              "id": 2,
              "text": "Árboles muy variados",
              "nextStepId": 10
            }, {
              "id": 3,
              "text": "Sin árboles",
              "nextStepId": 10
            }, {
              "id": 4,
              "text": "No lo sé",
              "nextStepId": 10
            }]
        },
        {
          "id": 10,
          "type": "SelectOne",
          "title": "¿Es transparente el agua?",
          "options": [{
              "id": 1,
              "text": "Clara",
              "nextStepId": 11
            }, {
              "id": 2,
              "text": "Marrón",
              "nextStepId": 11
            }, {
              "id": 3,
              "text": "Oscura",
              "nextStepId": 11
            }, {
              "id": 4,
              "text": "No lo sé",
              "nextStepId": 11
            }]
        },
```

```
205       {
206         "id": 11,
207         "type": "SelectOne",
208         "title": "¿Tiene mal olor?",
209         "options": [{
210             "id": 1,
211             "text": "No tiene olor",
212             "nextStepId": 12
213           }, {
214             "id": 2,
215             "text": "Tiene feo olor",
216             "nextStepId": 12
217           }, {
218             "id": 3,
219             "text": "No lo sé",
220             "nextStepId": 12
221           }]
222       },
223       {
224         "id": 12,
225         "type": "SelectOne",
226         "title": "¿Hay basura en la costa?",
227         "options": [{
228             "id": 1,
229             "text": "No hay"
230           }, {
231             "id": 2,
232             "text": "Poca"
233           }, {
234             "id": 3,
235             "text": "Mucha"
236           }, {
237             "id": 4,
238             "text": "No lo sé"
239           }]
240       }
241     ]
242   }
243 }
```

Código 7.1: Archivo de configuración para Samplers para generar una app equivalente a AppEAR.

Una vez completado el archivo de configuración, se ejecuta Samplers y se genera la app.

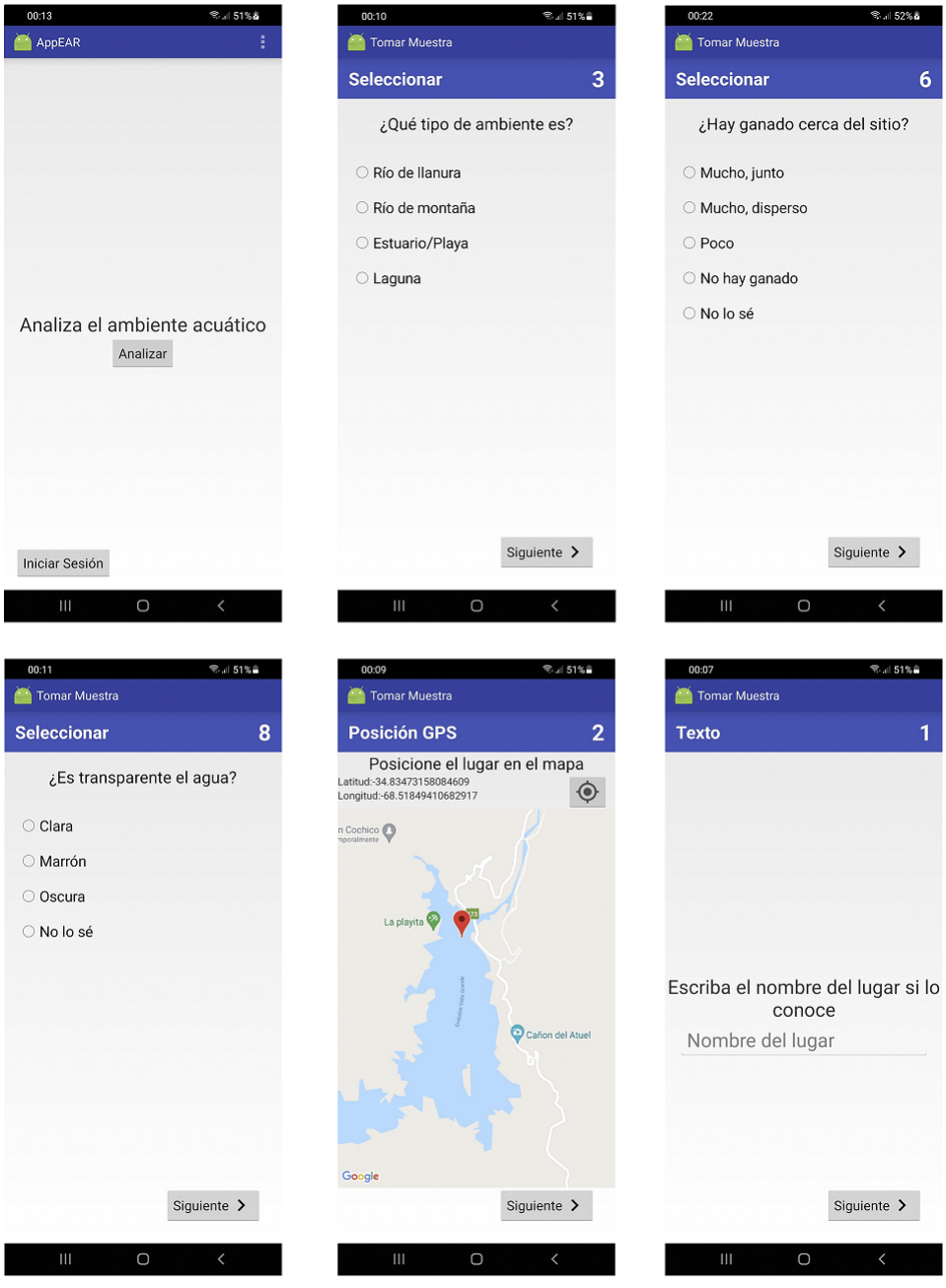

Figura 7.3: Capturas de pantalla de la app AppEAR generada por Samplers

7.3. Comparación y conclusión

Analizando y comparando ambas aplicaciones móviles, podemos ver algunas diferencias que se detallan a continuación.

Algunos pasos deben dividirse en dos pantallas: por ejemplo, el paso de seleccionar el tipo de ambiente e introducir el nombre del lugar están en una misma pantalla en AppEAR, pero en la app generada por Samplers están en dos pantallas diferentes, ya que se resuelven con un SelectOneStep para la selección del tipo de ambiente y un InsertTextStep para introducir el nombre del lugar.

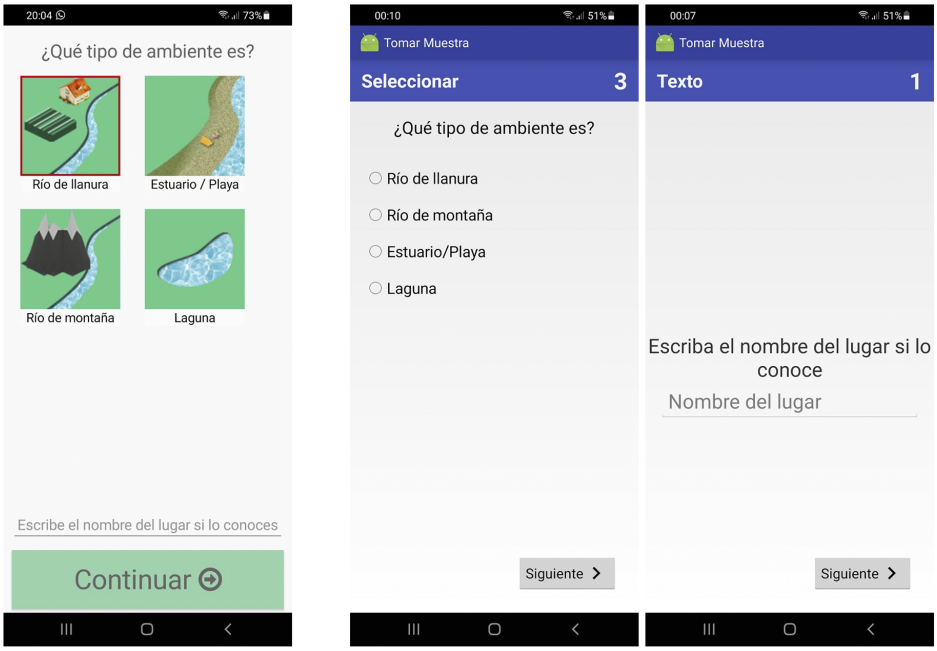

Figura 7.4: Izquierda: tipo de ambiente y nombre en una sola pantalla (AppEAR). Derecha: tipo de ambiente y nombre en dos pantallas (Samplers)

La selección de opciones no tiene imágenes: en la mayoria de los pasos de seleccionar alguna opción en AppEAR tienen imágenes de referencia al lado de cada opción, pero en la app generada por Samplers estas imágenes no están disponibles porque SelectOneStep solo muestra los radio-buttons con el texto de la opción para seleccionar.

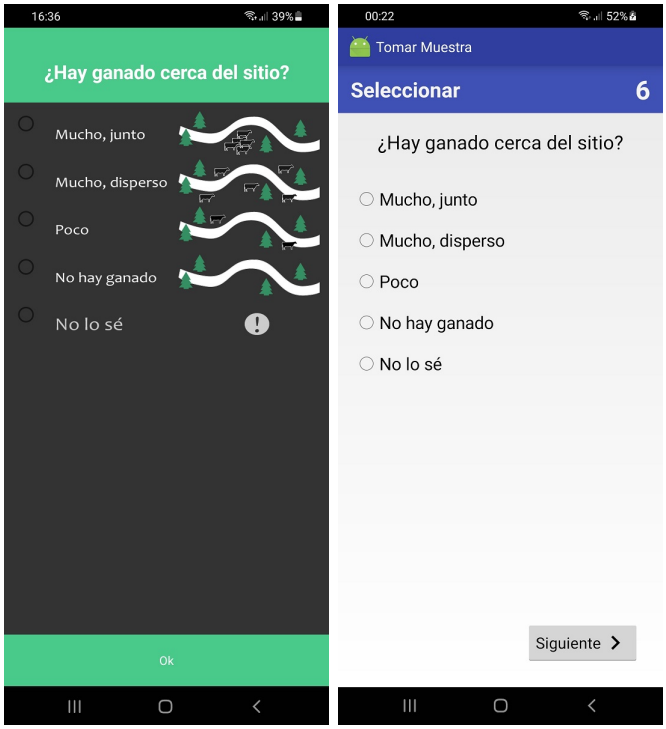

Figura 7.5: Izquierda: selección con imágenes (AppEAR). Derecha: selección simple (Samplers)

Pantalla principal sin imágenes: la pantalla principal en AppEAR tiene una imagen de fondo, que no está en la app generada por Samplers. Igualmente esta imagen se puede agregar con bastante facilidad, ya que como Samplers genera el código de la app la Activity principal (que es la que provee la pantalla principal) se puede editar, y Android Studio provee herramientas visuales muy intuitivas para diseñar la interfaz de una pantalla.

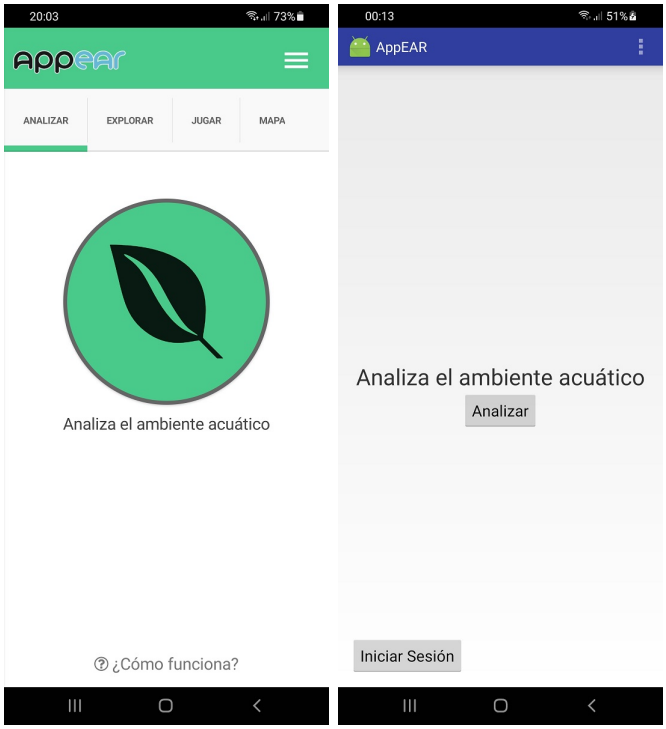

Figura 7.6: Izquierda: pantalla principal con imágenes (AppEAR). Derecha: pantalla principal sin imágenes (Samplers)

Pantallas especiales que no están: hay pantallas especiales en AppEAR que no representan ningún paso para tomar la muestra, pero que están dentro de la secuencia, como por ejemplo la pantalla que va mostrando un esquema del lugar que se está analizando con imágenes predeterminadas a medida que el usuario va completando los pasos. En Samplers no hay un Step que pueda representar eso, por lo que no se pueden agregar.

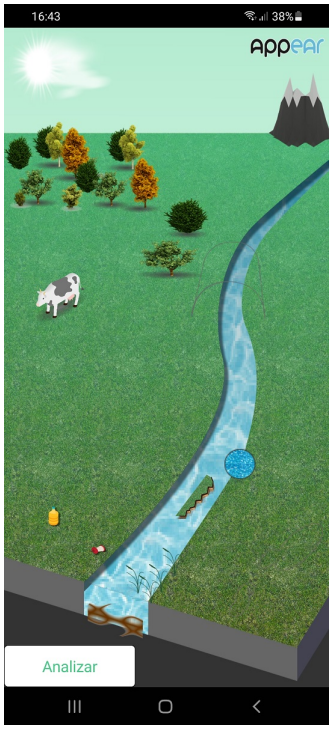

Figura 7.7: Ejemplo de pantalla especial (AppEAR)

Los mapas se muestran correctamente: en AppEAR al posicionar el lugar que se está analizando, si bien toma la posición con el GPS correctamente, el mapa de fondo no se muestra. En Samplers se muestra el mapa que provee la API de GoogleMaps y que funciona correctamente en todos los dispositivos móviles con Android.

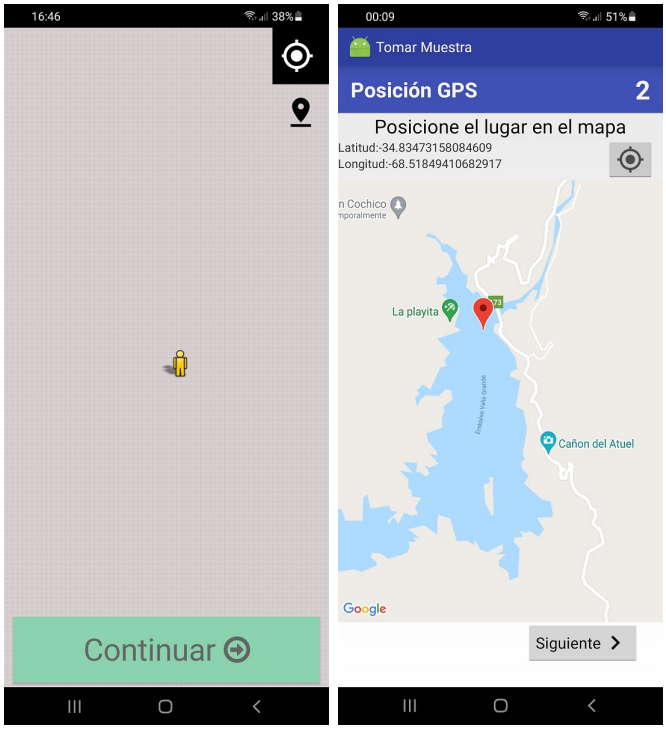

Figura 7.8: Izquierda: mapa con problemas de visualización (AppEAR). Derecha: mapa que se muestra correctamente (Samplers)

Más allá de las diferencias encontradas, la app generada por Samplers es funcionalmente equivalente a AppEAR y sirve para recolectar las muestras que necesita el proyecto. La ventaja principal de Samplers es que no se requieren conocimientos de programación para generar la app, sólo hay que completar el archivo JSON que se muestra en el código 7.1, lo que también reduce considerablemente el tiempo de desarrollo de la app. Además como Samplers genera el código de la app, la misma se puede editar para personalizarla y agregarle más funcionalidades, por lo que puede ser usada por un programador como punto de partida de un proyecto.

Capítulo 8

Conclusiones y Trabajo Futuro

En este capítulo se recordarán los objetivos propuestos al comienzo de este trabajo, los resultados obtenidos y en qué medida se cumplieron dichas metas. También se detallaran posibles trabajos a futuro que contribuirían a mejorar aspectos del proyecto como puede ser mejorar su usabilidad, mantener el proyecto para que siga cumpliendo los estándares requeridos para aplicaciones móviles o implementar una versión para iOS para ampliar la base de usuarios.

8.1. Conclusiones

En la sección 1.2 se definieron los objetivos de este trabajo, los cuales analizaremos si fueron cumplidos, cómo y en qué medida se alcanzaron.

A continuación se detallan dichos objetivos con su respectivo análisis:

- **Desarrollar un framework para instanciar aplicaciones móviles Android de ciencia ciudadana. El framework recibirá un archivo con la configuración requerida en formato JSON y generará una aplicación para ejecutarse en un dispositivo Android.**

  Como se detalla en el capítulo 5, el framework Samplers recibe un archivo de configuración dónde se especifican los pasos necesarios para recolectar una muestra y con ello produce el código de una aplicación móvil para Android. El código queda listo para compilar y ejecutar en

un dispositivo móvil con Android o en un emulador virtual de los que provee Android Studio.

De esta manera no solo se genera una app sino que al brindar el código fuente ésta puede ser personalizada y modificada para agregarle más funcionalidades en caso de que el usuario tenga conocimientos de programación.

- **En este archivo estará el conjunto de pasos que especifican el protocolo de recolección de muestras. Estos pasos pueden ser: captura de una foto, un video, un audio, una ubicación o un recorrido hecho con el dispositivo móvil, contestar una pregunta con respecto a la muestra -con una o múltiples respuestas posibles-, introducir anotaciones de texto, indicar una fecha y hora, mostrar información de orientación y ayuda para la toma de la muestra.**

  Como se explica en la sección 5.3.1 en el archivo de configuración que recibe Samplers debe venir configurado el Workflow que representa los pasos a seguir para tomar una muestra. Cada uno de los Steps que provee Samplers para armar el Workflow permite realizar cada una de las acciones especificadas en el objetivo. El detalle de los Steps que provee Samplers y como usarlos se explica en el anexo 6.6. Además Samplers permite que un usuario con conocimientos de programación defina sus propios Steps como se explica en la sección 6.6.10 del mencionado anexo.

- **La aplicación generada servirá para tomar muestras siguiendo el protocolo de recolección especificado y las almacenará y empaquetará en el dispositivo móvil hasta que pueda ser enviada a un servidor web.**

  La aplicación generada por Samplers almacena las muestras tomadas en el almacenamiento interno del dispositivo móvil como se explicó en la sección 5.2.3. Dichas muestras son comprimidas en formato ZIP y enviadas al servidor web previamente configurado cuando se detecta conexión wifi como se explicó en la sección 5.2.4.

- **Definir el formato del archivo de configuración de la aplicación y la información adicional necesaria, como pueden ser credenciales para acceder a los servicios de Google Services o el posicionamiento por GPS.**

Se definió el formato que debe tener el archivo de configuración que recibe Samplers como se explicó en la sección 5.3.1. La especificación detallada del mismo se encuentra en el anexo 6.3

- **Instanciar una aplicación básica de ejemplo con el framework en base a un archivo de configuración, que permita tomar algunas muestras y enviarlas a un servidor web que estará configurado para dicho propósito.**

  Se instanció una app usando Samplers que imita a una app de ciencia ciudadana que ya está en funcionamiento (AppEAR) y se hizo una comparación en el capítulo 7 analizando las diferencias. Para probar el envío de muestras se montó un servidor local en PHP, el cual se deja a disposición en el repositorio de Samplers en GitHub.

En base a lo detallado anteriormente, podemos concluir que los objetivos fueron alcanzados en su totalidad.

8.2. Trabajo Futuro

En esta sección se proponen posibles caminos a seguir para que el framework evolucione, pero que quedan fuera del alcance de este trabajo. Poder compilar para iOS ampliaría la base de usuarios. Aunque en el país el uso de dispositivos iOS, como mencionamos previamente en la sección 2.6, representa un 6 % en contraposición con el 93 % que representa Android, sigue siendo un porcentaje que queda excluido de la app que se puede crear con Samplers. Para desarrollar en iOS se debe contar con una computadora Mac que pueda ejecutar la última versión de Xcode. De la misma manera que Android Studio es el IDE que deben utilizar las apps desarrolladas para dispositivos Android, Xcode es el IDE para apps Apple que se ejecuten en Mac o en iOS. Para publicar la app en la App Store se debe ser miembro del Apple Developer Program. Teniendo en cuenta que es necesario contar con hardware específico [38] y que se requiere unirse al Apple Developer Program que tiene un costo anual [39] lo hace una mejora con un costo económico ya más alto que el que se necesita para desarrollar en Android.

Android actualiza regularmente su sistema operativo y en cada nueva versión introduce cambios, mejoras y nuevos requerimientos para las apps. Un trabajo a futuro sería mantener el código fuente actualizado para ajustarse a los cambios del sistema operativo, aprovechar las mejoras que puedan llegar a existir y respetar los estándares de seguridad requeridos. Lo mismo

si hiciera falta una adecuación a las políticas de privacidad, en el caso de que cambien a políticas más restrictivas.

Una mejora ya realizada por los participantes del trabajo de tesina Samplers2[40] es Muestre.AR, una interfaz web que permite a los investigadores definir el workflow de recolección de una muestra utilizando un sitio web y descargar la aplicación Android resultante. Con las instrucciones definidas por el usuario investigador, el sitio puede generar el archivo de configuración e ingresarlo en Samplers para crear una aplicación Andorid y descargar la app.

Anexos

Anexo A

Instalación del framework

En esta sección se describe como instalar el framework en Android Studio; se detallan los requisitos y los pasos a seguir para la instalación.

A.1. Requerimientos mínimos:

- Android Studio (Java): Si bien esta pensado para y probado en Android Studio, podria funcionar bien en otro entorno que use Java y deje importar archivos Android Archive (.aar).
- Android SDK API17: Android 4.2 (Jelly Bean) o superior.

A.2. Pasos para la Instalación:

1. Crear en Android Studio un nuevo proyecto vacío (sin ninguna Activity).
    - Seleccionar API17 o superior como versión mínima de Android SDK
2. Importar la librería del framework en el proyecto creado:
    - Descargar la última versión de **samplersFramework.aar** desde https://github.com/cientopolis/samplers/releases/
    - Importar la librería al proyecto: File -> New -> New Module -> Import .JAR/.AAR Package

3. Agregar el repositorio de Google:

   En el archivo build.gradle del proyecto agregar:

```
allprojects {
    repositories {
        mavenCentral()
        google()
    }
}
```

Código A.1: Configuración del repositorio de Google(línea 4)

4. Agregar las dependencias necesarias:

   En el archivo build.gradle de la aplicación agregar:

```
dependencies {
  implementation fileTree(dir: 'libs', include: ['*.jar'])
  androidTestImplementation('androidx.test.espresso:espresso-
    core:3.1.0', {
    exclude group: 'com.android.support', module: 'support-
    annotations'
  })

  implementation 'com.google.android.material:material:1.4.0'
  testImplementation 'junit:junit:4.13.2'

  // agregar la dependencia del framework Samplers
  compile project(":samplersFramework")
}
```

Código A.2: Configuración de la dependencia del framework (línea 11)

5. Instanciar:

   La instanciación puede ser manual o usando el generador de clases en Gradle como se explicó en la sección 6.

Bibliografía

[1] A. Wiggins and K. Crowston, "From conservation to crowdsourcing: A typology of citizen science," in *System Sciences (HICSS), 2011 44th Hawaii international conference on*, pp. 1–10, IEEE, 2011.

[2] J. Silvertown, "A new dawn for citizen science," *Trends in ecology & evolution*, vol. 24, no. 9, pp. 467–471, 2009.

[3] Joaquín Cochero, "Appear." https://app-ear.com.ar/. [Online; último acceso marzo/2022].

[4] LIFIA Centro asociado de la Comisión de Investigaciones Científicas, NOVA (Nuevo Observatorio Virtual Argentino), CENIT (Centro de Investigaciones para la Transformación, asociado a UNTREF), Facultad de Informática (UNLP), Facultad de Ciencias Astronómicas y Geofísicas (UNLP), "Cientópolis." https://www.cientopolis.org. [Online; último acceso: Octubre/2018].

[5] R. Louv and J. W. Fitzpatrick, *Citizen science: Public participation in environmental research*. Cornell University Press, 2012.

[6] E. C. D. Environment, "Science for environment policy indepth report: Environmental citizen science," *Science Communication Unit, University of the West of England, Bristol*, vol. 9, 2013.

[7] Audubon, "History of the Christmas Bird Count." http://www.audubon.org/conservation/history-christmas-bird-count. [Online; último acceso: mayo/2022].

[8] "Zooniverse." https://www.zooniverse.org. [Online; último acceso: mayo/2022].

[9] Juan Ignacio Yañez, Matías Celasco, Roberto Gamen, "Galaxy Conqueror: Astronomy, Citizen Scien-

ce and Gamification." https://www.cientopolis.org/wp-content/uploads/sites/40/2016/10/laclo-2016-camera-ready.pdf. [Online; último acceso: mayo/2022].

[10] "The Big Butterfly Count." https://bigbutterflycount.butterfly-conservation.org/. [Online; último acceso: mayo/2022].

[11] J. P. Cohn, "Citizen science: Can volunteers do real research?," *AIBS Bulletin*, vol. 58, no. 3, pp. 192–197, 2008.

[12] Galaxy Zoo, "The Science behind the Site." https://www.zooniverse.org/projects/zookeeper/galaxy-zoo/about/research, 4 2018.

[13] Fossil Finders, "What is Fossil Finders?." http://fossilfinder.coe.uga.edu/about/what-is-fossil-finders/, 4 2018.

[14] J. L. Shirk, H. L. Ballard, C. C. Wilderman, T. Phillips, A. Wiggins, R. Jordan, E. McCallie, M. Minarchek, B. V. Lewenstein, M. E. Krasny, *et al.*, "Public participation in scientific research: a framework for deliberate design," *Ecology and Society*, vol. 17, no. 2, 2012.

[15] R. Bonney, C. B. Cooper, J. Dickinson, S. Kelling, T. Phillips, K. V. Rosenberg, and J. Shirk, "Citizen science: a developing tool for expanding science knowledge and scientific literacy," *BioScience*, vol. 59, no. 11, pp. 977–984, 2009.

[16] S. S. Carey, *A beginner's guide to scientific method*. Cengage Learning, 2011.

[17] INDEC, "Acceso y uso de tecnologías de la información y la comunicación." https://www.indec.gob.ar/uploads/informesdeprensa/ mautic_05_19CF6C49F37A.pdf, 5 2019.

[18] Carrier y Asoc., "Mercado celular argentino 2019." https://carrieryasoc.com/index.php/2019/04/10/mercado-celular-argentino-2019/, Abril 2019. [Online; último acceso: Noviembre/2019].

[19] S. Kim, J. Mankoff, and E. Paulos, "Sensr: evaluating a flexible framework for authoring mobile data-collection tools for citizen science," in *Proceedings of the 2013 conference on Computer supported cooperative work*, pp. 1453–1462, ACM, 2013.

[20] G. Newman, A. Wiggins, A. Crall, E. Graham, S. Newman, and K. Crowston, "The future of citizen science: emerging technologies and shifting paradigms," *Frontiers in Ecology and the Environment*, vol. 10, no. 6, pp. 298–304, 2012.

[21] M. Weisfeld, *The object-oriented thought process*. Pearson Education, 2008.

[22] T. Budd, *Introduction to object-oriented programming*. Pearson Education India, 2008.

[23] M. Fayad and D. C. Schmidt, "Object-oriented application frameworks," *Communications of the ACM*, vol. 40, no. 10, pp. 32–38, 1997.

[24] R. E. Johnson, "Frameworks=(components+ patterns)," *Communications of the ACM*, vol. 40, no. 10, pp. 39–42, 1997.

[25] R. E. Johnson and B. Foote, "Designing reusable classes," *Journal of object-oriented programming*, vol. 1, no. 2, pp. 22–35, 1988.

[26] Yasser Ansari, Martin Ceperley, Peter Horvath, Bruno Kruse, "Project Noah." https://www.projectnoah.org/, 2010. NYU's Interactive Telecommunications Program.

[27] epicollect, "Epicollect." https://docs.epicollect.net/. [Online; último acceso: marzo/2022].

[28] CitSci, "CitSci." https://citsci.org/. [Online; último acceso: mayo/2022].

[29] Spotteron, "Spotteron." https://www.spotteron.net//. [Online; último acceso: junio/2022].

[30] "Zooniverse Blog." https://blog.zooniverse.org/2018/05/18/notes-on-mobile-launching-a-workflow/. [Online; último acceso: junio/2022].

[31] Google, "Documentation for android developers." https://developer.android.com/docs. [Online].

[32] Gartner, "Gartner says worldwide sales of smartphones grew 7 percent in the fourth quarter of 2016." https://www.gartner.com/en/newsroom/press-releases/2017-02-15-gartner-says-worldwide-sales-of-smartphones-grew-7-percent-in-the-fourth-quarter-of-2016, Febrero 2017. [Online; último acceso: Octubre/2018].

[33] Google, “Android studio.” https://developer.android.com/studio/. [Online; último acceso: Octubre/2018].

[34] GitHub, “Github.” https://github.com. [Online; último acceso: Octubre/2018].

[35] Android, “Documentación para desarrolladores android - opciones de almacenamiento.” https://developer.android.com/guide/topics/data/data-storage#filesInternal. [Online; último acceso: Noviembre/2018].

[36] Google, “Gson.” https://github.com/google/gson. [Online; último acceso: Noviembre/2018].

[37] Square Inc., “Okhttp.” http://square.github.io/okhttp/. [Online; último acceso: Noviembre/2018].

[38] Apple Inc, “Getting Started with iOS App Development.”

[39] Apple Inc, “Enrollment.”

[40] Rojas Lara Alex, Valentini Mac Adden Nicolás, “Samplers 2.” http://sedici.unlp.edu.ar/handle/10915/118539.

Índice de figuras

3.1. Atributos y Métodos de la clase Persona 21
3.2. Método abstracto en clase Persona 21

4.1. Ciclo de vida de una Activity 35
4.2. Ciclo de vida de un Fragment 37
4.3. Dos versiones de la misma pantalla en diferentes tamaños de pantalla 38

5.1. Capturas de pantalla de la app de ejemplo generada por Samplers 48
5.2. Diagrama de clases del core del framework 50
5.3. Ejemplo de cómo se guarda una muestra. 55
5.4. Ejemplo de la Activity principal. 57
5.5. Ejemplo de como se visualiza SamplesListActivity. 58
5.6. Ejemplo de como se visualiza SamplesListActivity. 59

6.1. Ejemplo de como se muestra la ayuda. 72
6.2. Ejemplo de un PhotoStep en ejecución. A la izquierda al momento de tomar la foto, y a la derecha al momento de mostrar la vista preliminar. 79
6.3. SoundRecordStep en ejecución. 80
6.4. InformationStep en ejecución. 82
6.5. SelectOneStep en ejecución. 84
6.6. MultipleSelectStep en ejecución. 86
6.7. LocationStep en ejecución. 87
6.8. RouteStep en ejecución. 89
6.9. InsertTextStep en ejecución. 91
6.10. InsertDateStep (izquierda) e InsertTimeStep (derecha) en ejecución. 93

7.1. Capturas de pantalla de la app AppEAR 97

7.2. Grafo direccional que representa el Workflow de AppEAR . . 99
7.3. Capturas de pantalla de la app AppEAR generada por Samplers106
7.4. Izquierda: tipo de ambiente y nombre en una sola pantalla (AppEAR). Derecha: tipo de ambiente y nombre en dos pantallas (Samplers) 107
7.5. Izquierda: selección con imágenes (AppEAR). Derecha: selección simple (Samplers) 108
7.6. Izquierda: pantalla principal con imágenes (AppEAR). Derecha: pantalla principal sin imágenes (Samplers) 109
7.7. Ejemplo de pantalla especial (AppEAR) 110
7.8. Izquierda: mapa con problemas de visualización (AppEAR). Derecha: mapa que se muestra correctamente (Samplers) . . . 111