UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

Titulo: Samplers 2
Autores: Rojas Lara, Alex – Valentini Mac Adden, Nicolás
Director: Dr. Torres, Diego
Codirector:
Asesor profesional: Dr. Firmenich, Sergio
Carrera: Licenciatura en Sistemas – Licenciatura en Informática

Resumen

Esta es una tesina que se encuentra estrictamente relacionada con los conceptos de ciencia abierta y ciencia ciudadana y busca colaborar de algún modo con estos movimientos.
La forma de colaboración es mediante la implementación de un sistema que permita a los científicos generar de manera uniforme una secuencia de pasos para la toma de muestras que sean necesarias para cualquier proyecto. Por otra parte, también permite a ciudadanos que quieran participar de estos proyectos recolectar muestras siguiendo la secuencia de pasos que los científicos hayan definido previamente.

Se buscó solucionar la falta de aplicaciones que se enfoquen en la recolección de datos y que a la vez se apoyen en tecnología móvil. Si bien existen aplicaciones similares, estas son más de propósito general, por lo que Muestre.AR vendría a suplir esta carencia.
La forma de resolverlo fue mediante la creación de una aplicación web centralizada que es donde los científicos van a poder crear o editar sus proyectos y una única aplicación móvil que es donde los ciudadanos van a poder participar de los proyectos creados previamente por los científicos en la aplicación web y van a poder realizar la recolección de muestras.

Palabras Claves

- Ciencia ciudadana
- Ciencia abierta
- Método científico
- Muestre.Ar
- Samplers
- Workflow
- Dashboard
- Android
- Django
- API

Conclusiones

Se propuso una aplicación web a través de la cual los científicos puedan crear una secuencia de pasos específicos y personalizados, para que luego los usuarios ciudadanos puedan colaborar con ellos recolectando una muestra de forma facil y sencilla. Esto se vio alcanzado con la implementacion de 3 aplicaciones comunicadas entre si para poder, por un lado, facilitar a los científicos crear proyectos, diseñar la recolección de una muestra para estos y analizar los resultados y por el otro, brindar un medio atractivo e intuitivo por el cual un usuario recolecte una muestra.
También podemos afirmar que Samplers 2 es una proyecto de ciencia abierta ya que tanto el código fuente y los datos recolectados son accesibles y distribuidas de forma libre y gratuita para toda la sociedad.

Trabajos Realizados

Desarrollo de una aplicacion Android, llamada Muestre.AR móvil utilizada para la toma de las muestras por parte de los usuarios ciudadanos.
Intregración de el framework Samplers en la aplicación móvil.
Desarrollo de una aplicación web como herramienta de administración de proyectos para los usuarios científicos.
Implementación de una aplicación servidor, encargada de comunicar las antes nombradas.
Investigación de librerias para gráficos.
Analisis de las APIs para realizar autenticación compartida entre distintas aplicaciones.
Exposición en CIACIAR 18.
Implementación con Samplers 2 de una aplicación destinada a la recolección de muestras llamada tiburones.

Trabajos Futuros

Una mejora a realizarse sobre la aplicación web es la creación de un Foro, donde los usuarios científicos puedan iniciar un debate sobre un proyecto. Adicionalmente, se podrían realizar gráficos con la información recolectada en las muestras.
Desde el lado de Muestre.AR móvil, se podría crear una sección con la información de cada usuario cuando este inició sesión. Otra mejora para la aplicación móvil podría ser la utilización de la pantalla principal como portal de información. También pensamos que para incentivar al usuario a recolectar muestras, la aplicación móvil podria tener algún componente lúdico y así recompensar a los usuarios para que utilicen la aplicación.
Por último, pensamos que se podría realizar una aplicación para los sistemas iOS.

Fecha de la presentación: 12 de Noviembre de 2019

Samplers 2

Nicolás Valentini y Alex Rojas Lara

11 de noviembre de 2019

Resumen

Esta es una tesina que se encuentra estrictamente relacionada con los conceptos de ciencia abierta y ciencia ciudadana y busca colaborar de algún modo con estos movimientos. La forma de colaboración es mediante la implementación de un sistema que permita a los científicos generar de manera uniforme una secuencia de pasos para la toma de muestras que sean necesarias para cualquier proyecto. Por otra parte, también permite a ciudadanos que quieran participar de estos proyectos recolectar muestras siguiendo la secuencia de pasos que los científicos hayan definido previamente.

Lo que se buscó resolver en esta tesina fue solventar las limitaciones de otro proyecto llamado Samplers. Samplers, es un proyecto que básicamente genera una aplicación Android por cada tipo de muestra que un científico profesional quiere realizar y luego los científicos ciudadanos tenían la posibilidad de descargarla. Detectamos que este último es limitado porque por cada edición o creación que el científico realizaba sobre los pasos que formaban parte de la muestra, se generaba una aplicación móvil distinta, resultando poco práctico para todos los usuarios.

Otro aspecto que se buscó solucionar fue la falta de aplicaciones que se enfoquen en la recolección de datos y que a la vez se apoyen en tecnología móvil. Si bien existen aplicaciones similares, estas son más de propósito general, por lo que Muestre.AR vendría a suplir esta carencia. La forma de resolverlo fue mediante la creación de una aplicación web centralizada que es donde los científicos van a poder crear o editar sus proyectos y una única aplicación móvil que es donde los ciudadanos van a poder participar de los proyectos creados previamente por los científicos en la aplicación web y van a poder realizar la recolección de muestras. Por lo que esta tesina resuelve el problema de las multi aplicaciones Android convirtiéndolo en un proyecto mucho más escalable.

Muestre.AR es el inicio del que creemos un proyecto que va a aportar mucho a la causa de la ciencia ciudadana. Si bien el alcance de esta tesina se acotó al flujo básico, creemos que puede ser expandido con muchas más funcionalidades que en esta tesina no se desarrollaron.

Índice general

Capítulo 1

Introducción

1.1. Motivación

Al hablar de ciencia abierta, nos estamos refiriendo a generar conocimiento científico el cual se pueda acceder de manera colaborativa y abierta, donde las investigaciones, metodologías y datos son distribuidos por todos los ciudadanos de manera libre y gratuita [1]. Esta forma de hacer ciencia tiene algunos beneficios, como aumentar la productividad científica y hacerla más eficiente, posibilitar que más actores se involucren con la democratización del conocimiento y fortalecer la relación ciencia-sociedad al abrirse a la comunidad en un intercambio sistemático, dando respuesta a las demandas sociales. La ciencia ciudadana, por su parte, es un concepto relacionado al de ciencia abierta y puede definirse como una forma de investigación colaborativa donde se involucran miembros del público general en proyectos de investigación científica para abordar problemas del mundo real[2]. La ciencia ciudadana está relacionada con programas que emplean monitoreo voluntario para el manejo de recursos naturales [3], y a menudo se emplea como una forma de educación científica informal o divulgación para promover la comprensión pública de la ciencia [4]. La ciencia ciudadana puede trabajar a escala masiva, generando una alta calidad de datos que conducen a resultados científicos fiables y válidos, así como ideas inesperadas e innovaciones[5]. Los entornos virtuales de contribución hacen posible que una audiencia más amplia se involucren en trabajos científicos, como en otros proyectos de contenido abierto. Un número y variedad cada vez mayor de proyectos de ciencia ciudadana están aprovechando las posibilidades de la tecnología para avanzar en la investigación científica [6]. Las formas de participación generalmente implican aportar datos de acuerdo con un protocolo

establecido o completar tareas de reconocimiento estructurado, clasificación o resolución de problemas que dependen de las competencias humanas [7]. Esta tesina está enfocada en ofrecer a la comunidad un entorno virtual en el cual se permite a científicos profesionales definir un protocolo de muestra, y que los usuarios ciudadanos pueden recolectar datos a través de una aplicación móvil. Muestre.AR, es un proyecto de ciencia ciudadana, centrado en la recolección de datos, lo cual quiere decir que se centra en los lineamientos de un protocolo de investigación científica académica que requiere la recolección de datos del medio físico. En este tipo de proyectos las actividades que realizan los usuarios ciudadanos se centran en la recolección de muestras de datos en el medio físico[8].

Para que la toma de estos datos sean todas comparables entre sí se define un protocolo de investigación científica académica encuadrado en el método científico.

1.2. Método científico

Según Stephen S. Carey [9], podríamos definir al método científico como un proceso de tres pasos a través del cual científicos investigan eventos de la naturaleza. Comienza observando algún aspecto de la naturaleza. Si se observa algo que no se comprende bien, se trata de comprender por qué sucede y luego se encuentra alguna forma de probar esas especulaciones. Este método está compuesto por 3 grandes etapas que se pueden definir como:

Observación

Se busca conocer los hechos. Puede ofrecer pistas sobre lo que podría implicar el evento observado.

Explicación

Se intenta demostrar cómo o por qué el evento en cuestión se ha convertido en el caso. Cuando algo no se entiende bien, su explicación no estará clara.

Explicaciones de prueba

Se lleva a cabo un experimento para determinar si algo que debiera pasar bajo las circunstancias apropiadas con la explicación correcta sucedió. Si obtenemos los resultados que esperábamos, tenemos buenas razones para creer que nuestra explicación es correcta. Si no los conseguimos, tenemos una razón inicial para sospechar que podemos estar equivocados o, al menos, que podemos necesitar modificar la explicación propuesta.[9]

Algunas características del método científico son:

Fáctico, es decir que se basa solo en lo que los hechos reflejan.

Es **objetivo**, a través del método científico se genera realizar juicios de valor atendiendo a los hechos y la lógica, y no a los propios sentimientos o sensaciones.

Es **riguroso y de orden lógico**, no se trabaja de cualquier manera en el método científico, no se pueden omitir pasos, tampoco alterar el orden lógico, ya que no se arribaría a ninguna conclusión válida en ese caso.

Requiere **experimentación controlada y sistemática** , plantea la etapa de experimentación de manera muy cuidadosa, tratando de que esta resulta reproducible y de que ningún factor se convierta en una amenaza para la robustez del experimento, aún cuando se acepta que existe siempre el denominado error experimental.

Exige permanente **análisis y síntesis**, mediante el análisis se identifican cada una de las partes que integran una realidad; a través de la síntesis todas las partes identificadas se integran en un todo más abarcador, que contribuye al entendimiento de un fenómeno.

Es **autocorrectivo**, las conclusiones pueden modificar o rechazar a partir de inconsistencias en el método. [10]

1.3. Objetivos, estrategia y contribución

1.3.1. Objetivos generales

Para cumplir con varias de las características que requiere un método científico antes mencionadas, proponemos Muestre.AR, siendo este un conjunto de aplicaciones que permiten definir a los usuarios científicos una secuencia de pasos específicos y personalizados para que los usuarios ciudadanos puedan recolectar una muestra. Estas muestras del mundo físico son obtenidas través de una aplicación móvil y luego los resultados son enviados y almacenados de forma centralizada y digital.

1.3.2. Objetivos específicos

- Desarrollar una aplicacion Android, llamada Muestre.AR móvil utilizada para la toma de las muestras por parte de los usuarios ciudadanos.
- Desarrollar una aplicación web como herramienta de administración de proyectos para los usuarios científicos.
- Implementar una aplicación servidor, encargada de comunicar las antes nombradas.

1.3.3. Metodología

Se llevó a cabo una reunión semanal con todos los miembros del proyecto, en las que cada uno de los miembros expresó sus ideas, se debatió acerca de estas y se trató de elegir la mejor alternativa para llegar a un acuerdo. Las primeras reuniones se concentraron en el análisis de las aplicaciones de ciencia ciudadana, específicamente en las que se centran en la recolección de muestras, evaluando ventajas y desventajas de cada una. Tambien se trabajó en el diseño (creación de mock ups), planeación de la aplicación y nición de requerimientos. Una vez finalizada esta etapa, se procedió a la codificación de la aplicación. Uno de los objetivos de utilizar esta metodología fue para que la aplicación se encuentre en funcionamiento en todo momento. Con cada nueva funcionalidad que se realizó, las que fueron agregadas anteriormente no fueron afectadas, logrando de esta manera que la aplicación pudo ser accesible al público en todo momento. Adicionalmente, se realizaron distintos tipos de prueba, como por ejemplo en el CIACIAR '18 que los usuarios utilizaban Muestre.AR y definian sus propios protocolos de muestras. Otra de las pruebas realizadas sobre Muestre.AR fue la simulación de una aplicación destinada a la recolección de muestras en funcionamiento.

Las contribuciones del presente trabajo son:

- Brinda a la sociedada un herramienta para la construcción y publicación de proyectos de ciencia ciudadana de recolección.
- Provee una aplicación móvil Android, para recolección de muestras del mundo físico, publicadas por la herramienta antes mencionada.
- Contribuye a que la sociedad se involucre en la ciencia ciudadana de una forma rápida y sencilla.
- Contribuye con los científicos profesionales brindándoles una herramienta para crear sus propios proyecto, diagramar y configurar un protocolo de muestra para este.
- Ayuda al científico a obtener mayor número de colaboradores en sus proyectos.
- Permite al usuario entretenerse y al mismo tiempo obtener conocimiento acerca de los proyectos de ciencia actuales.

1.4. Organización de la tesina

El Capítulo 2 corresponde al marco conceptual y temas relacionados, en el cual se describen los conceptos a tratar en la tesina. En la primer subsección del capítulo se define el concepto de ciencia ciudadana, se describe su historia y actualidad, se listan y describen las problemáticas que ataca y el valor aportado, se presentan 4 tipologías que clasifican a los diferentes proyectos de ciencia ciudadana, y finalmente se realiza una descripción de los voluntarios. La siguiente subsección describe los proyectos basados en la recolección de datos.

En el Capítulo 3 se desarrollan las diferentes decisiones en términos de implementación, junto con su justificación. También se profundiza en las tecnologias y metodologias utilizadas, los desafíos que afrontamos y sus resoluciones.

En el Capítulo 4, se compara a Muestre.AR con otros proyectos de ciencia ciudadana, destinados a la recolección de datos, donde se destinará una sección a cada uno de estos proyectos. En el final del mismo, se encuentra una tabla comparativa a modo de resumen.

En el Capítulo 5 se comparte experiencias y eventos en los que presentamos Muestre.AR.

En el Capítulo 6, se habla sobre las conclusiones de la tesina y mejoras futuras.

Capítulo 2

Temas relacionados y marco teórico

En el transcurso de este capítulo abordaremos conceptos relacionados a nuestra tesina que se referenciarán a lo largo de esta.

2.1. Ciencia Abierta

2.1.1. Definición

La ciencia abierta es un movimiento cuyo objetivo consiste en que todo el conjunto de metodologías e inviestigaciones cientfícas y datos obtenidas a partir de estas, sean accesibles y distribuidas de forma libre y gratuita para toda la sociedad. Esto hace que la contribución a la ciencia no sea exclusiva sólo para las personas que tengan conocimiento científico, sino también para aquellas que no lo tienen. Lo que caracteriza a la ciencia abierta es un acceso más democrático y equitativo a la información. Esto quiere decir que la información pueda ser accesible por cualquier persona sin importar su origen o procedencia y condición social.[11]

2.1.2. Categorías

La ciencia abierta se pueden clasificar de acuerdo al nivel de accesibilidad y disponibilidad del conocimiento científico. Estas pueden ser:

- Open Research: el acceso a la información es libre y gratuito. Se distribuye todo el proceso de investigación científico, tanto del propio como derivados, y se posibilita la colaboración entre profesionales.

- Open Access: Es similar a la anterior, pero con la diferencia de que el contenido es distribuido a través de Internet, y no se acepta la colaboración entre profesionales. Esto quiere decir que un científico profesional no puede inmiscuirse en un proyecto de otro científico profesional.
- Open Notebook: es la categoría más radical de la ciencia abierta. En esta categoría se publican absolutamente todos los casos del proceso de investigación. Aquellos que tuvieron resultados positivos como no positivos, los casos que fallaron, etc. Algunos científicos son reticientes a este tipo de ciencia.
- CrowdScience (ciencia ciudadana): similar a open research u open access, con la diferencia de que pueden participar ciudadanos u otros profesionales ajenos a la institución donde se lleva a cabo la investigación.

2.2. Ciencia Ciudadana

2.2.1. Definición

Es la participación del público en general en actividades de investigación científica en las que los ciudadanos contribuyen activamente, ya sea con su esfuerzo intelectual o con el conocimiento de su entorno o aportando sus propias herramientas y recursos.[12] Esta forma de hacer ciencia, beneficia a ambas partes de los participantes. Por un lado, los científicos profesionales se ven beneficiados ya que la contribución de los ciudadanos les permite a los científicos obtener muestras de una forma más eficiente, variada y rápida. Esto quiere decir que al ser más personas las que contribuyen a este fin, la recolección de muestras se lograría en menos tiempo. Y por el otro, los ciudadanos se ven beneficiados ya que el acceso a la información de las investigaciones científicas no sólo se limita a científicos con conocimientos profesionales, sino a cualquier persona que quiera ser parte de la investigación.

2.2.2. Historia

El concepto de ciencia ciudadana no es nuevo, sino que data desde hace siglos, incluso siendo mas antiguo que la profesionalización de la ciencia en sí. Cientıficos amateur han estudiado el mundo durante la mayor parte de la historia registrada. En un principio, los científicos buscaban ejercer otras

profesiones además del propio, ya que la labor científica nos les bastaba para poder solventarse. Es por esto que, una gran parte de la investigación científica era llevada a cabo por amateurs. La diferencia entre la ciencia ciudadana actual y la ejercida anteriormente radica en el alcance que la primera tiene: cualquier persona puede acceder a ella, y no unos pocos privilegiados. Algunos casos en los que se aplicó la ciencia ciudadana en la antigüedad son los siguientes:

- Un ornitólogo estadounidense que estaba interesado en las migraciones de aves, fue quien organizó un grupo de voluntarios y les pidió que recolectaran información sobre cuándo llegaban aves a determinado lugar, en qué momento eran más abundantes, y dónde se iban en la primavera y en el otoño. Comenzó en el año 1881 y se disol- vió en el año 1970, con un total de 6 millones de registros de migración de aves.
- Otro ejemplo, también relacionado con el estudio de las aves, estaba orientado a tratar de resolver la problemática de este momento: grandes cantidades de aves colisionaban contra faros, causándoles la muerte a las aves. Debido a esto, se comenzó un programa voluntario que involucraba contactar a todos los fareros sobre la costa del caribe, con una simple pregunta: "¿Podrıa contar la cantidad de aves que chocan contra el faro?". Sin embargo, la falta de un protocolo formal, junto con la dificultad de poder entrenar a los voluntarios a identificar correctamente a los especímenes, hizo que la calidad de los datos obtenidos sea irregular.
- El siguiente ejemplo es uno de los primeros casos de ciencia ciudadana con la particularidad de que sigue en vigencia: se trata de un proyecto que comenzó en diciembre de 1900, creado por Frank Chapman, en el cual un grupo de voluntarios forman varios grupos y realizan un recuento de aves coordinado en un círculo de 24 km de diámetro durante todo un día en época navideña.[13]

2.2.3. Ejes esenciales de la ciencia ciudadana

- La colaboración del ciudadano científico Los ciudadanos científicos son aquellos participantes que colaboran en un proyecto de investigación científica y juegan un rol muy importante ya que son los que aportan la recolección de datos, generalmente obtenidos del medio ambiente.
- Los proyectos son colaborativos El único requisito para que los ciudadanos puedan participar es que sean observadores y entusiastas a la

hora de recolectar datos, ya que cualquier ciudadano puede colaborar en las investigaciones, sin importar raza, sexo o edad.

- Se debe garantizar la retroalimentación La ciencia ciudadana depende de una fuerte interacción entre los ciudadanos los cuales aportan su tiempo y sus capacidades para recolectar datos a su alrededor; y los científicos quienes aportan sus conocimientos y enseñan métodos tanto para análisis como para el procesamiento de esos datos[14]

2.2.4. Clasificaciones

Las diferentes clasificaciones de ciencia ciudadana agrupan y describen a los proyectos de ciencia ciudadana bajo distintos criterios. Cada clasificación propuesta por los diferentes autores provee una definición más profunda del concepto de ciencia ciudadana, ya que esta tiene diferentes matices porque depende del ámbito o del tipo de participación de los ciudadanos, abarcando muchas áreas de la ciencia e involucrando a los ciudadanos de diferentes maneras.

1ra Clasificacion: Se concentra en los objetivos del proyecto, características organizacionales y uso de tecnologias. Por Andrea Wiggins y Kevin Crowston[15]

- Acción: en este tipo de proyectos, el rol de los ciudadanos es más preponderante que el de los científicos, ya que estos últimos solo actúan como colaboradores o asesores en las investigaciones, siendo los ciudadanos los impulsores de estas. En este tipo de proyectos las investigaciones se realizan en asuntos del medio ambiente local para los cuales actividades científicas están fuertemente ligadas al mundo físico.

- Conservación: estos proyectos se basan en investigaciones en el área de la ecología, relacionados a la administración de recursos naturales. La actividad de los participantes es la de recolección de datos.

- Recolección: también llamado de investigación, se basan en proyectos donde estos se nutren de datos provenientes de la recoleccion de un medio físico, con participación activa de ciudadanos no científicos. Suelen ser utilizados como material de estudio, y su alcance puede ser regional o internacional. Se caracterizan por tener un grado alto de participación de los ciudadanos.

- Virtual: la diferencia que existe con respecto de los proyectos de investigación de recolección tradicionales es que el aporte de datos se hace

a través del uso de la tecnología, sin elementos físicos de ningún tipo y, en su lugar, con elementos virtualizados como fotos, videos, capturas, etc. Suelen ser proyectos relacionados al área de la astronomía, paleontología y proteómica (estudio de las proteínas).

- Educación: el principal foco de estos proyectos consiste en la educación y en la difusión. Promueve el aprendizaje continuo, y en lugar de buscar generar resultados científicos válidos, lo que busca es la divulgación, el aprendizaje y el desarrollo de habilidades de investigación.

Se podría categorizar a Muestre.AR como un proyecto de recolección , ya que uno de los principales objetivos se enfoca en la obtención de datos del medio físico. No obstante, como dijimos anteriormente, también se enfoca en el análisis de la información obtenida.

2da Clasificacion: Esta clasificación la realiza el autor Candie C. Wilderman[16] y se caracterizan, al menos en parte, a partir de las respuestas a 5 preguntas:

- ¿Quién es el que define el problema? ó ¿Quién define la agenda de trabajo?
- ¿Quién es el que diseña el estudio?
- ¿Quién es el que recolecta las muestras?
- ¿Quién es el que analiza las muestras?
- ¿Quién es el que interpreta los datos?

A continuación, los 3 modelos definidos:

- Modelo de asesoramiento a la comunidad (Community Con- sulting Model): En este esquema, el conjunto de personas no científicas (entendidas como comunidad) son las que definen el problema, y los científicos profesionales son los que llevan aa cabo el estudio En este modelo, los científicos profesionales cumplen un rol de soporte a la comunidad. En otras palabras; la primera de las 5 preguntas se responde con “la comunidad”, mientras que el resto de las 4 (el resto del trabajo a realizar) se responde con “los científicos”.
- Modelo de comunidad como trabajadores (Community Wor- kers Model): Es el modelo opuesto al descripto anteriormente, ya que en este caso los científicos profesionales son los que plantean la investigación

y diseñan el estudio, y luego la "comunidad" es la encargada de recolectar las muestras, luego los científicos son los que sacan las conclusiones pertinentes. La tercer pregunta se responde con "comunidad", y la cuarta, con "comunidad" y "científicos". El resto del trabajo es también hecho por los científicos.

- Modelo de investigación participatoria basado en una comunidad (Community-based, Participatory Research Model). En este esquema, la comunidad es la encargada de definir el problema, arma el plan de estudio, la encargada de recolectar las muestras, así como también se encarga de su posterior análisis e interpretación de los datos. En este modelo, todas las preguntas se responden con "la comunidad".

Muestre.AR podría clasificarse como "Modelo de comunidad como trabajadores" ya que los científicos profesionales definen y diseñan el estudio y la comunidad recolecta las muestras, que finalmente son analizadas por los mismos científicos profesionales. Si bien este autor deja entrever que los científicos ciudadanos son algo así como "mano de obra barata", nosotros no compartimos en absoluto este pensamiento, sino que creemos que los científicos ciudadanos tienen igual o mayor importancia que los científicos profesionales, ya que Muestre.AR no podría funcionar sin su colaboración.

3ra Clasificacion: Clasifica a partir del grado de participación del público, y del grado de control que tienen los participantes sobre los diferentes pasos en la investigación, todo desde el punto de vista de la educación. Por Rick Bonney et al [17]

Se dividen los proyectos en tres grandes categorías:

- Contributivos: es la categoría típica de ciencia ciudadana, donde los científicos profesionales son los ideólogos de la investigación, y los ciudadanos son los que llevan a cabo la recolección de datos. Los científicos definen un esquema de preguntas y un protocolo a seguir, y los ciudadanos son los que aportan datos a partir de sus muestras. Los ciudadanos pueden participar en el análisis de datos mediante herramientas de visualización online, pero esta participación no resulta clave para la investigación. La mayoría de los proyectos de ciencia ciudadana encajan en esta clasificación.

- Colaborativos: en esta clasificación los ciudadanos comunes aparte de desempeñarse en la función de recolectores de datos, también pueden participar en el diseño del proyecto, así como también en el análisis e interpretación de los datos. También pueden presentar los resultados

a otros miembros del público y/o a otros científicos. Los ciudadanos comunes tienen un rol más preponderante en lo que respecta a la planificación del proyecto.

- Co-creados: diseñados por científicos y por miembros del público general en conjunto, en donde por lo menos uno de los ciudadanos no científicos se encuentra con participación activa en lo que respecta a los pasos del proceso científico. El impulsor de la investigación científica es el ciudadano. a partir de una necesidad o problemática observada (por ejemplo, la contaminación) y luego con la ayuda de los científicos se le da curso a la investigación. Constantemente se alienta al público a que colabore con la causa.

Muestre.AR podría clasificarse como un proyecto contributivo ya que los miembros del público recolectan datos relevantes siguiendo un protocolo definido por los científicos. Y sólo los científicos pueden diseñar el proceso de investigación.

2.3. Proyectos de recolección de muestras

2.3.1. Definición

Estos proyectos se basan en la recoleccion de muestras de datos a través del medio ambiente. La mayoría de estos proyectos tienen una finalidad educativa, más allá que algunas veces ese no sea el objetivo principal. Las acciones que se realizan sobre las investigaciones favorecen el aprendizaje continuo. Estos proyectos suelen generar una gran convocatoria y escalas de participación muy grandes. El modo de procedimiento de este tipo de proyectos es el siguiente:

- El ciuadano científico reliza una muestra del medio físico (que puede ser del jardín de su casa, por ejemplo) enviando los datos de la muestra en un formato, generalmente virtual, como fotos, mediciones, etc.

- El científico con conocimiento profesional toma los valores de recolección para continuar su investigación, a veces, por medio de un proyecto de ciencia ciudadana virtual o analizando la información con una forma más tradicional. Los proyectos más conocidos de este tipo ocurren en el marco de las ciencias de la biología. Sin embargo, también aparecen proyectos en materia de meteorología y climatología.

- El científico profeiosnal una vez obtenido los datos, realiza un análisis e interpretación de estos datos, y así saca sus conclusiones, los cuales puede compartir con otros científicos.

2.3.2. Ejemplos

El proyecto "Pluviómetros ciudadanos" es un proyecto de ciencia ciudadana de origen chileno que convoca a ciudadanos científicos para que instalen un pluviómetro en sus casas, regularmente tomen muestras de las precipitaciones y las carguen a través de un formulario Web que se encuentra en la página del proyecto.
El agua es un recurso vital para el desarrollo de un país, por lo que toda información que pueda ser recopilada es fundamental para una correcta administración del recurso. Es por ello que cada dato que se ingrese es de mucho valor, ya que permite conocer el comportamiento de las lluvias en el territorio de Chile, así como permitir futuros análisis gracias a la creación de una base de datos, producto de la información ingresada.
AppEAR (http://www.app-ear.com.ar), 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 preguntas, y otorga la posibilidad de tomar fotografías y registrar la posición geográfica por medio del GPS.
Otro reconocido proyecto en Argentina de ciencia ciudadana es el gestionado por la Asociación Ornitológica del Plata (Aves Argentina): eBird-Argentina, el cual recibe observaciones de aves de cualquier persona y en cualquier parte del mundo.

2.3.3. Soporte de las TIC

La tecnología jugó un papel muy importante en el desarrollo de la ciencia ciudadana. Como plantea Newman: **"En los últimos 20 años, varios desarrollos en informática -especialmente en aplicaciones web de interfaces gráficas con los usuarios, datos de sistemas de información geográfica que ahora pueden ser utilizados mediante los smartphones y otros dispositivos móviles– han sido vitales para la**

emergencia de la ciencia ciudadana".

La aparición de la tecnología de los smartphones y tablets no sólo provocó un beneficio en el ámbito de la ciencia ciudadana en lo inmediato sino que es considerada una tendencia en el futuro, en el corto y mediano plazo. Esta proliferación de la ciencia ciudadana con estos dispositivos se debe a la accesibilidad que estos brindan, y a la capacidad de estar presente en todas partes todo el tiempo.
La ciencia ciudadana aprovecha las siguientes características de los dispositivos anteriormente mencionados para potenciar sus investigaciones

- Portabilidad: la ventaja de poder transportarlos con uno mismo a cualquier lado es que las investigaciones se pueden realizar fuera de un laboratorio o fuera de cualquier ámbito tradicional.
- Conectividad: los científicos pueden mantenerse en contacto con otros científicos de otras partes del mundo.
- Interactividad: fomenta un entorno colaborativo entre los distintos participantes (por ejemplo: entre científicos y ciudadanos no científicos).
- Sensibilidad al contexto: este punto está relacionado con el de portabilidad, ya que un voluntario podría recolectar muestras independientemente del espacio fisico donde se encuentre y del contexto en donde se halle.

El impacto que tuvo el avance tecnológico en la ciencia ciudadana dio como resultado una facilitación en la compartición de la información científica y un mejor trabajo en conjunto entre los participantes, ya que la colaboración en la resolución de problemas se volvió más ágil y fluída.
"La tendencia en el futuro en proyectos de ciencia ciudadana con soporte de las TICS es el desarrollo de juegos y la incorporación de elementos lúdicos, lo que va a fomentar más aún la participación de los ciudadanos en estos proyectos". [18]

2.4. Protocolo de muestra

Es una secuencia de pasos definidas por el científico profesional que definen cómo se debe obtener una muestra para su posterior estudio. En ciertas situaciones como la toma, clasificación y etiquetado de muestras, el científico realiza un gran número de tareas simples, repetitivas, que no pueden ser automatizadas y que podrían ser ejecutadas por personas sin formación en la

materia si se las entrena y asiste con herramientas. El concepto de protocolo de muestra se ve implementado generalmente en los proyectos de recolección de muestras (definido anteriormente en 2.3.1).

2.5. Aplicaciones móviles

Definición

Una aplicación móvil o también conocida con los nombres de aplicación o sencillamente app (acrónimo de application), es básicamente un programa informático diseñado para que pueda ser ejecutado en smartphones, tablets o cualquier otro dispositivo móvil. Estas aplicaciones, según la finalidad que le quiera dar el usuario, pueden ser de tipo: profesional, de entretenimiento, educativas, de acceso a servicios, redes sociales, etc., facilitando las gestiones o actividades a desarrollar.

Ventajas

Al ser aplicaciones que fueron pensadas para ser ejecutadas en los dispositivos móviles, se puede hacer un uso mejor de los recursos que brinda el smartphone, aportando una serie de ventajas tales como [19]:

- La inmediatez y la sencillez con la que se accede a la información o a determinados servicios.
- Almacenamiento de manera segura de sus datos personales, lo que permite a los usuarios ahorrar tiempo y acceder de una manera rápida a sus preferencias
- Un sistema de notificaciones push que le informa al usuario sobre cualquier novedad o actualización.
- Mejorar la capacidad de conectividad y disponibilidad de servicios y productos

¿Qué necesito para descargar y usar una aplicación?

Para poder descargarse una aplicación y hacer uso de ella, solamente necesita un dispositivo móvil con acceso a Internet. Generalmente las aplicaciones se descargan de un repositorio de aplicaciones. Estos varían dependiendo del sistema operativo de cada dispositivo. Esto quiere decir que el sistema operativo Android va a tener su propia tienda de aplicaciones (en este caso, “Play

Store") y al mismo tiempo, IOS tiene uno propio (llamado "App Store"). Es importante remarcar esto, ya que una aplicación puede ser distribuida de forma exclusiva para dispositivos con sistema operativo Android, por lo que si alguien tiene un dispositivo que tenga IOS no va a poder descargar esta aplicación; y también sucede a la inversa.

¿A qué tipo de datos pueden acceder las aplicaciones?

Una aplicación puede necesitar utilizar un recurso del dispositivo o acceder a la información almacenada para su funcionamiento. Cuando esto ocurre, se le solicita una autorización al usuario para que este lo apruebe. Una aplicación puede acceder a cualquier tipo de dato, enumerado a continuación:

- Lista de contactos de teléfono y de email.
- Registro de llamadas.
- Datos transmitidos por Internet.
- Información de su calendario.
- Datos de localización del dispositivo.
- Al código de identificación exclusivo de su aparato.
- A información que indica la manera en que usted usa la aplicación propiamente dicha.

Algunas aplicaciones solamente pueden acceder a los datos necesarios para su funcionamiento. Otras pueden acceder a datos que no están relacionados con el propósito de la aplicación.[20]

2.5.1. Android

Definición

Android es el nombre de un sistema operativo diseñado por Google para ser ejecutado en dispositivos móviles como smarthpones o tablets basados en el S.O. de Linux. En un principio fue diseñado para estos dispositivos, aunque hoy en día existen otros dispositivos que cuentan con este sistema operativo, como son los Smart Tvs o incluso Smart Watches.

Características

La característica más importante es que Android es un sistema operativo

Open Source ("código abierto"). Esto quiere decir que el código fuente se encuentra público y accesible para todos. Por lo que cualquier programador puede sentirse libre de descargar el código y modificarlo a su antojo. Incluso puede adaptar el sistema operativo para que funcione en cualquier dispositivo. Esta es una de las características por los cuales este sistema operativo se ha vuelto tan popular y tan utilizado. Otras características que posee:

- Núcleo basado en el Kernel de Linux.
- Adaptable a muchas pantallas y resoluciones.
- Utiliza SQLite para el almacenamiento de datos.
- Personalización. Android puede personalizarse ampliamente. Esto está relacionado con la característica de que sea open source, ya que si no esto no sería posible. Cada compañía de celular (como por ejemplo: Samsung, Lg, Huawei) le añade una capa de personalización propia.
- Soporte de HTML, HTML5, Adobe Flash Player, etc.
- Incluye un emulador de dispositivos, herramientas para depuración de memoria y análisis del rendimiento del software.

[21]

Fragmentos

Los fragmentos en Android son un componente importante ya que son una clase .java que extienden de la clase Fragment, *la cual es una sección modular de una actividad que tiene su ciclo de vida propio, recibe sus propios eventos de entrada y que puedes agregar o quitar mientras la actividad se esté ejecutando*, y representan a una vista o pantalla de la aplicación. Los Fragmentos son responsables de la lógica de decidir qué datos se desean mostrar y de manejar los distintos eventos que pueden disparar los usuarios mediante la interacción. Archivos con formato XML, ubicados entre los recursos del proyecto, definen las estructuras de las vistas, donde se incluyen los botones, las imagenes o cualquier otro componente que se quiera incluir en esta. Adicionalmente, la tecnología brinda la posibilidad de crear un componente programáticamente, sin la necesidad de incluirlo en el documento donde se define la estructura, pero es recomendable hacerlo ya que a la hora de programar, le brinda al desarrollador una mera noción de la distribución de los componentes. Desde el Fragmento, en el método onCreateView el cual es el único obligatorio a implementar para extender la clase Fragment, es obtenido el archivo con formato XML a través de su identificador. Luego, cada elemento de esta vista es referenciado de la misma forma.

2.6. Samplers

2.6.1. Definición

Samplers es un framework (es decir, un programa para construir aplicaciones) de código libre para construir aplicaciones Android de ciencia ciudadana que utilicen las herramientas que brindan los dispositivos móviles para la recolección de muestras. Está pensada para que investigadores que tienen un proyecto de investigación que requiere recolectar muestras puedan hacer una aplicación de Android y distribuirla para que los usuarios de su aplicación colaboren juntando muestras. Un investigador puede definir el protocolo de recolección de una muestra como una serie de pasos a seguir en un archivo de texto. Samplers se encarga de leer esos pasos y en base a eso crear una aplicación Android para recolectar muestras para el proyecto. Las aplicaciones o app's que se creen con Samplers guiarán al usuario para que respete los pasos de la recolección de la muestra. La muestra puede incluir las respuestas a preguntas hechas al usuario, datos de geolocalización, fotos y audio. Al utilizar las funciones que ofrecen muchos de los dispositivos móviles aplicadas a recolectar muestras para proyectos de ciencia ciudadana permite por un lado que el colaborador utilice una herramienta con la que está familiarizado (como lo es su dispositivo móvil) y por otro lado permite al investigador plantear protocolos de recolección más complejos e incluir archivos anexos a las muestras. Una vez recolectadas las muestras, el usuario puede elegir enviarlas por Internet para que los investigadores del proyecto puedan analizarla. [22]

2.6.2. Alcance

Samplers toma como dato de entrada un archivo de texto con la configuración del protocolo de muestra definido por el cientifico, en un formato (formato JSON) donde era necesario contar con conocimientos informáticos para realizarlo. Y a partir de la secuencia de pasos definidos en este archivo de configuracion, Samplers realizaba una visualización de los mismos para que el usuario vaya completando los pasos y así completar el protocolo de muestra. Una vez hecho esto, mandaba los resultados a un servidor en un formato ZIP. Detectamos que el alcance de Samplers es limitado porque por cada edición o creación que el científico profesional realizaba sobre los pasos que formaban el protocolo de muestra, se generaba una aplicación móvil distinta, resultando poco práctico para todos los usuarios. Además, si un científico ciudadano desea tomar muestras para 3 proyectos distintos, deberán descargarse 3 aplicaciones, siendo también muy difícil poder identi-

ficar al mismo usuario en 2 proyectos distintos. Por otra parte, también sería complejo el análisis de las muestras, ya que si un protocolo de muestra cambia, se genera una aplicación nueva y las muestras de esta nueva aplicación serán distintas a las primeras.

2.7. Django

2.7.1. Definición

Django es un framework de desarrollo Web de código abierto, escrito en Python, que permite construir aplicaciones web de manera fácil y rápida. Es un framework que respeta el patrón de diseño comúnmente llamado MVC (modelo-vista-controlador), aunque para este framework al controlador se lo conoce como vista, y a la vista se la conoce como template. Por lo que las siglas quedarían como MVT. A continuación se explica cada una de las partes:

- M significa "Model"(Modelo): la capa de acceso a la base de datos. Esta capa contiene toda la información sobre los datos: cómo acceder a estos, cómo validarlos, cuál es el comportamiento que tiene, y las relaciones entre los datos. En el archivo models.py contiene esta información.
- T significa "Template"(Plantilla): la capa de presentación. Esta capa contiene las decisiones relacionadas a la presentación: cómo algunas cosas son mostradas sobre una página web u otro tipo de documento. Los templates en Django son los archivos .html.
- V significa "View"(Vista): la capa de la lógica de negocios. Esta capa contiene la lógica que accede al modelo y la delega a la plantilla apropiada: se puede pensar en esto como un puente entre el modelos y las plantillas. En el archivo views.py es donde está contenida esta información.

La siguiente figura 2.1 explica la comunicación entre sus partes:

El objetivo de Django es facilitar el proceso de construir aplicaciones Web que sean complejas. Y hace foco en la reutilización, la conectividad y extensibilidad de componentes, el desarrollo rápido y el principio No te repitas (DRY, del inglés Don't Repeat Yourself). Algunas características del lenguaje:

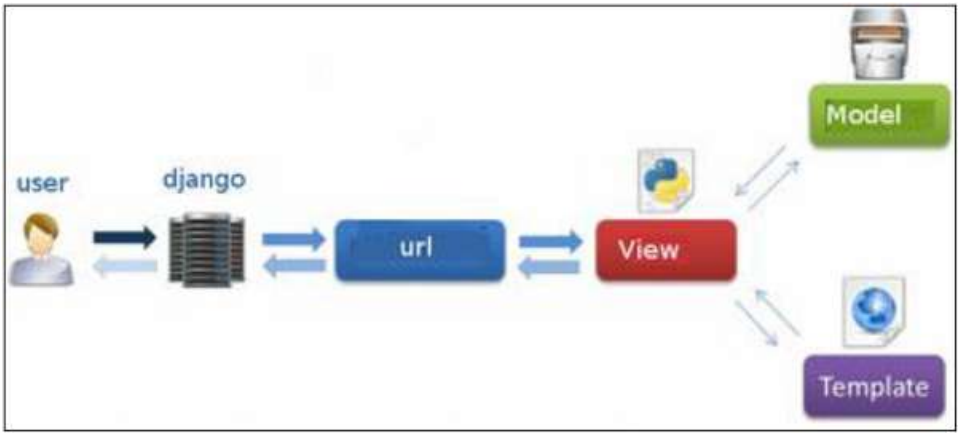

Figura 2.1: Patrón MVT en Django

- Rapidez: Django en su origen fue desarrollado para gestionar artículos de noticias, estas se subian muy rápido, y como los desarrolladores no pudieron estar a ese ritmo decidieron crear algo que sí lo haga, y fue ese motivo por el cual nace Django . Es por eso que ha sido diseñado de manera tal que se le facilite la tarea al desarrollador y en que el proceso de construir aplicaciones web sea rápido.
- Mantenible: Bajo la filosofía Dont dry yourself (traducido al español No te repitas) lo que se evita es la creación de códigos iguales y fomenta la reutilización del mismo, por lo que se logra un código mantenible.
- Panel de Control Admin: Django es el único framework que "por defecto" viene con un sistema de administración para las tablas de tu modelo, donde se podrán realizar operaciones de ABM sin ningún tipo de configuración por parte del desarrollador.
- ORM incorporado: las consultas a la base de datos no están atadas a ningún lenguaje en especial; realiza una abstracción del lenguaje de consulta a la base de datos por lo que después se puede configurar cualquier gestor de base de datos de forma indistinta y las consultas no se verán afectadas.
- Seguro: Django permite protección contra algunas vulnerabilidades de forma predeterminada, incluida la inyección SQL, scripts entre sitios, falsificación de solicitudes entre sitios, etc.

- Dinamismo: hoy en día ya casi no existen aplicaciones web con contenido estático. Tampoco se puede insertar código Python en una página .html porque el navegador no lo sabe interpretar. Es por esto que la forma de que Django proporcione dinamismo en una página web es a través de la incorporación de template tags. Los mismos se insertan en los htmls de la forma " ". En resumen, los template tags de Django nos permiten acceder desde el html a variables de la vista, para que se pueda construir sitios web dinámicos de forma más rápida y fácil.

[23]

Capítulo 3

Implementación

Este capítulo tiene como finalidad explicar el diseño y la implementación de la solución comenzando por las etapas de definición de requerimientos de cada una de las aplicaciones, explicando y fundamentando las decisiones tomadas. También hablaremos de la arquitectura utilizada para realizar la implementación y por último explicaremos en detalle cada uno de los desarrollos realizados para las distintas aplicaciones que componen Muestre.AR.

3.1. Requerimientos

Al realizar una investigación sobre las aplicaciones que existían hasta el momento, como se refleja en la tabla del capítulo 4, se observa que estas comparten varias de las acciones que debe realizar un usuario para recolectar una muestra. Si un usuario quisiera colaborar recolectando muestras para varios proyectos debería descargarse cada una de las aplicaciones de cada proyecto. Esto resulta molesto, incluso más si este debe autenticarse para realizar la recolección.

3.1.1. Usuarios en Muestre.AR

Muestre.AR contará con 2 tipos de usuarios, cada uno con distintos permisos, la cual comentamos a continuación

Usuarios ciudadanos

Como hablamos antes, los voluntarios, llamados de ahora en más *Usuarios ciudadanos*, son personas del público general que participan en proyectos

científicos. No es necesario que tengan algún tipo de entrenamiento, conocimiento o experiencia en el tipo del proyecto que participan, aunque por lo general, quienes colaboran muestran interés por el tema del proyecto al que se unen.

Usuarios profesionales

Por otra parte, los científicos, de ahora en más *Usuarios Científico*, son personas con un conocimiento más académico y que por lo general pertenecen a alguna institución. Estos optan por utilizar la ciencia ciudadana para la recolección de muestras para sus proyectos, cargandolos en Muestre.AR para que los usuarios ciudadanos puedan colaborar con ellos.

En base a estas observaciones, definimos la creación de una aplicación web, a través de la cual un usuario profesional pueda crear sus proyectos de forma personalizada. Estos proyectos serán creados por un usuario autenticado en la aplicación web e inicialmente contarán con un nombre y con una descripción. Una vez que el proyecto esté creado en Muestre.AR web, para poder comenzar a recolectar las muestras para éste, se deberán definir cada una de las etapas de la recolección y la relación entre estas etapas. De ahora en adelante llamaremos **Workflow** al conjunto de etapas o pasos que definen cómo será la recolección de una muestra para un proyecto y la relación entre éstas. Los científicos son quienes definen el protocolo de muestra, es decir las características de cada uno de los pasos para tomar la muestra. De ahora en mas, el término protocolo de muestra y Workflow serán usados indistintamente. Un ejemplo de workflow podría ser la definición de indicar la ubicación a la hora de recolectar la muestra y contestar una pregunta propuesta por quien construye el workflow.

Un workflow puede contener los siguientes tipos de pasos:

- Show Information: este paso se usa para mostrar determinado tipo de información que considere pertinente el científico. No requiere ningún tipo de interacción por parte del ciudadano científico.
- Take a Photo: en este paso se le indica al ciudadano científico que realice una captura con su dispositivo móvil de la muestra.
- Get the GPS Location: en este paso se pide la ubicación del usuario científico. Para eso se le pide al usuario que active el GPS y brinde su posicionamiento
- Select One option: en este paso se le otorga al recolector una pregunta y una lista de opciones. Sólo se puede elegir una opción y dependen-

diendo de la elegida, el workflow puede seguir (o no) una secuencia de pasos distinta. Es por esto que a este paso también se lo llama como de bifurcación o decisión

- Select Multiple options: en este paso se le otorga al recolector una pregunta y una lista de opciones. A diferencia del anterior paso, en este paso se pueden seleccionar varias opciones, y el siguiente paso siempre será el mismo. No se produce ninguna bifurcación

- Insert Text: en este paso el científico es libre de preguntar ya sea una pregunta cerrada o una pregunta abierta, y se le permite configurar que la respuesta del recolector sea de tipo numérico, decimal o un texto

- Insert Date: en este paso se le da la posibilidad de ingresar una fecha al ciudadano científico

- Insert Time: en este paso se le da la posibilidad de ingresar una hora al ciudadano científico

Para dar solución a estos requerimientos, se implementará una aplicación web la cual va a estar compuesta por una pantalla principal, donde cualquier usuario pueda entender la funcionalidad de Muestre.AR permitiéndole autenticarse o registrarse en el sitio. Para registrarse en el sitio, el usuario deberá proporcionar nombre de usuario, el cual debe ser único, contraseña y de forma opcional, una institución a la cual el científico pertenece. Una vez registrado de forma manual, el usuario podrá vincular su cuenta, con Gmail o con Facebook para acceder.

Una vez autenticado, un usuario en la aplicación web podrá visualizar sus proyectos, tanto los que están activos como los realizados en el pasado. Es importante que los científicos en una primera pantalla tengan la mayor cantidad de información posible para cada proyecto y el estado actual de estos. Por eso se resaltará de cada proyecto:

- Si tiene workflow asociado.

- Si posee resultados.

- Si se encuentra activo.

El científico creador del proyecto podrá invitar a participar del mismo a otros científicos. Al no definir una forma sencilla de diferenciar los proyectos propios de los que un usuario es invitado y para dar mayor claridad a la hora

de visualizar los proyectos, definimos que se mostrarán 2 listados diferentes. Uno de estos tendrá los proyectos propios de un científico, el cual será parte pantalla principal de la aplicación web, y otro sólo con los proyectos a los cuales fue invitado a participar, en una sección específica. Cuando un científico es invitado a participar de un proyecto, este adquiere los mismos permisos que el creador, a excepción de eliminar el proyecto o invitar a otro científico al mismo.

Las operaciones que podrá realizar un usuario autenticado en la aplicación web sobre sus proyectos son:

- Crear o editar el workflow asociado.
- Editar la información propia del proyecto.
- Invitar a otro científico.
- Ver los resultados.
- Eliminar el proyecto.

Crear o editar el workflow asociado.

Como se mencionó anteriormente, el workflow es la secuencia de pasos destinadas a guiar al usuario en la recolección de una muestra. Para la construcción de este se definió utilizar una interfaz gráfica, de forma de hacer más intuitiva y sencilla la construcción. Se definió la utilización de un gráfico con estructura de grafo, donde la interacción del usuario y el grafo se hace a través de una barra de herramientas que permite la construcción del workflow de forma personalizada. Al componente a través del cual el científico construye el workflow, lo llamaremos **Dashboard**. Adicionalmente, se definió que tanto un científico invitado como el creador del proyecto podrán modificar el workflow, si y sólo si este no tenga ninguna muestra recolectada al momento de la modificación. Esto se debe a que si el workflow cambia en su composición, los datos entre muestras serían distintos y difícil de comparar. Una vez finalizada la construcción del workflow, este es validado antes de ser enviado al servidor para almacenarse. Las validaciones que se realizan son que sólo haya definido un solo paso como inicial, que todas los pasos sean accesibles y que todos los pasos estén con la información necesaria de forma completa. Esta definición fue para simplificar la administración del workflow desde el lado del servidor, ya que de esta forma solo recibirá workflows que puedan ser utilizados para recolectar una muestra y no en estados incompletos. Estas validaciones son necesarias ya que Samplers, del cual hablamos oportunamente en el capitulo 2, posee dichas restricciones. De contener un

error en el workflow a validar, el usuario será notificado de forma correcta y específica de donde se encuentra el error.

Editar la información propia del proyecto.

En cualquier momento, será posible, tanto para el usuario que lo cree como para uno invitado cambiar el nombre o la descripción del proyecto.

Invitar a otro científico.

Sólo el científico creador del proyecto podrá invitar a otro ingresando el correo electrónico y un mensaje de invitación. El científico invitado será notificado a través del correo electrónico. De esta forma, el usuario invitado tendrá los mismos permisos sobre el proyecto de quién lo creó a excepción que no podrá eliminar o invitar a otro científico.

Ver los resultados.

Una vez que las muestras para un proyecto comienzan a ser recolectadas, estas son almacenadas en el servidor. Estos resultados podrán ser accedidos tanto por los científicos creadores del proyecto como por los invitados y visualizarse en forma de tabla, la cual indica para cada etapa de la recolección cuál fue el resultado. El formato de la tabla será una fila por cada muestra recolectada y cada paso del workflow será una columna. Adicionalmente podrán descargarse estos resultados en formato CSV.

Eliminar el proyecto.

Un proyecto podrá ser eliminado solo por el usuario que lo creó independientemente del estado en el que se encuentre.

Por otra parte, y como dijimos anteriormente, al contar con *Samplers*, que nos brinda el soporte para la visualización de cada uno de los pasos para la recolección de una muestra, implementaremos una aplicación móvil con las siguientes características.

La aplicación móvil será capaz de interactuar con el servidor para obtener la información de los proyectos, los pasos que componen cada workflow, el envío de los resultados, entre otras cosas como detallamos en la Sección 3.2. Al estar continuamente en comunicación con el servidor, es un desafío importante la implementación de un controlador de peticiones, el cual de forma eficiente sea capaz de realizar las distintas peticiones al servidor y de responder en caso de errores. Además, también consideramos importante que el usuario esté continuamente informado del estado de estas peticiones por lo que será necesario reflejar tal estado en la utilización de la misma. Esta aplicación también deberá listar todos los proyectos para los cuales un usuario pueda recolectar una muestra. Para esto se propuso una sección, en la cual se visualice dicho listado. Sobre cada proyecto presente en el listado se detalla el nombre y el usuario que lo creó. Si un usuario está interesado en conocer en detalle un proyecto del listado anteriormente nombrado, existirá

una sección con la información detallada de cada proyecto donde se indicará el nombre, la descripción, la fecha de creación y un botón con el cual un usuario podrá comenzar la recolección de una muestra para dicho proyecto.

Para la utilización de la aplicación móvil no será necesario estar autenticado. De todas formas brindará la posibilidad de autenticarse. No será posible registrarse desde la aplicación móvil ya que se decidió que era complejo notificar al usuario que si no estaba registrado se crearía un nuevo usuario con sus datos. De esta forma, cuando un usuario quisiera autenticarse y este no se registró previamente a través de la aplicación web, se le notificará que la autenticación fue errónea, el motivo por el cual lo fue y se le brindará un link a la aplicación web para que pueda registrarse.

3.2. Arquitectura

Como anteriormente mencionamos, Muestre.AR es una continuación del proyecto Samplers y supone una mejora del mismo. Para entender mejor la arquitectura de Muestre.AR, es necesario ahondar un poco sobre las limitaciones que presenta el proyecto Samplers y cómo Muestre.AR con su arquitectura resuelve estas. Como explicamos en el capitulo 2.6 Samplers es un framework que permite construir aplicaciones móviles de toma de información aprovechando las herramientas que brindan los dispositivos móviles para la recolección de muestras para proyectos de ciencia ciudadana. El mismo puede ser invocado o llamado de dos formas distintas:

- 1) De forma manual, ésta es la forma de invocar por código al framework.
- 2) Usando un generador de clases provista por Samplers. Lo que se hace acá es copiar el archivo JSON que representa el workflow en la ruta del proyecto y especificar la ruta en un parámetro de configuración. Samplers con su generador de clases, toma este archivo e instancia un workflow y lo muestra

Claramente el segundo método supone una limitación ya que por cada workflow generado por un científico iba a ser necesario la misma cantidad de aplicaciones móviles para su visualización. Además, una vez creado el workflow, el mismo ya no iba a poder ser editado por el científico dueño del proyecto. En un principio, la idea era usar este método, pero observamos que era una solución poco escalable y de poca usabilidad para el usuario ciudadano.

Para solucionar esto, nosotros propusimos la construcción de 3 aplicaciones, como se muestra en la figura 3.1

Por un lado tenemos una sóla aplicación móvil que agrupa todos los proyectos y una aplicación web que sirva como una herramienta para los científicos profesionales de crear de forma sencilla e intuitiva sus proyectos con sus workflows asociados. Para poder lograr esto, se eligió el primer método de comunicación con Samplers, ya que a través de este método, se podrían instanciar los workflows en tiempo real, que era lo que se imponía para la solución que habíamos propuesto. La forma de invocar al framework y los requisitos que impone Samplers para ser llamado se especificarán con más detalle en la sección 3.5. De esta forma, cuando un usuario decida participar en la recolección de la muestra para un proyecto en particular (previamente cargado por el usuario profesional en la aplicación web), para obtener la información del workflow, se realizará una petición al servidor para obtener los datos del mismo.

Como previamente mencionamos, la aplicación movil va a ser una sola y es en ésta donde los usuarios ciudadanos van a poder realizar la recolección de muestras y va a proveer otras funcionalidades que no brinda Samplers, como autenticación, elegir en qué proyecto participar, leer los detalles de cada proyecto, etc. Muestre.AR móvil va a hacer uso de Samplers y va a estar importada dentro de esta como un modulo y se va a llamar al momento de que el usuario decida completar el protocolo de muestra.

Notamos que tanto la aplicación móvil como la aplicación web necesitaban en algunos casos obtener la misma información, por lo que se decidió crear una api REST responsable de brindar servicios a ambas aplicaciones. Otra decisión por la que se optó construir una api Rest es para abstraernos del dispositivo o la tecnología que va a usarse para consumir dicha información. Esto quiere decir, que si el día de mañana se decide por implementar Muestre.AR para IOS, o si se decide que la visualización de los pasos de un workflow sea en una aplicación web, van a consumirse los mismos servicios de la api y estos no van a tener que ser modificados ni readaptados.

3.2.1. Comunicación entre las aplicaciones

El diagrama 3.2 representa el flujo de información entre las distintas aplicaciones que componen Muestre.AR.

Por un lado desarrollamos la aplicación web, en la cual los científicos luego de registrarse pueden crear sus propios proyectos. Luego se construye un *Workflow*, definido este concepto en la sección 3.1 . Una vez que el proyecto es creado, la información es enviada al servidor, y luego guardada en la base de datos.

La API es la encargada de recibir peticiones e información tanto desde

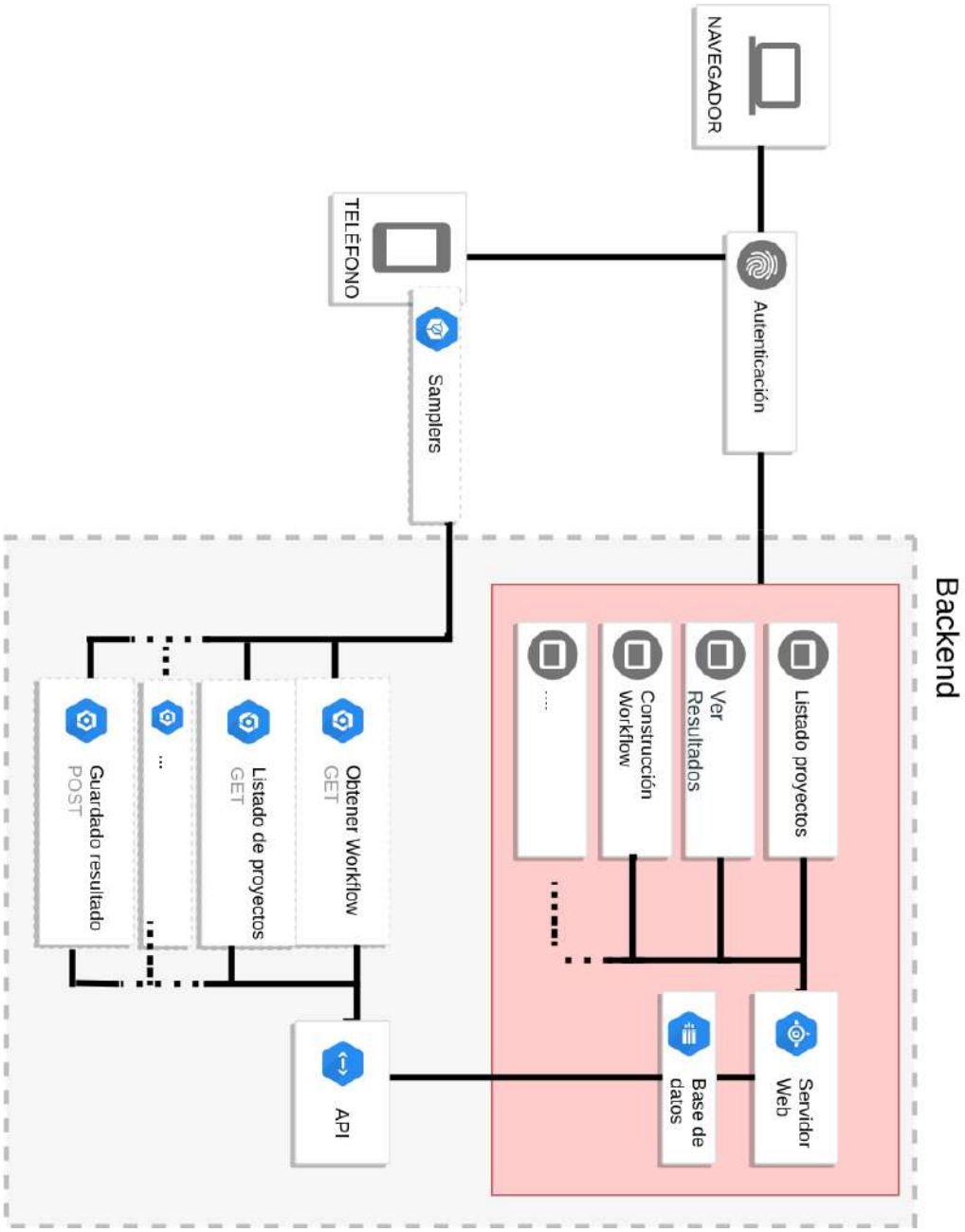

Figura 3.1: Arquitectura de Muestre.AR

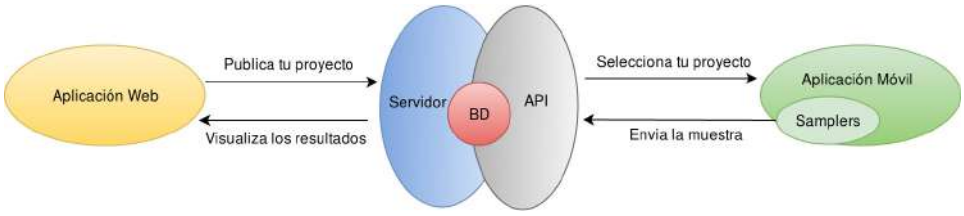

Figura 3.2: Diagrama de flujo de la información.

la aplicación web como de la aplicación móvil. Por ejemplo, a la aplicación movil, a través de un servicio, le brinda la información de los pasos que contiene un workflow. Este mismo servicio es usado tambíen por la aplicación web (más especificamente por el dashboard) al momento de pedir la información de un workflow cuando se desee editar el mismo.

Como dijimos, cuando un usuario ciudadano desde Muestre.AR móvil visualiza los proyectos disponibles y decide participar en uno, se realiza una llamada a la API para solicitar el workflow correspondiente para ese proyecto. Una vez recibida la respuesta, Muestre.AR móvil le cede el control del flujo a Samplers, quien es el encargado de visualizar los pasos correspondientes al workflow. Una vez que el ciudadano científico completa todos los pasos, desde Samplers se llama a la API para realizar el guardado del resultado. El resultado del mismo, como puede contener contenido multimedia, se manda en un archivo .ZIP.

La API cuando recibe este llamado, procede a descomprimir este archivo, y guardar los resultados de los distintos pasos en la base de datos. Los archivos multimedia se guardan en directorios creados dentro del directorio raiz del proyecto.

Cuando un científico desea ver los resultados de un workflow, se realiza un llamado al servidor. En el servidor se recupera los resultados para ese proyecto, y lo devuelve al navegador para luego ser renderizado y mostrado en forma de tabla.

3.3. Aplicación Web

Esta sección del documento está destinada a comprender tanto desde el punto de vista de implementación como funcional de la aplicación web. En la figura 3.3 se observa el diagrama de clases de la aplicación web. Por un lado tenemos una clase Usuario. Hay que tener en cuenta que esta clase es

proporcionada por el framework Django, y modela un Usuario base de una aplicación. Como solamente proporciona algunos datos básicos, se creo otra clase Profile relacionada con esta clase, de manera de poder agregar datos extra sobre el usuario. Por ejemplo, institución a la que pertenece, y nombre de usuario de gmail o de facebook. Los usuarios registrados pueden crear varios proyectos, los cuales tienen un solo workflow asociado al mismo. Un Workflow tiene un nombre que lo identifica, y a su vez tiene una lista de pasos (modelado como Step). Un paso puede ser de cualquier tipo de los enumerados en la seccion 3.1. de Requerimientos y tiene un next step que indica cúal es el paso que le sigue en la secuencia de pasos del workflow. En el caso de los tipos de paso de bifurcación o múltiples, estos tienen una lista de opciones asociadas.

Un Workflow puede tener varios resultados. Estos resultados estan modelados bajo el nombre de WorkflowResult. A su vez, cada WorkflowResult puede tener una lista de resultado por tipo de paso. Los tipos de resultado de los pasos se corresponden con los tipos de pasos que puede tener un workflow. El WorkflowResult tambien tiene el usuario quien realizó al workflow. Este campo es opcional, ya que la recolección de una muestra también lo puede realizar una persona que no se haya autenticado en Muestre.AR móvil

3.3.1. Tecnologías usadas

- **JavaScript:** Utilizamos esta tecnología ya que es un lenguaje de programación somple y potente, el cual es soportado por la gran mayoria de los navegadores, y además es compatible con los dispositivos más modernos. Al ser ejecutado desde el lado del cliente, es muy rápido y cualquier función puede ser ejecutada inmediatamente en lugar de tener que contactar con el servidor y esperar una respuesta. Particularmente en Muestre.AR utilizamos Javascript para realizar la comunicación entre el dashboard y la API que provee los datos para la creación del workflow. Con esta tecnología se completan los datos ya cargados a la hora de editar un workflow. También se implementaron transformadores, de los que hablaremos más adelante, encargados de transformar del modelo de datos utilizado por la API al que utiliza el dashboard y también otro transformador, el cual transforma datos del modelo del dashboard al modelo entendido por la libreria Vis para realizar la representación gráfica del protocolo de muestra.

- **Vis:** Es una librería que brinda soporte para graficar el estado del workflow en la pantalla del Dashboard. Elegimos esta en particular

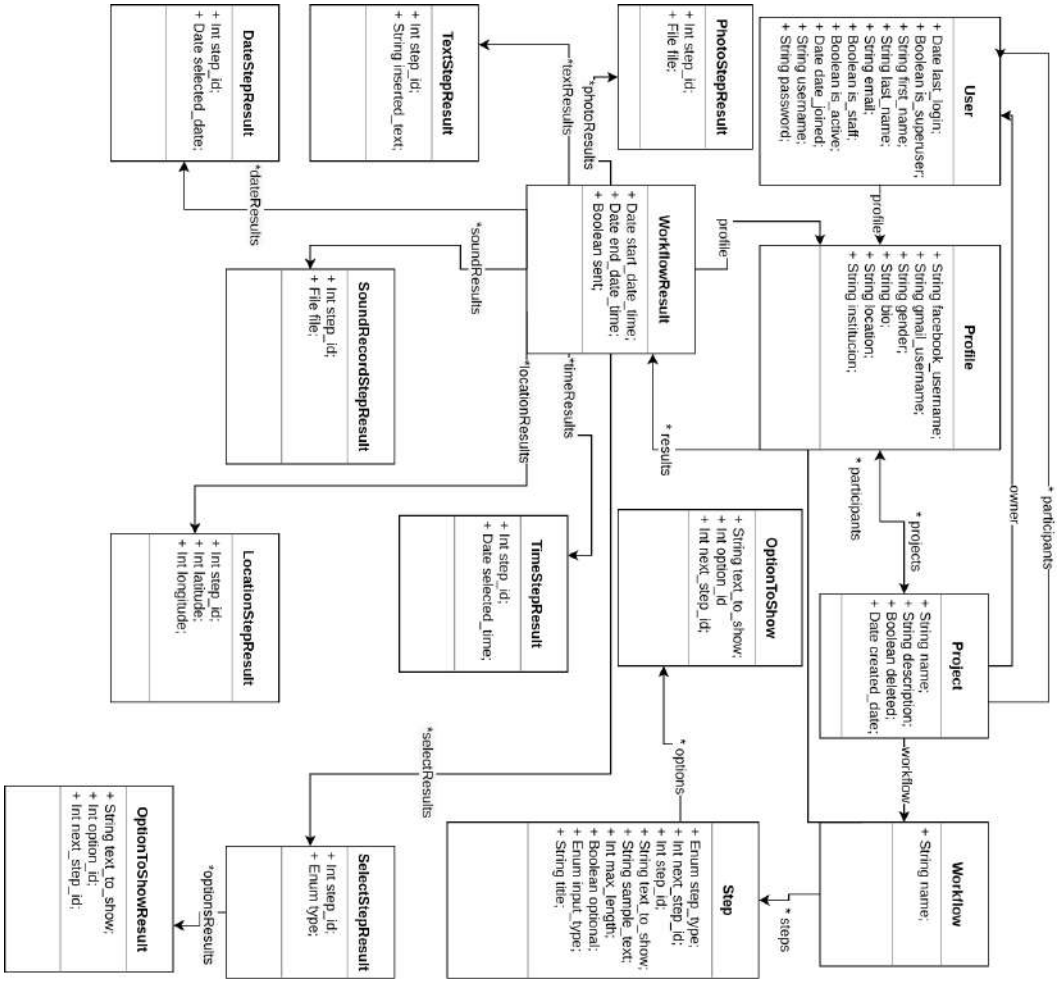

Figura 3.3: Diagrama de clases

debido a su tipo de licencia es libre permisiva. Esto quiere decir, que podemos utilizar y distribuir el software libremente[24]. Además nos resultó sencillo el modelo de datos que necesita para dibujarse y los graficos que se pueden realizar van acorde al workflow que se desea representar. El sitio oficial es *https://visjs.org/*.

- **Python:** este lenguaje de programación fue elegido y concensuado con los directores de tesis para desarrollar la parte del servidor de la aplicación web. Nuestra idea original fue desarrollarlo en Java, ya que estabamos más familiarizados con el lenguaje, pero encontramos en Python un lenguaje mas sencillo y liviano para lo que la aplicación web requería.

- **Django:** se eligió este framework ya que es uno de los más conocidos para facilitar el desarrollo de aplicaciones web en lenguaje Python. Es un framework que respeta el protocolo MVC (más detalles en la sección de trabajos relacionados) y se caracteriza por su sencillez, escalabilidad y tener una curva de aprendizaje alta.

- **SQLite:** motor de base de datos utilizado por Django por defecto. Para los propósitos de la aplicación es suficiente. Igualmente si se dea cambiar de motor de base de datos en un futuro, solamente basta por cambiar el motor por el que se desea en un archivo de configuración, sin mayores implicancias.

- **Python Social Auth:** librería encargada de permitir la autenticación contra muchas redes sociales. En la aplicación se utiliza para autenticarse contra Facebook y Gmail. Versátil y personalizable.

3.3.2. Implementación

Como se mencionó en la anterior sección, la parte del servidor de la aplicación está implemendata con el framework Django, que permite construir una aplicación respetando el patrón comocido como MVC. Debido a que la Ç"(controlador) es manejada por el mismo framework, la parte más importante se produce en los modelos, las plantillas y las vistas,

3.3.3. Configuración de la Base de Datos

Una de las ventajas que proporciona Django es que está desacoplado del motor de base de datos. Esto quiere decir que se puede usar indistintamente uno de otro. Django en su configuración por defecto utiliza una base de

datos sqlite ya que está pensado para usuarios sin experiencia con base de datos. A los efectos de la aplicación Muestre.AR web, nos pareció que esta de base de datos era más que suficiente. Esta configuración se encuentra en el archivo settings.py, y si alguna vez se decide cambiar por alguna otra base, solamente se debe modificar acá, y no es necesario modificar nada de código. La misma configuración es :

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

3.3.4. Interacción con Python Social Auth

Como se detalló en la sección de Tecnologías Usadas, se utilizó un framework para realizar la autenticación con Facebook y Gmail. Este framework se llama Python Social Auth [25]. La manera de realizar la autenticación, es a través del siguiente django template. Para realizar la autenticación con Facebook, se debe realizar la siguiente llamada en un template tag de django. El template tag de django fue debidamente explicado en el capítulo 2.7

```
{% url 'social:begin' 'facebook' %}
```

Para autenticarse con gmail, se debe colocar:

```
{% url 'social:begin' 'gmail' %}
```

El framework lo que hace internamente es realizar la llamada a la API de Facebook y Gmail, bajo los parámetros de configuración indicados en el archivo settings.py. Los cuales son los siguientes:

```
SOCIAL_AUTH_FACEBOOK_SCOPE = ['email']
SOCIAL_AUTH_FACEBOOK_EXTRA_DATA = ['first_name',

'last_name','username','email','token_for_business']

SOCIAL_AUTH_GOOGLE_OAUTH2_EXTRA_DATA = ['first_name',

'last_name','username']
SOCIAL_AUTH_ADMIN_USER_SEARCH_FIELDS = ['username', 'first_name',
```

```
'email', 'extra_data']
)
```

Se procede a explicar los parámetros de configuracion anteriores: Vemos que para la api de Facebook aparece un parámetro llamado SCOPE. Los scopes determinan qué acciones se pueden realizar en nombre del usuario. La aplicación pide ciertos scopes cuando un usuario se autoriza en una aplicación.El scope de correo electrónico permite que la aplicación lea el correo electrónico del usuario. Luego tenemos los parametros EXTRA DATA que son los campos que se desea que se incluyan en la respuesta de la API. Estos son first name, last name, username, email y token for business. El parámetro token for business se explica con más detenimiento en la sección 3.3.5 En lo que respecta a los parámetros de la API de Gmail, los parámetros que se desea que se incluyan en la respuesta son el username, el first name y un campo extra data. En el campo extra data se incluyen datos como fecha de autenticación, tiempo de expiración del token, tipo de token, email, etc.

El comportamiento que tiene por defecto el framework es que si el usuario no se encuentra registrado en la base de datos, automáticamente realiza el alta de un nuevo usuario. Como este comportamiento no era el deseado para Muestre.AR, el framework te permite modificar este comportamiento que viene por defecto para que se comporte de manera personalizada. Básicamente, el framework define un template method con una serie de "pasos"(llamados pipelines por el framework). Cada uno de estos pasos realiza un comportamiento por defecto, que se los puede o bien redfinir, o bien agregas más pasos. Esta configuración se realiza en el archivo settings.py. A continuación, se listan los pasos que se realizan por defecto:

```
SOCIAL_AUTH_PIPELINE = (
    'social_core.pipeline.social_auth.social_details',
    'social_core.pipeline.social_auth.social_uid',
    'social_core.pipeline.social_auth.auth_allowed',
    'social_core.pipeline.social_auth.social_user',
    'social_core.pipeline.social_auth.associate_user',
    'social_core.pipeline.social_auth.load_extra_data',
    'social_core.pipeline.user.user_details',
)
```

Dado que por las definiciones del sistema se tenía que impedir que se realice un alta de un usuario cuando no existiese ese usuario en la aplicación, se agregó un paso que valide esto:

```
SOCIAL_AUTH_PIPELINE = (
```

```
'social_core.pipeline.social_auth.social_details',
    'social_core.pipeline.social_auth.social_uid',
    'social_core.pipeline.social_auth.auth_allowed',
    'social_core.pipeline.social_auth.social_user',
    'webpage.pipelines.user_must_exists',
    'social_core.pipeline.user.get_username',
    'social_core.pipeline.social_auth.associate_user',
    'social_core.pipeline.social_auth.load_extra_data',
    'social_core.pipeline.user.user_details',
)
```

El paso en cuestión que se agregó es 'webpage.pipelines.user-must-exists el cual contiene una lógica para validar que si el usuario no existe en la base de datos, se lance una excepción. Esta excepción es capturada en el manejador de excepciones, y lo que hace es redirigir al usuario al formulario de carga normal.

3.3.5. Autenticación con Muestre.AR web y Muestre.AR móvil

Muestre.AR permite la autenticación por medio de Facebook y Gmail. A continuación se procede a explicarlo:

- Facebook: para facebook se utilizó un campo llamado token-for-business. En la configuración de la aplicación web (settings.py, detallado más arriba) se agregó como opción para que se realice la llamada al servicio de Facebook con este parámetro. Cabe aclarar que este campo no viene por defecto, sino que hay que indicar que lo mande. Esto lo que hace es que en la respuesta del servicio de Facebook se devuelva un access token, con lo que se puede identificar al usuario independientemente de donde se esté autenticando. Este access token es persistido en la base de datos junto con toda la información que se recibe desde el servicio. Esto permite que cuando se vuelva a autenticar desde muestre.AR móvil, el servicio de Facebook devuelva el mismo access token y se pueda identificar que es el mismo usuario que el previamente registrado en la aplicación web. Cabe aclarar que si primero un usuario se quiere autenticar desde el dispositivo móvil y no se registró previamente en Muestre.AR web, el sistema va a informarle que primero se debe registrar en la aplicación web.

- Gmail: similar a facebook, pero Gmail en vez de utilizar un token, utiliza un identificador que es el email del usuario, por lo que no hay

que especificar ningún parámetro extra en el archivo de configuración. Desde la aplicación móvil cuando un usuario se autentica, si el usuario se registró previamente en la aplicación web, el servicio responde con este identificador y de esta forma se verifica que se corresponde a un usuario registrado. De lo contrario, se le va a pedir al usuario que primero se registre en Muestre.AR web.

3.3.6. Vistas

En esta subsección se describirán las distintas vistas que posee la aplicación web junto con su implementación.

Inicio de Sesión

Esta es la pantalla de inicio que se le muestra a un usuario que entra por primera vez a la aplicación web y no se encuentra registrado en el sistema. La misma se muestra en la figura 3.4 En esta pantalla se encuentran dos opciones:

- Hola, ingrese aquí: Esta opción es para cuando el usuario se registró previamente en el sistema. Si se aprieta el botón, se despliega el pop up de la figura 3.6

  En esta instancia, el usuario podrá elegir iniciar sesión con sus credenciales o a través de Facebook o Gmail. Hay que tener en cuenta que el usuario sólo va a poder autenticarse a través de Facebook o Gmail si previamente se registró de forma manual y haya asociado las cuentas. Si no se realiza esto y se decide ingresar igual, el sistema hará una redirección a la pantalla del registro manual invitando al usuario a que se dé de alta en el sistema.

- ¿ Aún no tenés usuario? Registrate ahora: Esta opción es para registrarse en el sistema. Si se selecciona, se mostrará el siguiente formulario.

  Se deberá completar todos los campos. A tener en cuenta que todos los campos son obligatorios, a excepción del campo Institución(institución a la cual pertenece el científico). Una vez completado todos los campos, el sistema validará los mismos, y de ser correcto los mismos, se le mostrará la pantalla principal de la aplicación.

Figura 3.4: Index.

Figura 3.5: Registro

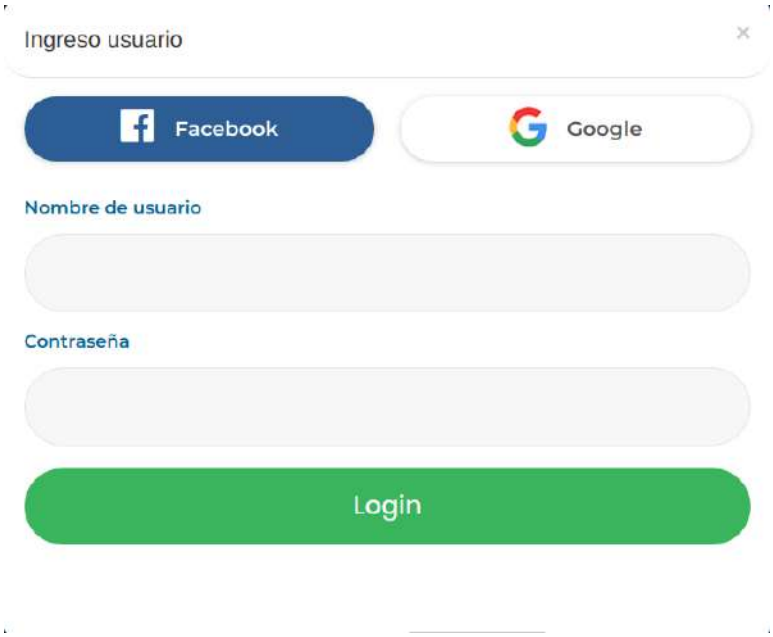

Figura 3.6: Inicio Sesión

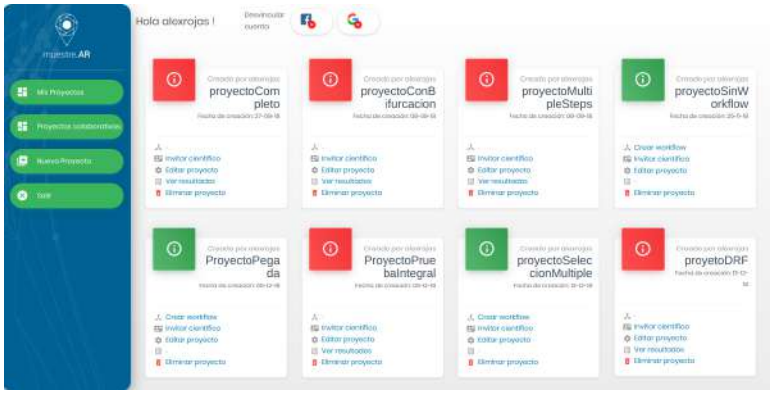

Figura 3.7: Pantalla Principal

Principal

Esta es la pantalla que se le presenta a un usuario despuúes de loguearse. En esta pantalla podemos identificar 3 secciones:

En la parte superior, están las opciones para que se pueda vincular la cuenta activa tanto con Facebook y/o Gmail. Esto lo que permite, es que la próxima vez que se desee iniciar sesión en la aplicación web, directamente se ingrese con la opción de Facebook y Gmail, sin necesidad de poner el nombre de usuario y contraseña que se utilizó a la hora de realizar el registro manual. Si se vinculó la cuenta, aparecerá la opción para desvincularla por si algún motivo el usuario desee hacerlo. En la parte lateral izquierda tenemos el panel de navegación. En él se encuentran las opciones de :

- Mis proyectos: seleccionando esta opción, se listan los proyectos creados por el científico de la sesión activa.
- Mis proyectos: seleccionando esta opción, se listan los proyectos a los cuales el científico de la sesión activa fue invitado a participar.
- Nuevo proyecto: opción para crear un nuevo proyecto. El mismo requiere un nombre y una descripción.

Por último, en el sector medio tenemos el listado de proyectos. Dentro de cada proyecto se pueden realizar las siguientes acciones:

- Crear workflow: esta opción permite al científico crear su protocolo de muestra en un dashboard intuitivo. En la sección Dashboard se brindará más detalle.

- Invitar científico: esta opción permite invitar a un científico a formar parte del proyecto. Al científico invitado se le enviará un mail con un mensaje que informa del proyecto invitado e información del científico que lo creó. A su vez, si el científico no existe en el sistema, se le enviará un mail invitándolo a que se registre. El científico invitado podrá editar un workflow, ver sus resultados, pero no podrá invitar a otro científico a formar parte del mismo ni podrá eliminar el proyecto.
- Ver Resultados: opción para mostrar los resultados de un workflow. Más detalle en la sección Resultados.
- Eliminar Proyecto:: seleccionando esta opción se procede a eliminar un proyecto. A tener en cuenta que la eliminación del proyecto no provoca un borrado físico en la base de datos, sino lógico.

Un detalle a tener en consideración en el listado de proyectos es el color del cuadrante en la esquina superior izquierda. El mismo puede tener dos colores: verde o rojo. Si el color es verde, significa que todavía no se obtuvieron resultados de muestras para ese proyecto, por lo que el científico puede seguir editando el workflow a su conveniencia. En cambio, si para el proyecto ya existen resultados de muestras, el mismo tomará el color rojo, impidiéndole al científico editar el workflow.

Resultado

Al seleccionar la opción Ver Resultados de un proyecto, se mostrará la siguiente pantalla

Esta pantalla muestra en forma de tabla los resultados de los proyectos. En el encabezado de la tabla se detalla la fecha de comienzo del protocolo de muestra, la fecha de fin, y todos los steps que forman parte del protocolo de muestra. Tener en cuenta que los steps se pueden repetir por protocolo de muestra. También estos mismos datos se pueden descargar en formato csv, esto se logra presionando sobre el botón Descargar Csv.

Dashboard

El *dashboard* o pizarra es uno de los componentes principales de la aplicación web ya que es la herramienta a través de la cual el usuario profesional diseña y configura el workflow para la recolección de una muestra. El objetivo de este dashboard es ser el medio a través del cual un usuario científico configura una secuencia de pasos y obtiene como resultado una secuencia de pantallas que se reproducirán en la aplicación móvil para ayudar a un

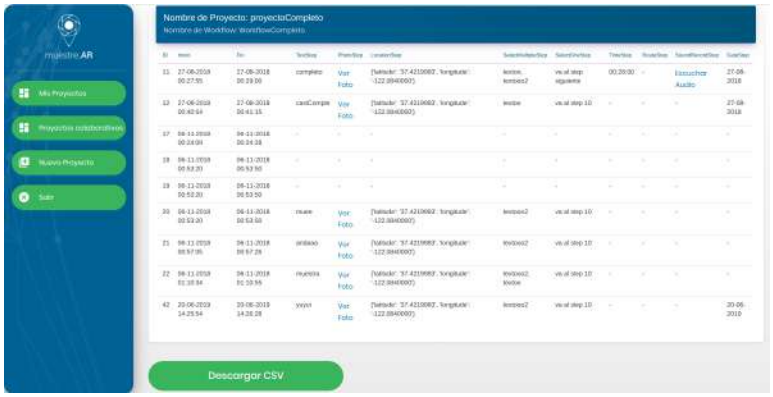

Figura 3.8: Resultado.

usuario ciudadano a recolectar una muestra. Estas pantallas sirven para que quien está recolectando la muestra tome la información necesaria para esta. La figura 3.9 muestra un ejemplo de un dashboard configurado y la secuencia de pasos generada a partir de este.

La implementación del dashboard es en JavaScript y utilizando de soporte la librería Vis comentada anteriormente. Para representar el workflow existe un modelo de datos asociados como puede verse en la figura 3.10, que tiene una clase principal, llamada **Workflow**. La función de esta clase es mantener la relación entre un conjunto de pasos. De esta forma, la clase Workflow es la encargada de agregar nuevos pasos o borrarlos, dependiendo de las acciones que ejecute quien diseña el workflow. A continuación se describe la funcionalidad de los métodos enumerados en la figura 3.10.

- **addStep():** Se ejecuta para agregar un paso nuevo indicando cual es su predecesor. Como explicaremos más adelante, el comportamiento del paso agregado dependerá de su tipo.

- **DeleteStep():** Este método es el encargado de borrar un step existente en el workflow. Si se borra un nodo intermedio, es decir un paso que tiene uno antes y uno siguiente, se genera un grafo no conexo. Otra posibilidad, es que el tipo de paso a borrar sea multiple, en ese caso, se borran también todas las opciones que tenga relacionadas.

- **isValid():** Su objetivo es validar que el workflow cumpla con todas las condiciones necesarias para ser guardado en el servidor. Entre estas

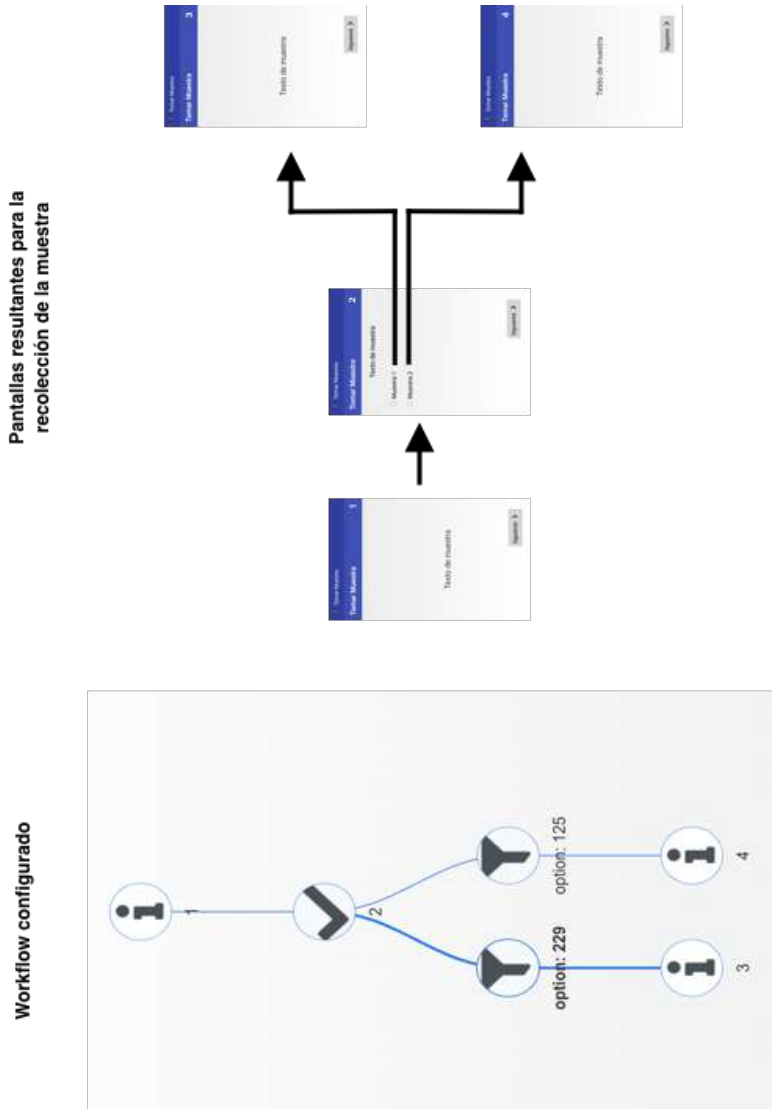

Figura 3.9: Workflow ejemplo con pantallas Android asociadas.

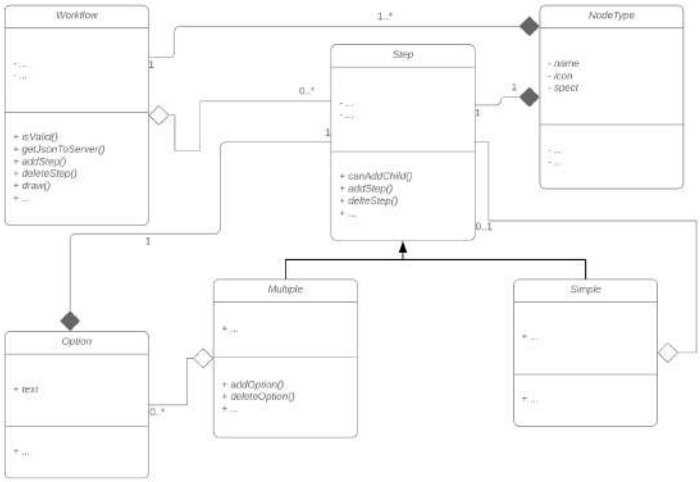

Figura 3.10: Modelo de clases del dashboard.

condiciones podemos nombrar que solo exista un paso inicial o que todos los atributos de la configuración de cada paso esten completos.

- **getJsonToServer():** Una vez validado el workflow, se ejecuta este método, para obtener el modelo de datos requerido por la API para la creación del workflow.
- **draw()** : Tiene la función de *Transformador*, a partir de un workflow representado a traves de un modelo de datos genera otro conocido por la libreria Vis para poder graficarlo.

La clase **Step**, es la encargada de modelar el comportamiento que tienen en común los pasos. Esta clase es abtracta y todas las clases que extiendan de esta deben implementar los siguientes métodos:

- **addStep(step)** .
- **DeleteStep(step)**.
- **canAddChild()**.

Cada Step, contiene un objeto de la clase **NodeType**, la cual tiene como objetivo modelar los distintos tipo de paso que se puede representar en el

dashboard. Cada objeto de este tipo cuenta con un nombre, un ícono asociado y spects. Cada tipo de paso tiene su ícono que lo identifica visualmente cuando se grafica este paso en el dashboard. En el atributo spects es donde cada tipo de paso guarda la información específica que es cargada por el usuario científico cuando diseña la muestra. Como por ejemplo, pueden ser el texto a mostrar en cada pantalla o las opciones entre las que se puede elegir en un paso de selección multiple.

Los objetos de tipo Workflow conocen una colección de objetos NodeType. Dependiendo de la cantidad de objetos de este tipo que conozca el workflow son los posibibles tipos entre los que va a poder optar para configurar un paso el usuario que disene la muestra. Es decir, si un workflow tiene una colección de dos objetos de tipo NodeType, el usuario que diseña la muestra, solo podrá configurar pasos de esos dos tipos.

En nuestra implementación existen dos clases que extienden o heredan de Step, **Simple** y **Multiple**. La primera es la encargada de representar a los pasos de tipo: Selección multiple, Localización, Hora, Fecha, Texto, Audio, Foto e Información ya que entre estos existe un comportamiento en común. De esta forma, el paso de tipo decisión, es representado a través de la clase Multiple. El nombre de éstas, se deben a la cantidad de pasos siguientes que pueden tener. De ahora en adelante llamaremos hijo al paso siguiente de un paso determinado dentro de un workflow. Mientras que para los objetos de la clase Simple solo pueden tener un único hijo, los de tipo Multiple pueden tener uno o más. Esta diferencia en la funcionalidad se alcanza con la implementación de los métodos de la clase abstracta. Por ejemplo, como se muestra en el siguiente extracto de código, se puede ver las distintas implementaciones del método **canAddChild()**.

```
class Simple extends Step {
        canAddChildStep() {
                return !this.next;
        }
}

class Multiple extends Step {
        canAddChildStep() {
                return true;
        }
}
```

De esta forma, se puede apreciar en el código que para validar la condición que los objetos de tipo Simple permitan agregar un hijo es sólo si este

no tiene un hijo ya definido. En cambio, por su parte, los de tipo Multiple no tienen ninguna condición y siempre pueden agregar un hijo nuevo.

Otros método en el cual varía bastante la funcionalidad es en el **addStep(step)**. El parámetro que se recibe *step*, es el paso siguiente al que ejecuta la función. Debido a que los objetos de tipo Simple, solo cuentan con un sólo paso siguiente, en el caso de estos objetos, se asigna al atributo next el valor recibido en el parámetro step. En contraparte, los de tipo Multiple, crean una opción y como atributo next de esta, ponen el valor del parámetro.

Los objetos de tipo **Option**, sirven para representar a cada una de las opciones entre las que puede elegir un usuario que está recolectando una muestra. Cada una de estas opciones está asociada a un paso siguiente. Además cada Option tiene un texto, que es el que lo representa cuando el usuario debe elegir. Gráficamente, cada opción es modelada como un nodo en el dashboard, es por eso que también tiene un atributo de tipo NodeType.

Cuando se desea crear un nuevo workflow, la configuración inicial del dashboard es con un único paso de tipo información y sin ningún dato. Si se quiere editar un workflow, la configuración será la misma que cuando lo guardó.

Operaciones disponibles sobre dashboard

El dashboard cuenta con varias operaciones que permiten al usuario científico diseñar de forma personalizada la recolección de la muestra para su proyecto. Entre las operaciones que se ofrecen están las que se realizan sobre un paso específico. A partir de ahora llamaremos *Nodo* a cada representación gráfica de un paso en el dashboard. También existen operaciones que se realizan sobre las conexiones entre estos nodos, a las cuales llamaremos *Arístas*. Ademas, existen nodos intermedios, que representan a cada una de las opciones por las que puede elegir un usuario en el paso de decisión. A estos nodos intermedios, los llamaremos *Opciones*. Para las opciones también se definieron operaciones especiales.

Operaciones sobre nodos

Una vez seleccionado el nodo sobre el que se quiere realizar la operación, una barra de tareas se despliega para poder trabajar con éste. Como se puede ver en la figura 3.11 las operaciones disponibles sobre un nodo son:

- **Agregar siguiente paso:** Esta operación agrega un paso como siguiente del seleccionado. Este nuevo paso agregado es por defecto de tipo información, pudiendo luego modificarse.

- **Configurar:** A través de esta operación, el usuario que diseña la muestra puede especificar qué tipo de paso va a representar dentro de la

recolección. Ademas de seleccionar el tipo, el usuario debe completar los datos que cada tipo de paso requiere. Hablaremos más adelante de la configuración de cada uno de los nodos.

- **Agregar paso de decisión como siguiente:** Esta operación permite al nodo seleccionado agregarle un hijo que sea de tipo decisión. Este tipo de paso no puede ser modificado, pero al igual que el resto debe ser configurado.

- **Agregar padre:** Esta operación permite que el nodo seleccionado sea el siguiente de otro nodo existente que no tiene hijo o sea de tipo multiple.

- **Borrar**: Con esta operación se borra el nodo seleccionado y todas las aristas relacionadas a éste. Si es un nodo de tipo Opcion, también borra el nodo siguiente asociado.

Operaciones sobre arista

Al igual que los nodos, cuando una arista es seleccionada una barra de tareas es desplegada y la única operación disponible es borrarla. Si la arista seleccionada representa la relación entre 2 pasos, solo se borra esta relación. En caso de ser entre una opción y un paso se borran ambos.

Operaciones sobre opciones

Cada opción tiene como posible operaciones:

- **Borrar:** Borra la opción seleccionada y el nodo asociado que representa al siguiente paso.

- **Configurar:** Al igual que los nodos, las opciones deben ser configuradas. En este caso, se indica el texto que representará a esta opción cuando el usuario este recolectando la muestra.

Configuracion cada paso A partir de cada paso configurado en el dashboard, cuando se finaliza la construcción del Workflow, se interpreta la información y se almacena de forma tal que luego pueda ser recibida e interpretada por la aplicación móvil y con la utilización de Samplers, se termina generando una pantalla asociada a dicho paso destinada a la recolección de la muestra. A continuación se describen como se puede configurar cada tipo de paso. Como fue detallado anteriormente, para configurar un paso es necesario seleccionarlo y optar por la operación *configurar*. Esto hará visible un componente llamado **Modal de configuración**, el cual cuenta de 2 pasos. El primero para sirve para seleccionar el tipo de paso que se desea configurar, tal cual muestra la figura 3.12 y el segundo, donde el usuario científico

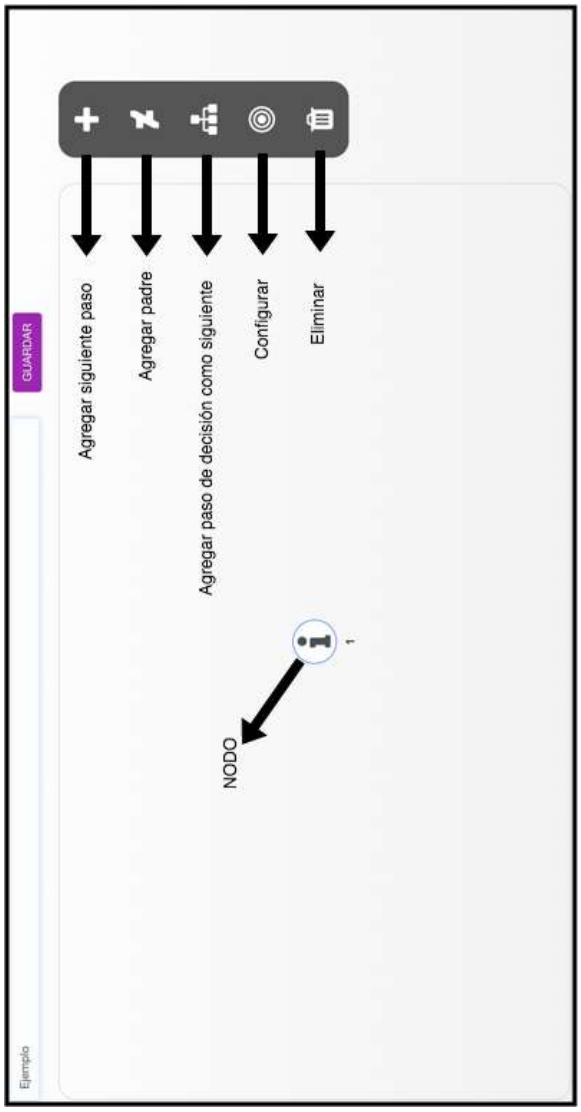

Figura 3.11: Dashboard inicial con barra de operaciones disponibles sobre un nodo.

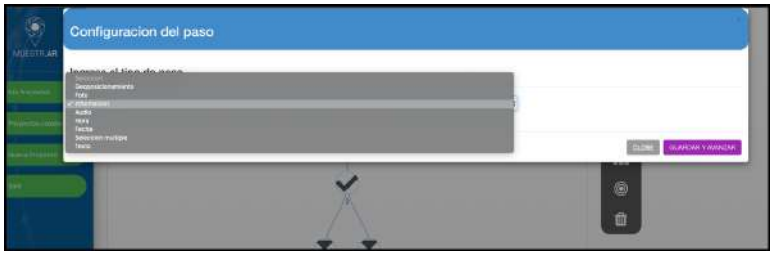

Figura 3.12: Primer paso del modal de configuración

Figura 3.13: Modal de configuración de un paso de información con la correspondiente pantalla generada en la aplicación Android.

que diseña la muestra carga información específica para cada paso como se describe a continuación.

El tipo de paso que se utiliza por defecto es el de Información, como se muestra en la figura 3.11, cada vez que un nuevo nodo es creado se corresponde a este tipo. Para estos pasos, sólo es necesario carga un texto a mostrar, obteniendo como resultado de la configuracion la pantalla android reflejada por la figura 3.13. Los tipo de pasos de fecha, hora, localización, audio y foto al igual que el de información sólo requieren un texto a mostrar para ser configurados.

A diferencia de los anteriores, los pasos de Seleccion múltiple para ser configurados además del texto a mostrar, se deben cargar las posibles opciones entre las cuales el usuario que recolecta la muestra debe optar. El usuario ciudadano puede seleccionar una o más de estas opciones disponibles. Para agregar una nueva opción se selecciona en el texto *Click aqui para agregar una opcion ...* y luego se completa el nuevo campo mostrado con la opción que se desea mostrar.

A diferencia de los pasos anteriores para configurar el de decisión es necesario primero y principal crearlo con la operación sobre nodos previamente

mencionada, *Agregar paso de decisión como siguiente*. Cuando este es configurado, como muestra la figura 3.12, no es posible cambiarle el tipo y lo que se puede cargar es un texto a mostrar. Este es el único tipo de paso que puede tener más de un paso siguiente. Para poder relacionar cada una de las opciones que puede optar el usuario que está recolectando la muestra con su paso siguiente se deben configurar unos pasos intermedios llamados *Opciones*, y para esto sólo se requiere un texto, el cual va a representar a una de las opciones entre las cuales puede seleccionar el usuario ciudadano. Cada opción obligatoriamente tiene un paso siguiente, el cual puede ser de cualquier tipo. Cuando una opción se borra del dashboard, su paso siguiente también es eliminado.

3.3.7. Diseño visual

Esta tarea fue llevada a cabo por nuestra Asesora Profesional en esta tesina llamada Julieta Lombardelli, graduada en Licenciatura en diseño multimedia, con quien tuvimos varios encuentros en los cuales fuimos mejorando los diseños.

Antecendentes

Se realizó el diseño de la web de Muestre.AR, considerando la importancia del diseño centrado en el usuario. Se decidio que era necesario generar una identidad de producto y facilitar el aprendizaje de la herrameinta por parte de los usuarios.

Objetivos

Entre los objetivos propuestos para este diseño son:

- Mejorar la usabilidad.
- Apoyar los contenidos para una comprensión rápida y clara para los usuarios.
- Obtener una navegabilidad intuituva de la pagina.
- Generar una identidad clara y definida.

Descripción

- **Impresión general:** diseño limpio, con terminaciones redondeadas y organización en la pagina por bloques y cajas. El diseño redondeado en

cajas y botones responde a la implementación de mejorar la experiencia de usuario, puesto que son más fácilmente reconocibles para el ojo humano dentro de la organización de la página. Esto posibilita una navegación más intuitiva y ágil. En el logo se combina la imagen de un diafragma de cámara de fotos con nodos interconectados y el icono de geolocalización en el mapa.

- **Paleta de colores:** Se dispone una paleta fría con acentos de colores brillantes y en contraste, para enfatizar los distintos pasos en la navegabilidad y acceso a información.
- **Tipografía:** sans-serif, contemporánea, legible, en los distintos tamaños se pondera la jerarquización de los contenidos.

3.4. API

Esta es una RestFul Api que surgió como necesidad de ofrecer los mismos servicios tanto a la aplicación móvil como a la aplicación web. La respuesta de los servicios son en formato JSON, ya que creemos que es el formato de intercambio más sencillo y el más estandarizado para la comunicación de aplicaciones. La función principal de esta API es cuando desde Muestre.AR móvil un usuario ciudadano decide participar de un proyecto, se realiza una petición para obtener el workflow correspondiente para este proyecto, la API atiende esta petición, obtiene el workflow persistido en la base de datos, serializa esta información y la devuelve en formato JSON. Este servicio también es llamado desde el dashboard al momento de visualizar un workflow ya creado.

Los servicios que ofrece y su estructura detallados a continuación:

- Obtener workflow: este es el servicio explicado en el párrafo anterior. Dado un identificado de workflow, devuelve el workflow con todos los pasos asociados a el.

  Endpoint: **GET** /webpage/workflow/:workflowId

  Path Parameter:

  workflowId (Requerido) : identificador del workflow

  Ejemplo de Respuesta:

```
{
"data": {
    "id": 10,
```

```
            "name": "Workflow 2 pasos",
            "project": 12,
            "steps": [
                {
                    "step_type": "InformationStep",
                    "next_step_id": 2,
                    "id": 1,
                    "text_to_show": "Informacion"
                },
                {
                    "step_type": "TextStep",
                    "next_step_id": 1,
                    "id": 2,
                    "text_to_show": "Texto a mostrar",
                    "sample_text": "texto de muestra editado",
                    "max_length": 2,
                    "optional": true
                }
            ]
        },
        "status_code": 200
}
```


Respuesta

200: respuesta OK

404: en el caso de que no se haye el identificador del workflow mandado en la url

- Actualización de un workflow este servicio es llamado desde la aplicacion web (dashboard) al momento de editar un workflow y mandarlo a guardar.

  Endpoint: **PUT** /webpage/workflow/:workflowId

  Path Parameter:

  workflowId (Requerido) : identificador del workflow a actualizar

  Ejemplo de Request Body:


```
{
"data": {
```

```
        "id": 10,
        "name": "Workflow 2 pasos",
        "project": 12,
        "steps": [
            {
                "step_type": "InformationStep",
                "next_step_id": 2,
                "id": 1,
                "text_to_show": "Informacion"
            },
            {
                "step_type": "TextStep",
                "next_step_id": 1,
                "id": 2,
                "text_to_show": "Texto a mostrar",
                "sample_text": "texto de muestra editado",
                "max_length": 2,
                "optional": true
            }
        ]
    },
    "status_code": 200
}
```


Respuesta

200: respuesta OK

- Creación de workflow: este servicio es llamado desde el dashboard al momento de crear un nuevo workflow. El mismo espera el workflow con todos los pasos asociados al mismo.

  Endpoint: **POST** /webpage/workflow

  Ejemplo de Request Body:


```
{"name": "WF2Locations",
    "project": 55,
    "steps": [
        {
            "step_type": "InformationStep",
            "next_step_id": 2,
```

```
                "id": 1,
                "text_to_show": "texto_informativo"
            },
            {
                "step_type": "LocationStep",
                "next_step_id": 3,
                "id": 2,
                "text_to_show": "posicion1"
            },
            {
                "step_type": "LocationStep",
                "next_step_id": null,
                "id": 3,
                "text_to_show": "posicion2"
            }
            ]
}
```


- Listado de proyectos : este servicio es llamado desde la aplicación móvil para listar todos los proyectos que se hubiesen dado de alta desde la aplicación web.

  Endpoint: **GET** /webpage/projects/

  Ejemplo de Respuesta:


```
{
 "data": [
     {
         "id": 1,
         "name": "proyectoCompleto",
         "owner": "alexrojas"
     },
     {
         "id": 2,
         "name": "proyectoConBifurcacion",
         "owner": "alexrojas"
     },
     {
         "id": 4,
         "name": "proyectoMultipleSteps",
         "owner": "alexrojas"
```

```
        }
    ],
    "status_code": 200
}
```


Respuesta

200: respuesta OK

- Detalle de proyecto (GET): este servicio es llamado desde la aplicación móvil al momento de seleccionar un proyecto de la lista de proyectos devuelta por el servicio anterior. El mismo devuelve todos los campos del proyecto.

  Endpoint: **GET** /webpage/project/:projectId

  Path Parameter:

  projectId (Requerido) : identificador del proyecto a devolver

  Ejemplo de Respuesta:


```
{
    "data": {
        "name": "proyecto2LocationSteps",
        "description": "2LocationSteps",
        "workflow": 26,
        "created_date": "2019-06-20"
    },
    "status_code": 200
}
```


  Respuesta

  200: respuesta OK

  404: en el caso de que no se haye el identificador del proyecto mandado en la url

- Guardado de resultado de un workflow (POST): este servicio es llamado desde la aplicación móvil (específicamente desde Samplers) cuando se completaron todos los pasos de un workflow por el usuario ciudadano y se requiere guardar los resultados. Este servicio recibe un archivo .zip con la información completa de todos los pasos, junto con los archivos multimedia (si es que los hay). Acá se deserializa toda la información, para luego persistirla en la base de datos.

Endpoint: **POST** /webpage/workflow/:workflowId:/result

Path Parameter:

workflowId (Requerido) : identificador del workflow sobre el cual se está guardando el resultado

Ejemplo de Request Body:

archivo .ZIP

Respuesta

200: respuesta OK

- Validación de usuario logueado: este servicio es llamado desde la aplicación móvil luego de autenticarse con Facebook o Gmail. Luego de la respuesta de la api de estos servicios, desde Muestre.AR móvil se llama a este servicio con un parámetro para indicar si se autenticó con Gmail o Facebook, y este servicio valida si el mismo se encuentra dado de alta en la base de dados de Muestre.AR. Si se encuentra, el usuario completa la autenticación y se loguea en el sistema. Si no se encuentra, se le indica que se registre primero en Muestre.AR web.

  Endpoint: **GET** /webpage/login?uid=uid provider=provider

  Query Parameters

  uid = identificador del usuario.

  provider = nombre del proveedor, ya sea Facebook o Gmail.

  Ejemplo de Respuesta Exitoso:

```
{
"data": {
    "msg": "User exists",
    "exists": true,
    "user_information": {
        "username": "alexrojas",
        "email": "alexrl_lp@hotmail.com",
        "id": 1
    }
},
"status_code": 200
}
```

  Ejemplo de Respuesta No Exitosa:

```
    {
    "data": {
        "msg": "User not exists",
        "exists": false,
        "redirect_url": "http://localhost:8000/login/"
    },
    "status_code": 404
}
```

Respuesta

200: respuesta OK

404: usuario no encontrado en la base de datos

En esta sección explicaremos la tecnología usada y cómo se implementó.

3.4.1. Tecnología usada

Se utilizó DjangoRestFramework (en adelante DRF) que es uno de los frameworks más usados a la hora de construir una aplicación Api REST de forma sencilla sobre Django. Ofrece una alta variedad de métodos y funciones para el manejo, definición y control de los recursos (endpoints).

Estructura básica

DRF consta de 3 partes esenciales: los serializadores, los routes y las vistas. A continuación se explicará cada una, mostrando capturas a modo de ejemplo de forma de ilustrar cómo se implementó en la aplicación.

- Views: no son más que extensiones de las classs-views que vienen en Django.pero enriquecidas de una forma tal que se sea más fácil la interaccion con los routers, los serializadores y los objetos de nuestro modelo. En lugar de renderizar un html como respuesta (que es lo que devuelve las views de Django), estas devuelven un Json, Xml o cualquier otra estructura de datos que se desea que se devuelva como respuesta. La estructura elegida para que sea devuelta en esta Api es JSON. En el siguiente fragmento de código se muestra un ejemplo de cómo se implementó el servicio que devuelve el detalle de un workflow (se llama con el verbo GET)

  ```
  class WorkflowDetail(APIView):
  ```

```
def get(self, request, pk, format=None):
logger.info("Getting workflow with id: %s",pk)
workflow = self.get_object(pk)
serializer = WorkflowSerializer(workflow)
data = serializer.data
return Response({"data":data, "status_code": 200},
status= 200)
```

En el anterior fragmento de código lo que se realiza es primero obtener el workflow correspondiente de la base de datos a partir del id que le llega como parámetro. Cabe mencionar que el id del workflow se manda a través de la url, y se corresponde con el nombre de parámetro "pk". Luego lo que se hace es serializar el objeto de nuestro modelo en un objeto (de nombre data) para que pueda ser devuelto en formato JSON . Luego, se arma un objeto como respuesta que contiene: la respuesta del workflow en formato JSON, y el código HTTP 200 indicando que la solicitud ha tenido éxito.

- Routers: en esta parte se definen las urls de nuestra API de una manera legible y sencilla. Básicamente se lista qué método de una class view se ejecutará al llegar un request HTTP contra el path concreto definido usando un verbo HTTP. Las mismas se detallan en el archivo urls.py. El siguiente fragmento de código muestra cómo seria el router para el servicio anteriormente definido:

```
url(r'^workflow/(?P<pk>[0-9]+)$',
views.WorkflowDetail.as_view()),
```

Acá se define una regex para que cuando una solicitud se corresponda con la misma, se llame al metodo especificado de la vista (en este caso WorkflowDetail). Tener en cuenta que este metodo cuenta con 2 servicios, uno encargado de devolver el detalle de un workflow, y el otro encargado de actualizar un workflow. La diferencia es que para el primero se llama con el verbo Get, y el segundo con el verbo Put. Al ser el mismo endpoint pero con distinto verbo HTTP, DRF se encarga de direccionar el request con el servicio correspondiente.

- Serializers:Los serializadores nos permiten definir en detalle cómo serán las respuestas que devolverá nuestra API y cómo procesaremos el contenido de las peticiones que nos lleguen. Siguiendo con el ejemplo anterior, en el archivo serializers.py es donde definimos nuestros serializadores.

El siguiente serializer se encarga de transformar un proyecto de nuestro modelo en representacion JSON. Este serializer es el que usa el servicio que devuelve el listado de los proyectos.

```
class ProjectSerializer(serializers.ModelSerializer):
        class Meta:
                model = Project
                fields = ('id','name','owner')
```

El siguiente serializer se encarga de transformar un workflow en formato JSON en un workflow de nuestro modelo. Este serializer es el que usa en el momento de creación de un workflow desde el dashboard.

```
    class Meta:
        model = Workflow
        fields = ('name', 'project', 'steps')

    def create(self, validated_data):
        steps_data = validated_data.pop('steps')
        workflow = Workflow.objects.create(**validated_data)
        createWorkflowWithSteps(steps_data, workflow)
        return workflow
```

Lo que se hace acá es guardar en una variable los pasos que componen el workflow. Luego lo que se hace es crear un workflow con el nombre y el proyecto enviado, y se guarda en una variable el objeto workflow creado. Luego, se llama al metodo createWorkflowWithSteps que lo que hace es persistir en la base de datos los pasos asociados al workflow.

3.5. Aplicación Móvil

En esta sección del documento vamos a explicar cómo fue el desarrollo de la aplicación Muestre.AR móvil; justificando la elección de la tecnología, explicando detalles sobre su implementación y comentando cuestiones de diseño y experiencia de usuario que se tuvieron en cuenta para determinar tanto la apariencia como la disposición de los componentes y la funcionalidad de la aplicación.

3.5.1. Tecnología

Para la recolección de una muestra optamos por la tecnologia móvil ya que es accesible todo el tiempo para la mayoría de las personas, esto permite que el usuario pueda realizar esta actividad en cualquier momento y en cualquier lugar mientras tenga conexión a internet. La tecnología móvil que elegimos fue Android ya que es libre, gratuita y multiplataforma [26]. Además, al optar por Android, contamos con un framework llamado Samplers, el cual es el encargado de la visualización del workflow para la recolección de una muestra.

3.5.2. Implementación

Comunicación con Samplers

Como bien explicamos en la Sección 3.2, la forma de comunicación con Samplers es de forma manual, llamando al framework de forma programática. Para realizar esto, Samplers requiere que se cree una clase Activity (nosotros la llamamos StartWorkflowActivity) que se extienda de la clase SamplersMainActivity (provista por Samplers). Esta clase tiene 2 metodos abstractos que requieren ser implementados:

- 1)Por un lado hay que implementar el método getWorkflow(). Este método devuelve un objeto de tipo Workflow. Este objeto de tipo Workflow tiene una lista de pasos y una lista de parámetros configurables.

- 2)El otro método que se tiene que implementar es el onCreate(). Acá se setean prámetros de conectividad, por ejemplo, se setea la url a la cual se le tiene que enviar el resultado de la muestra recolectada

Cuando el usuario decide participar en la recolección de un proyecto, la clase StartflowActivity es instanciada y realiza una peticion HTTP a la API de Muestre.AR para obtener la secuencia de pasos configurados por el usuario científico previamente cargados en Muestre.AR web. La API responde con un JSON que representa esta secuencia de pasos mencionada, para luego ser mapeada a la clase WorkflowModel que es un objeto propio del dominio de Muestre.AR móvil. Esta clase es usada solamente para mapear la respuesta de tipo JSON en un objeto de nuestro dominio. Esta clase WorkflowModel contiene datos como la lista de pasos, el nombre del workflow, y el id de proyecto al cual pertenece este workflow. Se creó una clase utilitaria llamada WorkflowMaker.java,cuya función es realizar el mapeo entre el objeto de tipo

Workflow provisto por Samplers, y el WorkflowModel previamente descripto. Cabe mencionar que el objeto de tipo Workflow posee una lista de parámetros personalizables, el cual se creó con la idea de agregarle cualquier tipo de información extra que se desee incluir. Por ejemplo, Muestre.AR móvil agrega a esta lista el identificador de usuario de quien realiza el workflow (en caso de que el usuario sea un usueario registado en Muestre.AR).

Lo anteriormente explicado se puede resumir en la siguiente secuencia de pasos:

- 1)Un usuario elige un proyecto que le interese colaborar de la lista de proyectos y selecciona participar
- 2) Muestre.AR móvil realiza una llamada a la Api para obtener el JSON que representa la información del workflow
- 3) Muestre.AR mapea la respuesta del JSON en objetos de tipo dtos propios del dominio de Muestre.AR
- 4) Muestre.AR (en el método gerWorkflow())instancia la clase de tipo Workflow correspondiente al framework Samplers junto con la lista de pasos a partir de los dtos mapeados del paso anterior. Si el usuario es un usuario registrado, se agrega el identificador de usuario a la lista de parametros configurables
- 5) Muestre.AR cede el control del flujo al framework Samplers
- 6) Samplers se encarga de visualizar el workflow correspondiente
- 7) Una vez que se completo el workflow, desde Samplers se realiza una llamada a la Api para persistir los resultados
- 8) Samplers devuelve el control del flujo a Muestre.AR móvil

Implementación de Muestre.AR Móvil

Como se puede ver en la Figura 3.14, Muestre.AR Móvil cuenta con una clase principal, que es la MainActivity.java. Esta es la clase encargada de instanciar la navegación lateral, manejar su comportamiento, e instanciar el fragmento que sea necesario en base a la actividad del usuario. Además esta clase es la encargada de actualizar la información mostrada en la barra de navegacion lateral, cuando el usuario se autentica en la aplicación; más adelante hablaremos con mayor profundidad sobre la navegación lateral. La

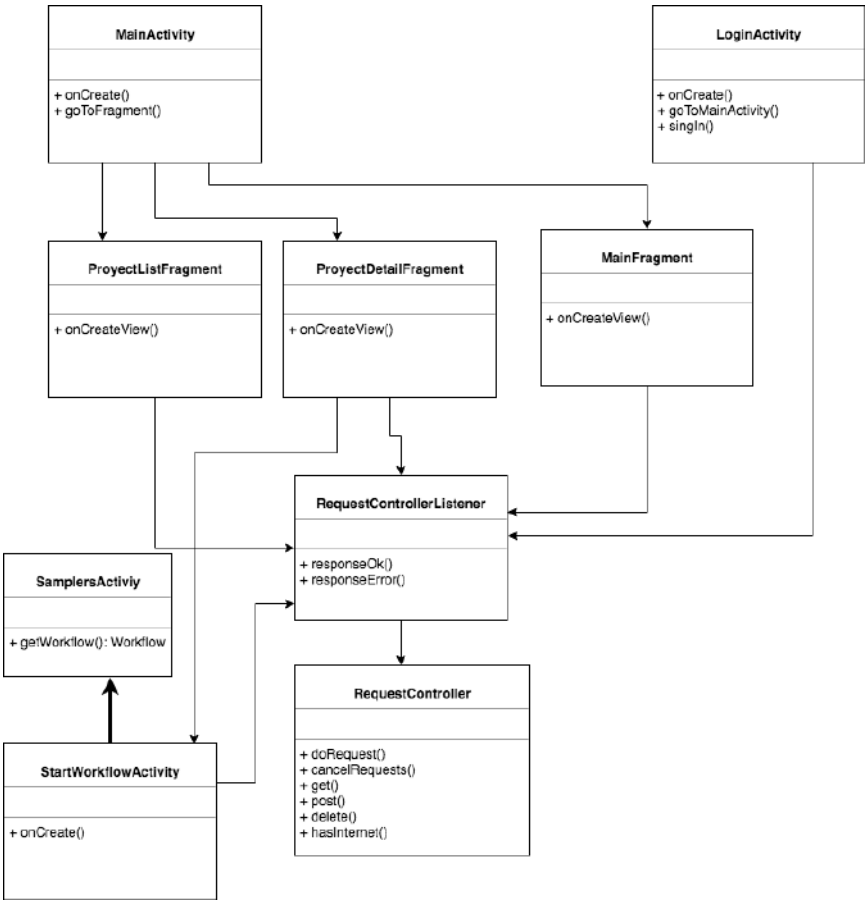

Figura 3.14: Diagrama de clases de la aplicación Android.

clase RequestController.java, la cual apoyada sobre la libreria volley, *librería desarrollada por Google para optimizar el envío de peticiones HTTP desde las aplicaciones Android hacia servidores externos*[27], es la encargada de realizar todas las peticiones HTTP que realiza Muestre.AR móvil. Cuenta con varios metodos que sirven para ejecutar los distintos métodos HTTP, como por ejemplo delete, post, get, patch. El mismo RequestController.java comprueba si se puede realizar dicha petición, verificando si el dispositivo dispone de conexión e informando a la clase correspondiente en caso de ocurrir un error. Al conocer el modelo de respuesta que va a recibir, el cual se infiere por la implementación de la interfaz RequestControllerListener, de la que hablaremos más adelante, y utilizando la libreria Gson, *es un API en Java, desarrollada por Google, que se utiliza para convertir objetos Java a JSON (serialización) y JSON a objetos Java (deserialización)*, mapea la respuesta de la API de tipo JSON a objetos que luego son pasados a los Fragmentos correspondientes. De existir algún error, el RequestController.java lo captura y se lo comunica al fragmento a traves del metodo responseError el cual es implementado por la interfaz RequestControllerListener.

En cuanto al mapeo de los datos, como se puede ver en la Figura 3.15, se utilizan clases que sirven de modelo; existe una clase principal, llamada ResponseDTO, la cual cuenta con los datos básicos de respuesta de una petición, como por ejemplo el status de la petición, un mensaje o los datos particulares, como se muestra en el siguiente extracto de código. Como también se puede ver en el siguiente fragment de código, ResponseDTO posee un método constructor, el cual es utilizado por la librería Gson comentada anteriormente para crear una nueva instancia de él.

```
public class ResponseDTO<T> {
    private String statusCode;
    private String message;
    private T data;
    private Integer errorCode;

    public ResponseDTO(String statusCode, String message, T data,
    Integer errorCode) {
        this.statusCode = statusCode;
        this.message = message;
        this.data = data;
        this.errorCode = errorCode;
    }
}
```

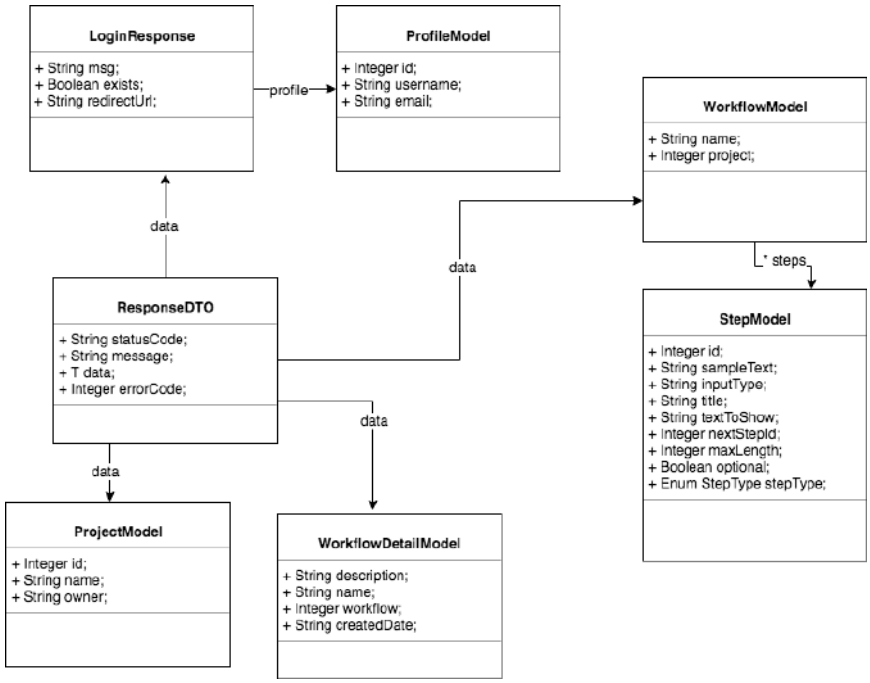

Figura 3.15: Diagrama de los modelos de datos utilizados en la aplicación Android.

Adicionalmente, se crea una clase especial para cada modelo de datos que se desea recibir, como por ejemplo la clase WorkflowDetailModel.java, la cual contiene la información que envia la API como respuesta a una petición en la cual se consultó por el detalle de un determinado proyecto. Como se observa a continuación, la clase tiene atributos como descripción, nombre y fecha de creación del proyecto consultado.

```
public class WorkflowDetailModel implements Serializable {

    private String description;
    private String name;
    private Integer workflow;
    private String createdDate;

    public WorkflowDetailModel(String description, String name,
    Integer workflow, String createdDate) {
        this.description = description;
        this.name = name;
        this.workflow = workflow;
        this.createdDate = createdDate;
    }
}
```

Se puede observar en el extracto anterior, la implementación del método contructor de la clase WorkflowDetailModel utilizado para mapear los datos recibidos desde el servidor.

La aplicacion Muestre.AR Móvil contiene 3 fragmentos:

MainFragment, el encargado de dibujar la pantalla inicial.

WorkflowsFragment, este es quien pinta el listado de proyectos existentes.

WorklowDetailfragment, dibuja el detalle de un proyecto.

Para realizar las peticiones HTTP, cada Fragmento, implementa la interface llamada RequestControllerListener. Este es de un tipo genérico, para poder especificar que modelo de datos vamos a recibir en cada una de las peticiones enviadas. Esta interfaz, brinda la posibilidad de abstraerse del modelo de datos en cada implementacion del requestController, solo basta con definir qué modelo de datos espera recibir. Hablaremos de estos modelos de datos más adelante. La interfaz está compuesta por 2 métodos como se muestra en el siguiente fragmento de código. Uno de ellos es **responseOk()**, el cual se ejecuta cuando la respuesta no tuvo ningún error y el **responseError()**, que se ejecuta cuando sí se produce uno.

```
public interface RequestControllerListener<T> {
```

```
    void responseOk(Integer id, ResponseDTO<T> response);
    void responseError(Integer id, ResponseDTO<T> response);
}
```

Cada fragmento que necesita hacer una petición instancia un RequestController pasando como parámetro su contexto y la referencia de sí mismo, a través de la implementacion del RequestControllerListener, del cual hablaremos a continuacion, para poder ser notificado con el resultado de la petición que realizó.

A diferencia del resto de las vistas, la de logueo no está representada a través de un fragmento sino de una actividad. Esto se debe al uso de las APIs tanto la de Google como la de Facebook para la autenticación del usuario, donde la documentación de estas recomiendan utilizarlas de esa forma. La actividad que se encarga de la lógica para la autenticación de un usuario es la llamada LoginActivity.java, la cual interactua con las APIs anteriormente mencionadas y con la de Muestre.AR, para poder realizar la operación.

Las clases LoginActivity.java y StartWorkflowActivity.java, implementan la interfaz RequestControllerListener y utilizan el RequestController ya que realizan peticiones HTTP para consultar a la API del proveedor correspondiente (ya sea Facebook o Gmail) y también a la de Muestre.AR para obtener el workflow para poder tomar la muestra.

3.5.3. Interfaz gráfica

Pantallas

La aplicación cuenta con 4 pantallas, más las que están destinadas a la recolección de una muestra. Una de las 4 pantallas es la pantalla inicial, la cual tiene la función de dar la bienvenida al usuario a la aplicación. En esta podemos ver el logo de Muestre.AR. Se puede ver esta pantalla en la figura 3.16.

Como hablamos antes en este capítulo, la forma de interactuar entre las pantallas es a través de una navegación lateral. Esta navegación es la recomendada por los estándares de experiencia de usuario definidos [28] y cuenta con 3 secciones. Como se puede ver en la figura 3.17, la primer sección es *Inicio*, la cual es representada por la pantalla inicial, mencionada previamente.

La segunda sección en la navegación de Muestre.AR móvil es *Proyectos*, y se representa a través de otra de las pantallas de la aplicación. Esta pantalla contiene un listado de los proyectos activos para los cuales un usuario ciudadano puede recolectar una muestra. Este listado es el resultado de una

Figura 3.16: Pantalla inicial

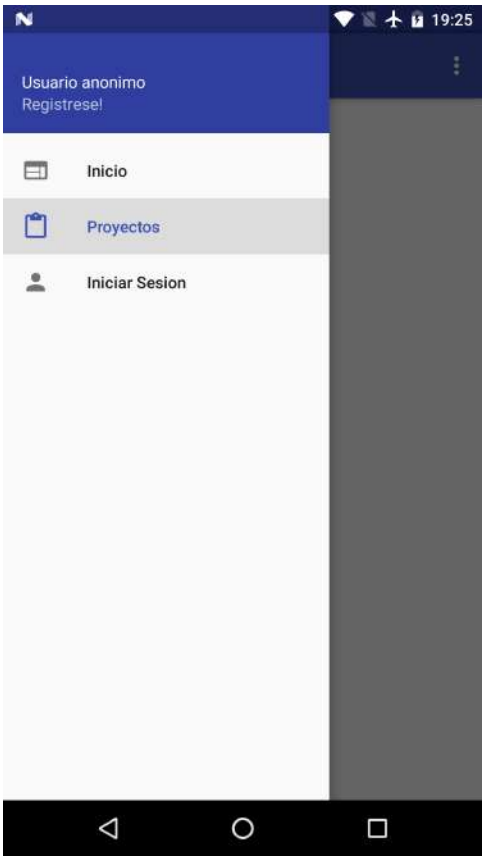

Figura 3.17: Navegación lateral

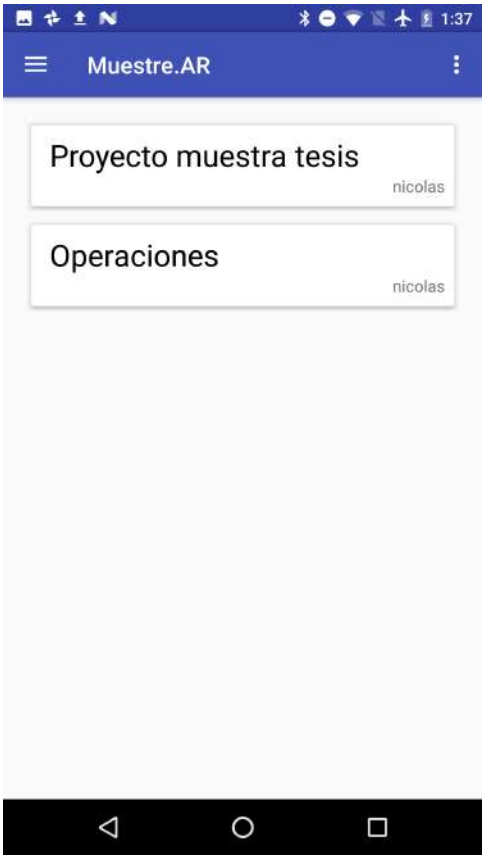

Figura 3.18: Listado de proyectos

petición HTTP hacia la API de Muestre.AR, es decir, se obtienen bajo demanda. Como muestra la figura 3.18 cada proyecto del listado muestra su nombre y la fecha de su creación. Si el proyecto es del interés del usuario ciudadano, este puede ver informacion detallada presionando sobre el proyecto. Este evento determina que se muestre otra de las pantallas, la del detalle de un proyecto. En la pantalla del detalle de un proyecto, se puede observar el nombre, una descripción, fecha de creación y un botón con el texto "Participar", tal como lo muestra la figura 3.19.

Cuando el botón antes mencionado es clickeado se realiza una nueva petición HTTP, donde son solicitados los pasos que componen el workflow asociado al proyecto para recolectar una muestra. Estos pasos son interpretados por Muestre.AR móvil en la clase StartWorkflowActivity.java como

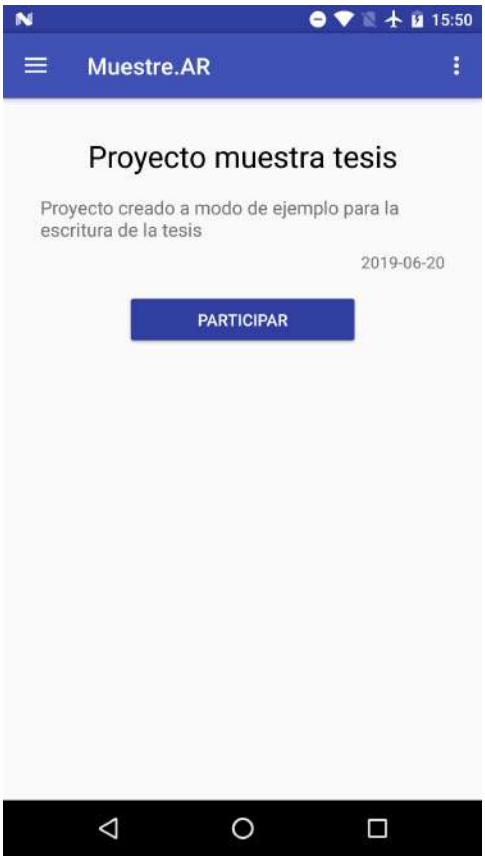

Figura 3.19: Pantalla de detalle de un proyecto

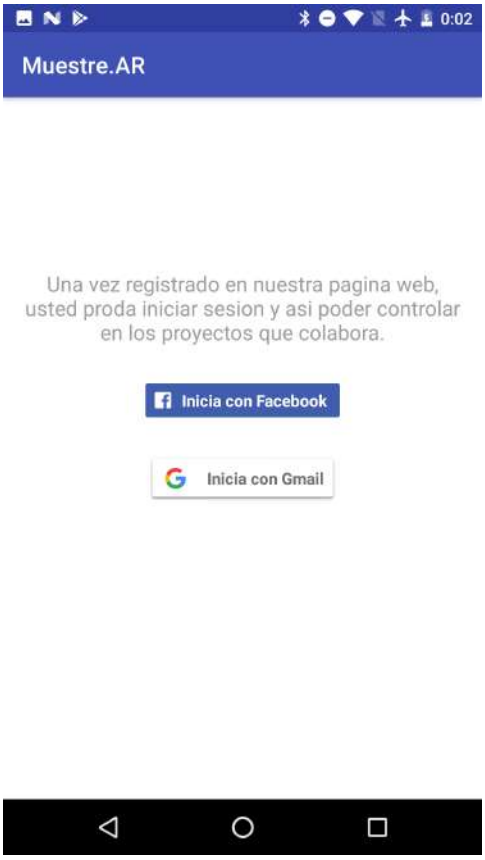

Figura 3.20: Pantalla de autenticación

fue detallado anteriormente y permiten al usuario ciudadano a recolectar la muestra.

La tercer sección de la aplicación es la de autenticación, la cual es representada por la pantalla de mismo nombre y es la que se muestra en la figura3.20. A través de esta pantalla un usuario puede autenticarse en Muestre.AR móvil para mantener un registro de todos sus recolecciones. En caso de estar autenticado, la navegación lateral contiene el nombre y el correo electrónico del usuario en cuestión y la tercer opción en la navegacion sirve para desautenticarse. La autenticación de un usuario va a ser exitosa, sólo si éste ya se registró previamente en la aplicación web de Muestre.AR. De no ser así, será notificado e invitado a dirigirse al sitio web para registrarse. En caso de que la autentificación fuera exitosa, el usuario es llevado a la pantalla

inicial y sus datos son reflejados en la navegación lateral, y cambiando la última opción para poder desautenticarse.

Un problema presentado en la implementación fue permitir que un usuario se autentique en dos aplicaciones distintas. Esto quiere decir que dado un usuario que se registre en la aplicación web y haya vinculado su cuenta con Facebook, luego pueda autenticarse desde la aplicación móvil a través de Facebook y sea el mismo usuario en ambos contextos. Para lograr esto, se tuvieron que seguir dos estrategias distintas para cada caso.

Para Facebook tuvimos que utilizar un atributo adicional llamado "token for business". El cual permite relacionar usuarios entre 2 aplicaciones distintas.

Para Gmail, en cambio, al compartir el mismo identificador entre dos aplicaciones distintas, no fue necesario hacer una implementación adicional.

Cada vez que una nueva pantalla es mostrada y esta requiere hacer una petición a la API, una imagen con un efecto que hace que esta gire es mostrada para comunicar al usuario que se está esperando la información. Una vez que esta se encuentra disponible, la imagen es retirada y se muestran los datos solicitados. Si en la petición comentada anteriormente sucede un error, la pantalla que la realizó es notificada y lo comunica al usuario, de forma que éste siempre este informado de lo que está sucediendo mientras utiliza la aplicación como lo muestra la figura 3.21.

Pantallas para la recolección

Entre las posibles pantallas destinadas a la recolección, que dependerán de cómo el usuario científico diagramó el workflow se encuentran:

El objetivo de cada una, los campos que tiene, los posibles valores para cada campo,

TextStep

Como se muestra en la Figura 3.22 esta pantalla está compuesta por un texto de muestra, el cual se puede utilizar para indicar al usuario la acción que debe realizar en ese paso de la obtención de la muestra y por un campo para ingresar un texto o un número, el cual es configurado por el usuario científico que diseña el workflow. El resultado de esta pantalla para la muestra puede ser un texto o un numero dependiendo de la configuración.

InformationStep

Esta pantalla tiene como objetivo darle información al usuario ciudadano. Está compuesta por un texto, el cual es cargado por el usuario cientifico que diseñe la muestra. En la Figura 3.23 se puede ver un ejemplo con

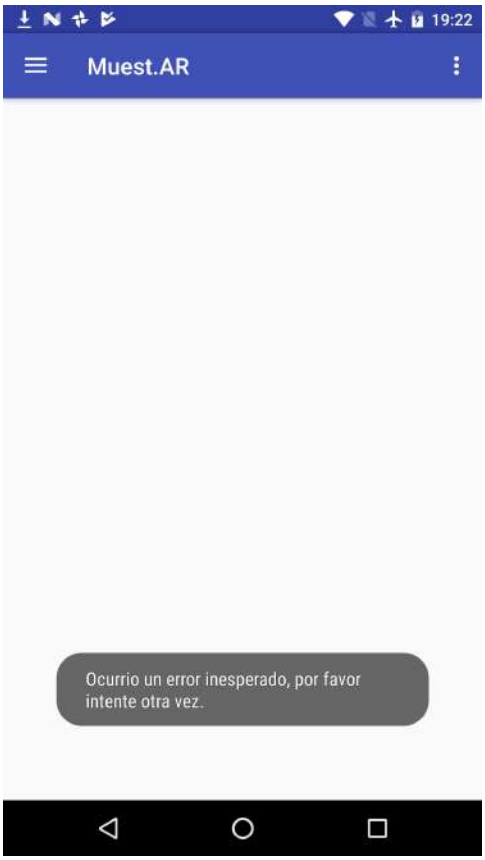

Figura 3.21: Notificación de error al usuario.

Figura 3.22: Pantalla TextStep

Figura 3.23: Pantalla InformationStep

un texto cargado a modo de ejemplo. Esta pantalla tiene como objetivo informar al usuario.

PhotoStep

En la Figura 3.24 se puede observar la pantalla encargada de capturar una foto para la recolección de la muestra. Sobre la pantalla se puede observar un texto que es configurado por el usuario cientifico que diseñó el workflow para indicar al usuario ciudadano qué debe realizar en esa instancia. El objetivo de esta pantalla es la captura de una fotografía.

LocationStep

Como se puede ver en la figura 3.25, esta pantalla está conpuesta por un mapa, un botón y al igual que las pantallas anteriores un texto para informar al usuario ciudadano. El botón de esta pantalla tiene como funcionalidad

Figura 3.24: Pantalla PhotoStep

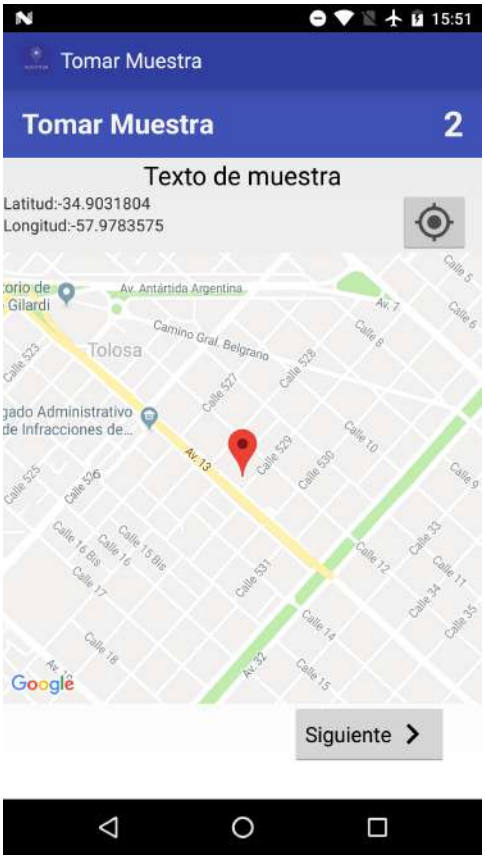

Figura 3.25: Pantalla LocationStep

tomar la ubicación del usuario de la aplicación en ese momento y marcarlo en el mapa. El objetivo principal de esta pantalla es brindarle al usuario cientifico que diseñó el workflow las coordenadas donde el usuario ciudadano está recolectando la muestra. El resultado que aporta esta pantalla a la muestra son las coordenadas geograficas del usuario que la recolectó.

SelectOneStep

Este paso tiene como objetivo que el usuario elija entre 2 o más opciones en base a una pregunta realizada y a partir de esta elección, se siga por un flujo de la recolección o por otro. Por ejemplo, en la figura 3.26 el usuario debe elegir entre las opciones **Muestra 1** y **Muestra 2** en respuesta a la pregunta **Texto de muestra**. Si el usuario selecciona la opción **Muestra 1**, la siguiente pantalla será la LocationStep. En cambio, si selecciona la otra,

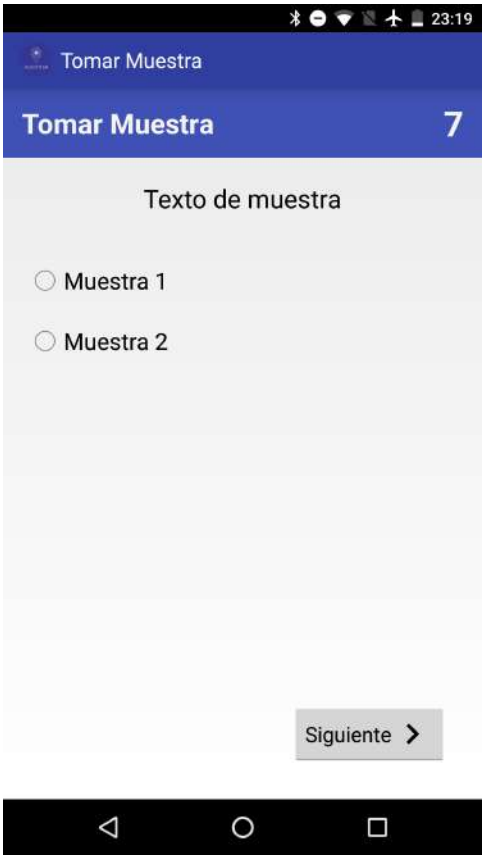

Figura 3.26: Pantalla SelectOneStep

la siguiente pantalla será PhotoStep. El gráfico 3.27 refleja la situación antes mencionada.

SelectMultipleStep

En esta pantalla el usuario que está recolectando la muestra puede observar una pregunta para la cual tiene que seleccionar una o más opciones que se le presentan debajo. Como se ve en la Figura 3.28 el texto *"Texto ejemplo"* es la pregunta y *"Muestra 1,Muestra 2,Muestra 3,Muestra 4"* las opciones que configuró el usuario científico. En el caso de la figura 3.28, el usuario que está recolectando la muestra seleccionó las muestras 2 y 4. El objetivo de esta pantalla es hacer elegir al usuario por una o muchas opciones.

TimeStep

Figura 3.27: Flujo SelectOneStep

Figura 3.28: Pantalla SelectMultipleStep

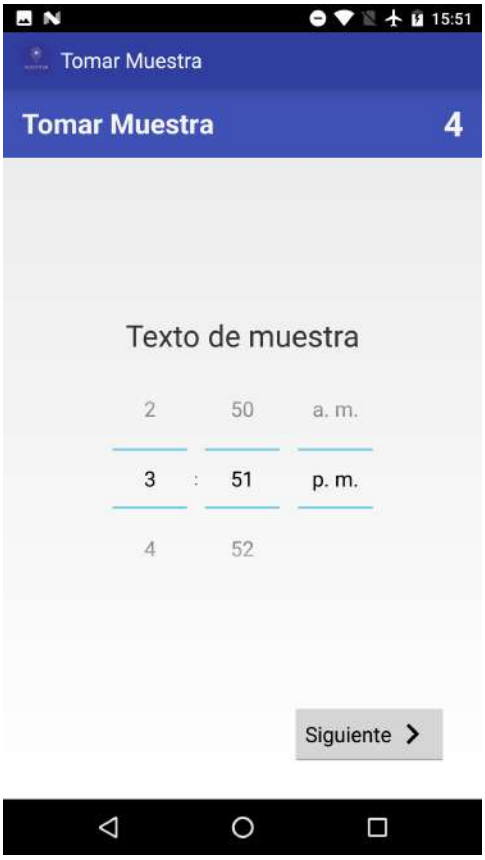

Figura 3.29: Pantalla TimeStep

La Figura 3.29 representa esta pantalla y se puede ver que al igual que la mayoría de las pantallas destinadas a la recolección, tiene un texto configurable por quien diseñó el workflow y además contiene un campo que permite la selección de la hora, los minutos y si es antes o después del mediodía. El resultado de esta etapa de la muestra es una hora.

DateStep

Esta pantalla está compuesta por el mismo texto que las demás y por un calendario para seleccionar una fecha. La Figura 3.30 muestra un ejemplo de la misma. El resultado de esta etapa de la muestra es una fecha.

SoundRecordStep

En la Figura 3.31 se puede observar que esta pantalla está compuesta por un texto indicando el tiempo que se lleva grabando, 2 botones *parar* y

Figura 3.30: Pantalla DateStep

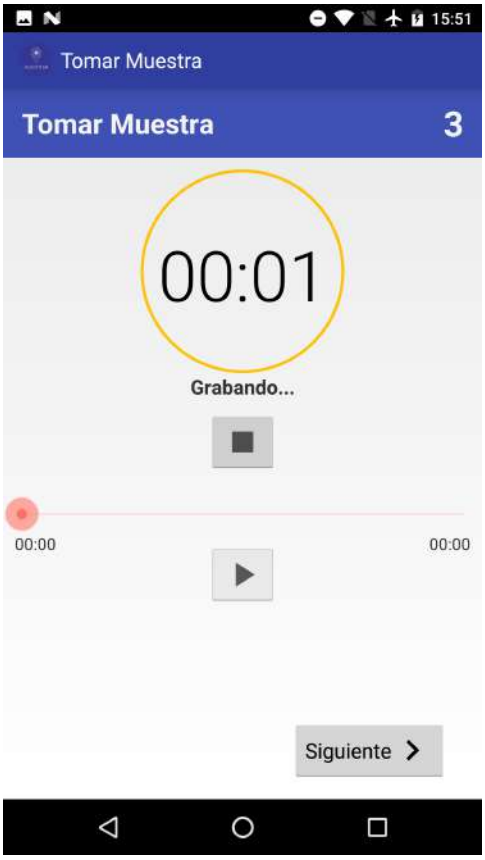

Figura 3.31: Pantalla SoundRecordStep

comenzar y una linea para representar el tiempo de grabación. Se obtiene como resultado en esta pantalla una grabación.

Cuando el workflow es finalizada, es decir, la muestra fue recolectada, el usuario es redireccionado a la pantalla de los *Proyectos* y el resultado de todos las pantallas es agrupado y encolado para su posterior envio al servidor. La muestra que fue encolada será enviada al servidor, asociado el usuario que la recolectó dicha muestra, solo si este estuvo autentificado al momento de la recolección.

Capítulo 4

Trabajos relacionados

Este capítulo está destinado a investigar, probar y comparar otras aplicaciones comprendidas dentro del concepto de ciencia ciudadana para poder observar ventajas, desventajas, puntos de mejoras y carencias de Muestre.AR. En primera instancia, se describirá a grandes rasgos cada una de las aplicaciones, comparandolas con respecto a las plataformas en las que funciona, si es necesario estar registrado para poder utilizar la plataforma, si cuenta con gamificación, si es de código abierto, entre otras características. Al final de la sección se presentará una tabla comparativa con los aspectos principales que se quieren resaltar de la comparación de todas las aplicaciones mencionadas.

4.1. App-ear

- Sitio web: http://www.app-ear.com.ar/

App-ear es un proyecto argentino de ciencia ciudadana, el cual está destinado al cuidado de los ambientes acuáticos. Tiene como funcionalidad básica la toma de muestras de los ambientes acuáticos a través de una aplicación móvil o página web, con la cual determinan su posición, y adicionalmente se brinda información acerca de ese ambiente acuático. Como comentamos anteriormente está compuesto por una página web y aplicación Android ,y consiste básicamente en que un usuario ciudadano, recolecte un conjunto de muestras, a traves de un conjunto de pasos definidos que son:

- Posicionarse en la ribera de un ambiente acuático,
- Responder preguntas o tomar fotografías, y finalmente enviar los resultados.

A su vez el sitio proporciona a los usuarios, el estado en tiempo real de los ambientes acuáticos de la nación. Como algunas de sus características, podemos decir que las muestras se pueden tomar desde una página web o desde una aplicación Android y para poder utilizar el sistema, es necesario estar logueado. Otra característica importante es que el código de la aplicación android es abierto. En la Figura 4.1 puede observarse la secuencia de pasos para obterner una muestra. Estas muestras son enviadas para luego procesarse y así brindar información de los ambientes acuáticos a sus usuarios.

App-ear cuenta con un sistema de *gamificación*, técnica de aprendizaje que traslada la mecánica de los juegos al ámbito educativo-profesional con el fin de conseguir mejores resultados, el cual recompensa al usuario con puntos con cada muestra recolectada, lo cual es un incentivo para el usuario para utilizar la aplicación. La funcionalidad de App-ear podría ser implementada en partes con Muestre.AR. Es decir, un científico puede crear un proyecto llamado App-ear en Muestre.AR web, donde este tenga un protocolo de muestra, que incluya los mismos pasos que la aplicación de App-ear, y luego el científico podría visualizar los datos obtenidos en las muestras y realizar con estos un mapa con el estado de los ambientes acuáticos. Lo que no podría implementarse es la gamificación con la que cuenta App-ear. Otra característica similar es que tanto Muestre.AR como App-ear son de código abierto. Una diferencia importante entre ambas es que para tomar una muestra es necesario estar logueado, mientras que en con Muestre.AR no lo es.

4.2. iNaturalist

- Sitio web: https://www.inaturalist.org/

iNaturalist es un sistema de identificación de especies de colaboración colectiva y una herramienta de registro de ocurrencia de organismos. Puede ser usado para registrar observaciones propias, obtener ayuda con las identificaciones, colaborar con otros para recopilar este tipo de información para un propósito común, o acceder a los datos de observación recopilados por los usuarios.[29].

Los usuarios pueden recolectar una muestra, capturando una foto de la observación, también pueden agregar una especie,notas, día y horario de la muestra, ubicación, si el organismo estaba en cautividad/cultivado.

Para todas las muestras se respeta el mismo protocolo, es decir, que los pasos para recolectar cualquier tipo de muestra son siempre igual. Esta

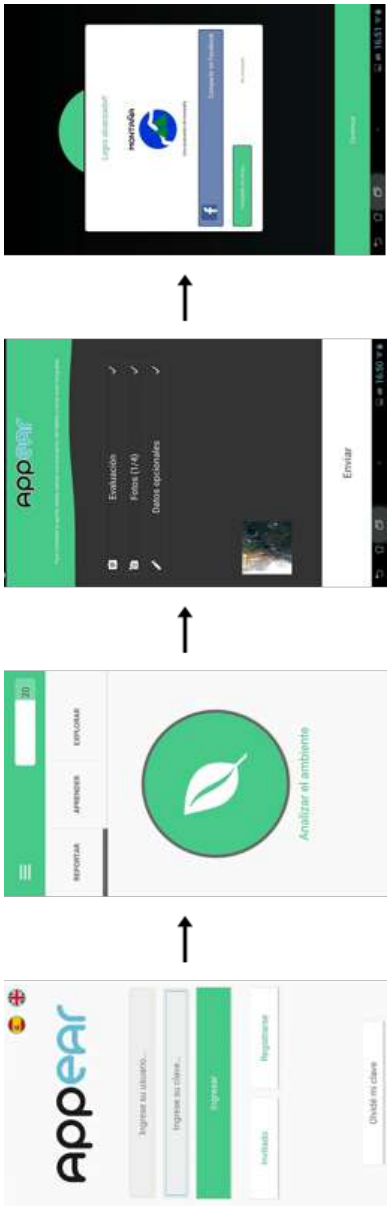

Figura 4.1: Secuencia de pasos básicos para recolectar una muestra con la aplicación App-ear

característica, demuestra una ventaja de Muestre.AR sobre iNaturalist, ya que este último no permite la modificación del protocolo por lo tanto todas las muestras que se quieran recolectar deben adaptarse a este. En cambio, Muestre.AR, permite definir el protocolo a medida para cada cosa en particular. Las muestras recolectas son compartidas con los usuarios profesionales sirviendoles a estos de material para sus estudios.

Al igual que App-ear, iNaturalist podría ser implementada mediante Muestre.AR parcialmente, ya que cuenta con una aplicacion iOS y una página web que permiten la toma de muestras, las cual no se podría implementar con Muestre.AR. Además permite comentarios sobre las observaciones que los usuarios profesionales realizan. Muestre.AR cuenta con la ventaja de poder descargar los resultados obtenidos y poder manipularlos como los profesionales quieran.

4.3. Natusfera

- Sitio web: https://natusfera.gbif.es/

Natusfera es una adaptación de iNaturalist, disponible para Android y iOS que permite a los usuarios subir resultados de muestras, para ser compartido con el resto de los usuarios. A su vez esto sirve a los científicos contribuyendo datos útiles sobre la distribución de especies. Al igual que en Muestre.AR existe el concepto de usuario científico, que son quienes participan o crean un proyecto. Tiene como objetivo educar al usuario, ya que si este no está interesado en su observación, no la comparte, no aprende. Este es un punto a favor en contraste con Muestre.AR. Otro punto a favor que tiene es que cuenta con una aplicación iOS para la recolección de muestras, lo cual no puede ser implementado con Muestre.AR. Una característica común que vemos entre Natusfera y Muestre.AR es que fueron desarrolladas con un propósito general, es decir no están abocadas a centrar las investigaciones o las muestras en un objetivo específico.[30] Como punto en contra, se podría decir que tanto Natusfera como iNaturalist no son de código abierto.

4.4. Caza Mosquitos

- Sitio web: http://www.ilpla.edu.ar/cazamosquitos.html

Es un proyecto desarrollado por investigadores del CONICET con el objetivo de conocer las distintas especies de mosquitos que habitan nuestro

país con ayuda ciudadana. Caza Mosquitos es el nombre de la aplicación, la cual está disponible para Android, con el objetivo de generar una base de datos de la distribución y ocurrencia de los mosquitos. Una vez que el usuario descarga la aplicación, debe registrarse para poder comenzar a recolectar muestras. El funcionamiento de la aplicación es muy sencillo: el usuario toma una fotografía de un mosquito, la carga en la aplicación reportando la presencia del insecto en su zona, y mediante diferentes diagramas y tablas comparativas puede diferenciar de qué especie se trata. Los datos enviados son recopilados y se establece el mapa con el detalle de las regiones que habita cada una. Para distinguir de qué especie se trata, el usuario recibe unas estructuras comparativas de la cabeza, tórax, abdomen y patas que muestran los diferentes patrones de coloración. El usuario puede recolectar 2 tipos de muestras, donde cada una tiene su secuencia de pasos específicos. Una de estas, es tomar una muestra de un mosquito, donde la secuencia de pasos a completar para obtener los resultados son:

- Seleccionar el tipo de mosquito examinado (la aplicación brinda la información necesaria para poder identificar la especie)
- Seleccionar la ubicación donde fue encontrado (seleccionando la misma en un mapa).
- Adjuntar fotos (hasta 4)
- Agregar notas y comentarios adicionales.
- Por último, los resultados son enviados.

En la Figura 4.2 puede observarse la secuencia de pasos para obterner una muestra de un mosquito.

El otro tipo de muestra que se puede realizar es de un criadero. Donde la secuencia de pasos para obtener estos resultados es distinto. En este caso consiste en:

- Contestar 3 preguntas
- Seleccionar la ubicación,
- Adjuntar notas y comentarios,
- Se pueden agregar fotos,
- Luego de esto, los resultados son enviados.

También cuenta con una parte educativa, donde brinda información acerca de los mosquitos en Argentina, los diferentes tipos que existen y sus características.

La aplicación tiene un sistema de ranking de puntos los cuales son obtenidos con el uso de la aplicación, enviando muestras por ejemplo, lo cual incentiva al usuario a utilizarla.

Esta aplicacion podría ser implementada con Muestre.AR de forma parcial ya que cuenta con el sistema de ranking antes mencionado, el cual no podría proveerse con Muestre.AR. A pesar de esto, cuenta con algunas desventajas con respecto a este, como por ejemplo que es necesario estar registrado para la toma de muestras, el protocolo de muestras no es personalizable, no es de código abierto. Es un proyecto creado con un propósito específico que fue comentado anteriormente.

4.5. GeoVin

- Sitio web: http://www.geovin.com.ar

GeoVin es un proyecto de ciencia ciudadana elaborado por investigadores del CEPAVE, cuyos objetivos principales son:

- Generar una base de datos de la distribución geográfica de las vinchucas.
- Prevenir sobre la proliferación de vinchucas.

GeoVin es una aplicación Android pública y gratuita, y su fin es orientar a todo tipo de usuarios en la identificación de posibles vinchucas que encuentren y que puedan implicar un riesgo epidemiológico. En base a los datos que se registren de localizaciones de las diferentes especies de vinchucas, la aplicación elabora mapas geográficos donde los usuarios pueden visibilizar sus hallazgos junto con los de otros usuarios.[31]

Tiene como funcionalidad básica la recolección de muestras de vinchucas a través de una aplicación móvil o página web, en la cual los usuarios determinan su posición, y adicionalmente brindan información acerca de una vinchuca, como fotos o el hábitat. Estas muestras son enviadas para luego procesarse y así brindar información acerca de la población de vinchucas en Argentina.

Para poder recolectar una muestra es necesario estar logueado. La recolección de una muestra consiste en:

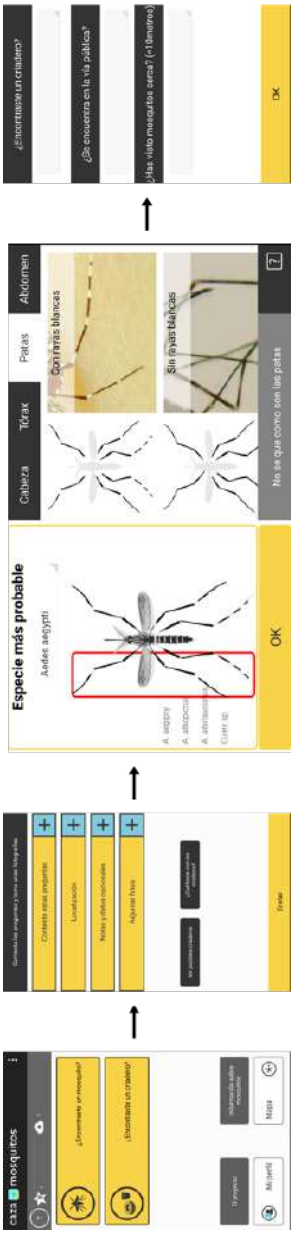

Figura 4.2: Secuencia de pasos básicos para recolectar una muestra de un mosquito con la aplicación Caza Mosquitos

- Seleccionar la ubicación (en un mapa).
- Capturar 2 fotos (de arriba, de abajo).
- Seleccionar el hábitat.

Esta aplicación podría ser implementada con Muestre.AR de forma parcial ya que cuenta con una página web que permite la recolección de muestras. A pesar de esto, cuenta con algunas desventajas con respecto a Muestre.AR, como por ejemplo que el protocolo de muestras no es personalizable y que no es de código abierto. Es un proyecto creado con un propósito específico que fue comentado anteriormente.

4.6. CitSci.org

- Sitio web: https://www.citsci.org

CitSci.org es un conjunto de aplicaciones que brinda soporte en la investigación a científicos, al proporcionarle herramientas y recursos que permiten personalizar un procedimiento científico, tales como: crear nuevos proyectos, administrar miembros del proyecto, crear hojas de datos personalizadas, analizar datos recopilados y recopilar comentarios de los participantes. Permite investigar preguntas científicas propias o sumarse como voluntario para un proyecto existente. [32] En comparación con Muestre.AR, se podría decir que en funcionalidad total, es el más similar. Comenzando con que esta destinado a ser multipropósito y multiproyecto, también existen 2 tipos de usuarios, entre otras características. Algunas ventajas de Muestre.AR es que cuenta con una aplicación móvil que facilita la recolección de la muestra y fundamentalmente, que es de código abierto. En contraparte, Citsci cuenta con un foro para debatir sobre algún proyecto y todos los usuarios registrados pueden ver las observaciones de otros proyectos. Una diferencia entre los 2, es que en Citsci primero se define un protocolo, y este es agregado a uno o más proyectos, mientras que en Muestre.AR cada proyecto tiene su propio protocolo. Pensamos que esta también es una ventaja de CitSci sobre Muestre.AR.

4.7. Tabla comparativa

En la tabla comparativa 4.1 que se encuentra a continuación, se comparan las aplicaciones previamente comentadas bajo las siguientes características:

Sistema	Auth	PP	iOS	Android	Web	Gamificación	Múltiple	Código abierto
App-ear	Si	No	No	Si	No	Si	Si	Si
Natiusfera	No	No	Si	Si	No	No	Si	No
iNaturalist	Si	No	Si	Si	Si	No	Si	No
CazaMosquitos	Si	No	No	Si	No	Si	Si	No
GeoVin	No	No	No	Si	Si	No	Si	No
CitSci	Si	No	No	No	Si	No	Si	No

Cuadro 4.1: Tabla comparativa a modo de resumen entre las distintas aplicaciones mencionadas en el capítulo

Explicación de los campos de la tabla

- Sistema: indica cual es el sistema a comparar.
- Auth: Si es necesario estar logueado para recolectar.
- PP: Indica si la aplicación permite personalizar el protocolo para recolectar una muestra.
- iOS: Este campo indica si el sistema cuenta con una aplicación iOS, mediante la cual permite recolectar muestras.
- Android: Similar al campo anterior pero para sistema operativo Android.
- Web: Igual a los 2 campos anteriores pero para sitio web.
- Gamificación: Si el sistema cuenta con algún tipo de gamificación para el incentivo de su uso.
- Múltiple: Indica si se puede recolectar muestras para multiples proyectos con la misma aplicación. (dudas de este campo, porque en natusfera e inaturist, se puede clasificar en proyectos las muestras obtenidas).
- Código abierto: Indica si el código de las aplicaciones es de código abierto.

Capítulo 5

Experiencias

La temática de este Capítulo es contar las distintas experiencias que tuvimos con Muestre.AR, donde en la primer sección comentaremos la presentación en el CIACIAR 18 y luego, en la segunda sección definiremos con detalle cómo a través de Muestre.AR puede ser implementada en su totalidad otra aplicacíon destinada a la recolección de muestras.

5.1. CIACIAR 18

El CIACIAR o Congreso de Ciencia Abierta y Ciudadana en Argentina, es un espacio de encuentro para científicos, investigadores, entusiastas de la ciencia, programadores, funcionarios de Ciencia y Tecnología y áreas afines, estudiantes universitarios y divulgadores de la ciencia y la tecnología. En la edición realizada en 2018, la cual se desarrolló en la Universidad Nacional de San Martín el día 2 de Noviembre, presentamos una versión de *Muestre.AR* cuya implementación no estaba finalizada. Esta versión, no contaba con los tipos de paso de audio y localización. Además la interfaz gráfica de las aplicaciones no tenían ningún tipo de estilo, por lo cual no era muy atractivo visualmente.

A pesar de las limitaciones que comentamos, fue posible presentar el proyecto con su funcionalidad total, ya que era posible configurar un workflow y luego recolectar una muestra para el mismo

El evento nos permitió dar a conocer a Muestre.AR a la comunidad, donde expusimos a través de charlas el funcionamiento de este, detalles sobre su utlización y posibles usos. Luego de esto, contestamos preguntas de los usuarios y una vez finalizado, presentamos la parte práctica donde los usuarios pudieron utilizar el sistema.

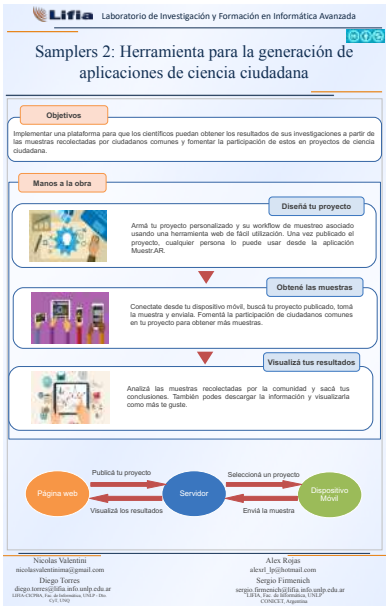

Figura 5.1: Poster presentado en CIACIAR 2018.

Para esta presentación, utilizamos el poster de la Figura 5.1

Durante esta experiencia intentamos que los usuarios puedan familiarizarse con el sistema, que puedan definir un protocolo y luego recolectar una muestra para el mismo. A pesar de contener limitaciones sobre todo desde lo visual, fue de mucha ayuda ya que nos permitió comprobar que es un tema de interés general, y que la herramienta tiene utilidad en el ámbito académico. Tal es así, que muchas personas nos pidieron un dato de contacto para seguir los avances del proyecto.

5.2. Implementación de Tiburones con Muestre.AR

5.2.1. Introducción

Para poner a prueba Muestre.AR con un ejemplo real y concreto, optamos por imitar una aplicación destinada a la recolección de muestras llamada "Tiburones", la cual está destinada a ser el medio por el cual un usuario recolecte una muestra. Esta muestra contiene información sobre especies acuáticas, ya sean tiburones, rayas o bien restos de animales. A pesar de que la aplicación a simular nunca fue publicada en el *Play Store*, consideramos

que es un buen ejemplo para poner a prueba Muestre.AR.

En primer lugar, detallaremos todos los posibles flujos que tiene la aplicación a imitar para luego crear un proyecto nuevo en Muestre.AR con este fin. A este nuevo proyecto se le configurará un workflow para que se terminen generando los mismos flujos pero en la aplicación Muestre.AR móvil. De esta forma podemos poner en práctica con un ejemplo real la utilidad del conjunto de aplicaciones que componen Muestre.AR.

5.2.2. Flujos

La primer pantalla, está compuesta por un texto donde se informa al usuario que recolecta la muestra que siga las instrucciones que se presentarán a lo largo de la recolección. La siguiente pantalla contiene una pregunta, la cual está destinada a que el usuario ingrese sobre qué tipo de especie acuática está relacionada la muestra. Existen 3 posibles alternativas, dependiendo de la elección de alguna de estas el flujo de la recolección será distinto. La posibilidades son:

- **Tiburones.**
- **Rayas.**
- **Restos/Huevos.**

Si la selección por parte del usuario es *Tiburones*, la siguiente pantalla está destinada a sacar una foto del espécimen avistado. Al continuar, la cuarta pantalla, presenta nuevamente una pregunta. En este caso, se le pregunta al usuario si desea tomar una foto de la cabeza del tiburón. Si el usuario selecciona que sí, la siguiente pantalla abre nuevamente la cámara del dispositivo móvil para que el usuario tome la foto. Si selecciona que no quiere la foto, la pantalla siguiente será la misma que se muestra al usuario luego de tomar la segunda foto. En esta pantalla se le pregunta al usuario cuanto mide el espécimen, expresado en centímetros. Una vez que el usuario ingresa estos datos, pasa a la última pantalla de todas, que sirve para saber la ubicación del usuario que recolecta la muestra. Luego de este paso, la muestra será recolectada completamente.

Por otro lado, si la selección del usuario que recolecta la muestra en vez de ser Tiburones, es *Rayas*, la pantalla siguiente a la selección es también para tomar una fotografía, luego se pide ingresar el largo del especimen y por ultimo también selecciona su ubicación.

Figura 5.2: Leyenda *Seguí las instrucciones* en segundo paso del modal de configuración del workflow

Por último si la elección es *Restos/Huevos*, el usuario que recolecta la muestra primero debe tomar una foto y luego seleccionar entre 4 posibilidades. Las posibilidades del usuario recolector son:

- **Restos de tiburones.**
- **Restos de raya.**
- **Huevos de rayas.**
- **Huevos de tiburones.**

Después de cada uno de estas pantallas independientemente de cual sea la decisión del usuario, la siguiente pantalla es la de seleccionar la ubicación.

5.2.3. Creación del proyecto

Para la implementación de la aplicación anteriormente detallada con Muestre.AR, como primer paso debemos acceder a Muestre.AR web y seleccionar la operación *Nuevo proyecto*. Después de esto, completar los datos solicitados, los cuales para el ejemplo no tienen importancia. Una vez creado el proyecto, seleccionamos la operación *Crear workflow*, para comenzar a configurar la recolección de la muestra.

5.2.4. Configuración del workflow

Como paso inicial, dejaremos el paso por defecto que ofrece el dashboard cuando se configura un workflow nuevo. De todas formas debemos configurarlo de forma adecuada. Para ésto, en el primer paso del modal de configuración mantenemos el tipo información como mencionamos anteriormente y en el segundo paso, en texto a mostrar, lo completamos con la leyenda *Seguí las instrucciones* como muestra la figura 5.2.

Figura 5.3: Configuración de opción *Tiburones*

A continuación, se agrega un paso de decisión, y se configura como detallaremos a continuación. En el segundo paso del modal de configuración, se agrega el texto *Selecciona sobre que va a ser la muestra* y se guarda. A este paso se le agregan 3 pasos siguientes posibles, donde cada uno de ellos tiene una opción asociada. Estas opciones son configuradas con los textos *Tiburones, Raysas* y la última *Restos/huevos*. En la figura 5.3 se puede ver como se configura la primera de las opciones.

A partir de este paso de decisión, se desprenden 3 ramificaciones, cada una de ellas asociada a una de las opciones configuradas. La ramificación relacionada con la opción *Tiburones*, comienza con un paso de tipo foto, donde este tiene configurado el texto *Tomar una foto del cuerpo entero del tiburon*. A este paso se le agrega como siguiente otro paso de decisión, configurado con *Selecciona una opcion* como texto a mostrar. En este caso, este paso de decisión tendrá 2 posibles opciones. La primera de estas opciones está configurada con el texto *Tomar foto de la cabeza del especimen*, y si es seleccionada, tendrá como paso siguiente uno de foto. Este paso de foto tendrá como texto a mostrar *Toma una foto de la cabeza del especimen*. Luego, el siguiente paso a este será el mismo que si se selecciona la opción *Omitir* en el paso de decisión anteriormente comentado. Para esto seleccionamos el paso de texto, presionamos la operación *Agregar padre* y luego seleccionamos el paso de decisión. La figura 5.4 muestra una configuración parcial del workflow en el cual se ve reflejado el flujo para recolectar una muestra relacionada a tiburones.

La figura 5.5 refleja la configuración para el paso de texto anteriormente nombrado. En esta configuración se demuestra que el texto a mostrar será *¿Que tamaño tiene el especimen?(en cm)*, el tipo de datos a ingresar por el usuario será un *Número*, el texto de muestra será una medida de ejemplo o vacío, *No será opcional* y como largo máximo podrán ser 4 o 5 dígitos.

El último paso, es el paso de localización, el cual viene a continuación del de texto. En la configuración de éste se debe poner como texto a mostrar *Selecciona tu ubicación*.

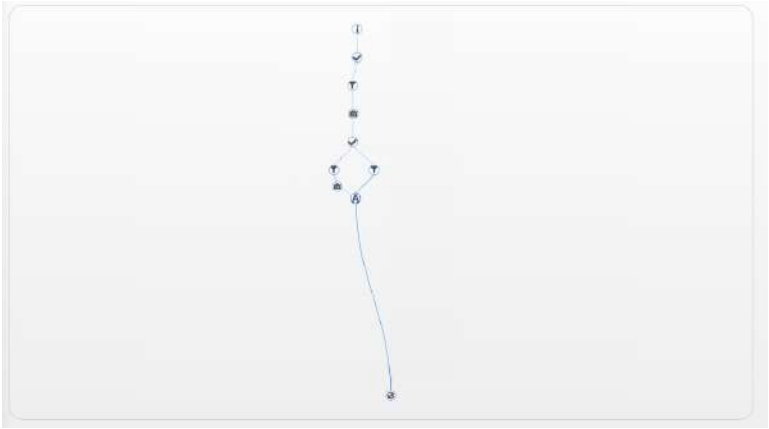

Figura 5.4: Workflow parcialmente configurado, solo con la ramificación *Tiburón*

Figura 5.5: Configuración de paso del tipo texto

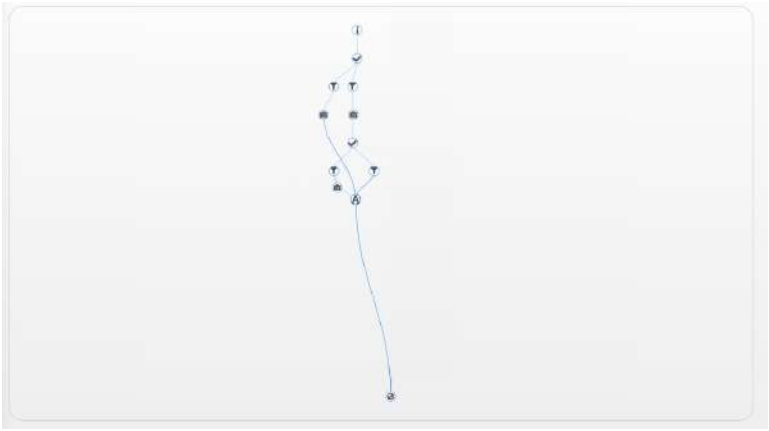

Figura 5.6: Configuración del workflow para tiburones y rayas

De la misma forma que está configurada la ramificación para la opción *Tiburones*, se configurará para la de la opción *Rayas*. Se agrega como paso siguiente uno de tipo foto, el cual tiene como texto a mostrar *Tomar una foto del espécimen*. Como el flujo a partir de ahora es ingresar un texto y compartir la ubicación, se puede reutilizar la configuración realizada para parte del flujo anterior. Se selecciona el nodo de texto configurado anteriormente y se ejecuta la operación *Agregar padre* y luego se selecciona el paso de tipo foto que queremos que sea el padre. De esta forma nos queda un workflow como el que refleja la figura 5.6.

Por último para la opción *Restos/Huevos*, el paso siguiente al de decisión es recolectar una muestra de tipo foto, con el texto a mostrar *Toma foto de los restos/huevos*. El paso siguiente es otro de tipo decisión y lo que tiene de particular esta decisión es que todas las opciones tienen como siguiente al mismo paso, el cual ya esta configurado y es el de localicación. Para realizar esta configuración se selecciona el paso de localización existente, se ejecuta la operación *Agregr padre* y se selecciona el paso de decisión que estamos configurando. De esta forma, se crea una opción la cual también debemos configurar. En total se ejecuta 4 veces la operación *Agregar padre* con las mismas características, ya que las 4 opciones tienen el mismo paso como siguiente. Para las 4 opciones nuevas, cada una es configurada con uno de los siguientes textos:

- **Restos de tiburones.**
- **Restos de raya.**

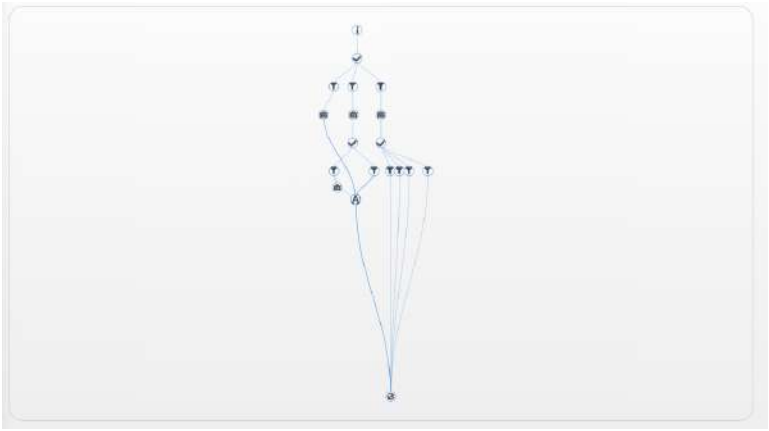

Figura 5.7: Configuración final del workflow

- **Huevos de rayas.**
- **Huevos de tiburones.**

El workflow resultante es el que refleja la figura 5.7.

Finalmente, utilizando Muestre.AR móvil, accedemos al proyecto *Tiburones* y seleccionamos *Participar*. A partir de ese momento, el usuario que recolecta la muestra tiene el mismo flujo de pantallas que si utilizara la aplicación de Tiburones. Las figuras 5.8 y 5.9 son algunas capturas donde se presenta el primer paso y el segundo respectivamente en Muestre.AR. A partir de que se recolecta la primer muestra, el usuario científico que creó el proyecto tiene acceso a los datos. De esta forma podemos asegurar que la aplicación tiburones puede ser implementada en su totalidad con Muestre.AR.

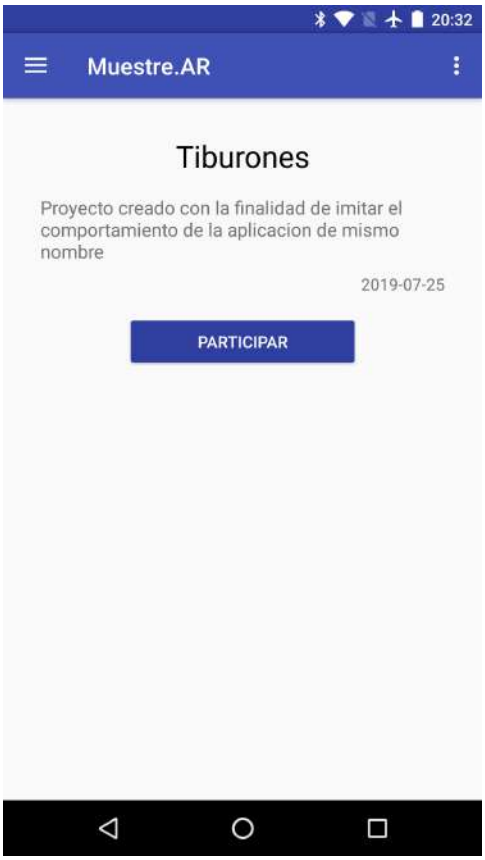

Figura 5.8: Pantalla de descripción del proyecto de tiburones en Muestre.AR

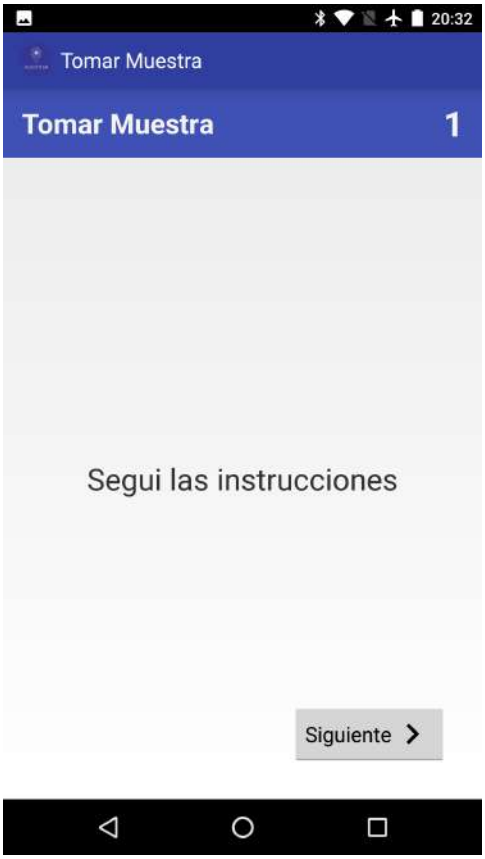

Figura 5.9: Primer pantalla de la recolección de tiburones en Muestre.AR

Capítulo 6

Conclusiones y mejoras futuras

Este Capítulo está destinado a volcar las conclusiones obtenidas en el desarrollo de esta tesina. En primer lugar hablaremos sobre los objetivos planteados, detallando si éste fue cumplido y cómo. Además, enumeraremos algunas mejoras futuras las cuales exceden esta tesina, pero entendemos que la podrían afectar positivamente.

6.1. Conclusiones

En el Capítulo 1 se definieron algunos objetivos, los cuales analizaremos si fueron cumplidos con la implementación realizada y de qué forma se obtuvieron. En primer lugar se propuso una aplicación web a través de la cual los usuarios científicos puedan crear una secuencia de pasos específicos y personalizados. Esto se ve solventado con el *Dashboard* el cual como detallamos anteriormente, es un componente de la aplicación web que sirve como herramienta para el diseño de una muestra. Esta herramienta brinda varias características que el método científico debería cumplir, como por ejemplo en la **objetividad** de este, ya que permite detallar cómo será la recolección y no acude a la interpretación del usuario que la recolecta. Además, permite que sea **riguroso** y de **orden lógico** ya que es quien diseña la recolección de la muestra el encargado de determinar el orden y las posibilidades en los pasos de la recolección. Esta característica antes planteada también permite una **experimentación controlada y sistemática**, ya que los pasos configurados por los científicos son interpretados por una aplicación Android y solo quien la diseñó podría modificarlos. Adicionalmente, se propuso una

aplicación Android donde los usuarios ciudadanos puedan recolectar una muestra. Como detallamos en Capítulos anteriores, Muestre.AR móvil es la aplicación que cumple con esta finalidad y está implementada de forma tal que permite que la recolección sea **fáctica**, recolectando datos reales como fotos y ubicaciones, **riguroso** y de **orden lógico** ya que no se puede alterar la configuración realizada por el científico para la muestra. Con la implementación de una API, los datos obtenidos en la recolección de las muestras se almacenan de forma **centralizada y digital**. Estos datos son brindados a los científicos a cargo del proyecto y permiten el **análisis y síntesis de los resultados**.

Por eso podemos decir, que Muestre.AR es un conjunto de herramientas, comunicadas entre sí, a traves de las cuales científicos pueden crear proyectos, diseñar la recolección de una muestra para estos y analizar los resultados. Además Muestre.AR móvil brinda el medio por el cual un usuario recolecta la muestra.

Como característica negativa de Muestre.AR móvil, pensamos que no brinda ningún beneficio al usuario ciudadano. Ya que estos recolectan las muestras pero no adquieren ningún conocimiento sobre el proyecto para el que colaboran, como sí lo tiene *Appear*, manteniendo un mapa con el estado de los ambientes acuáticos en su web. Tampoco tiene un contendido educativo como sí lo tiene por ejemplo *Caza Mosquitos* que fue comentada en la sección 4.4. Otras aplicaciones, cuentan con un sistema de *gamificación*, el cual permite motivar a los usuarios que sigan utilizando la aplicación. Muestre.AR no cuenta con ningún sistema de de este tipo.

6.2. Mejoras futuras

Para finalizar mencionaremos algunas mejoras que podrían implementarse en el futuro, pero que quedan por fuera del alcance de esta tesina. Una mejora que podría realizarse sobre la aplicación web es la creación de un *Foro*, donde los usuarios científicos que son invitados en un proyecto puedan iniciar un debate sobre éste. Para nuestra consideración, este desarrollo tiene un costo alto ya que deberían definirse varios aspectos como por ejemplo saber qué usuarios podrían participar de este, cómo se representaría graficamente, entre otros. Desde el lado de Muestre.AR móvil, se podría incluir una nueva sección destinada a que un usuario pueda ver un detalle de su perfil, pudiendo ver información de las muestras que recolectó y algunas estadísticas sobre éstas. Esta mejora no tendría un costo de implementación alto, ya que la aplicación cuenta con la sección implementada, pero no es accecible

ya que no encontramos datos relevantes que mostrar. Actualmente lo que no está implementado es la posibilidad de contar con los proyectos en los que colaboraró un usuario ciudadano. Otra mejora para Muestre.AR móvil es la utilización de la pantalla principal como portal de información. Esta mejora implica un costo de implementación alto ya que habría que diseñar como se mostraría la información, y además requeriría que ésta sea cargada. Otra mejora que a nuestro criterio se podría realizar es sobre la pantalla de dashboard para la creación de un workflow. Tanto el diseño del mismo como el apartado visual de la misma podrían ser mejor aprovechados, ya que actualmente el apartado visual es algo tosco. Por último, para darle más impulso a la aplicación móvil se podría incorporar un componente lúdico para fomentar la utilización de la aplicación por parte de los usuarios ciudadanos y recompensarlos por su participación. Existe una aplicación llamada Metagame que ofrece la posibilidad de agregar elementos de ludificación a cualquier proyecto de ciencia ciudadana. En un futuro Muestre.AR podría integrarse con ésta Como se mencionó a lo largo de la tesina, Muestre.AR móvil está desarrollado sobre el sistema operativo Android, por lo que sólo está disponible para aquellos celulares que posean ese sistema operativo. Por lo que los usuarios que tengan celulares con dispositivo IOS no van a poder hacer uso de muestre.AR móvil. Por lo que se podría proponer como mejora futura, implementar muestre.AR móvil para IOS. Por último, otra mejora que para nuestro entender aportaría mucho valor a Muestre.AR web y no es muy costosa de implementar es *Realizar gráficos con los resultados obtenidos*. Es decir que a partir de los datos obtenidos de las muestras recolectadas por los usuarios, los científicos a cargo del proyecto, puedan observar los mismo en forma de gráficos que estos mismos puedan personalizar. Decimos que es de fácil implementación ya que existen muchas librerías gratuitas que brindan esta funcionalidad. Pensamos que el desafío en esta funcionalidad está en la perzonalización de los gráficos y en como relacionar los datos entre los distintos pasos que contenga la muestra.

Bibliografía

[1] D. García Aristegui and C. Rendueles, "Abierto, libre... y público. los desafíos políticos de la ciencia abierta," *Argumentos de Razón Técnica, 17, 45-64.*, 2014.

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

[3] K. Firehock and J. West, "A brief history of volunteer biological water monitoring using macroinvertebrates," *Journal of the North American Benthological Society*, vol. 14, no. 1, pp. 197–202, 1995.

[4] D. Brossard, B. Lewenstein, and R. Bonney, "Scientific knowledge and attitude change: The impact of a citizen science project," *International Journal of Science Education*, vol. 27, no. 9, pp. 1099–1121, 2005.

[5] D. J. Trumbull, R. Bonney, D. Bascom, and A. Cabral, "Thinking scientifically during participation in a citizen-science project," *Science education*, vol. 84, no. 2, pp. 265–275, 2000.

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

[7] A. Cho and D. Clery, "Astronomy hits the big time," *Science*, vol. 323, no. 5912, pp. 332–335, 2009.

[8] "Cientópolis: Motorizando la ciencia ciudadana." https://digital.cic.gba.gob.ar/bitstream/handle/11746/6858/104-20LIFIA.pdf?sequence=1. Accessed: 2019-09-24.

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

[10] "Método Científico." https://www.caracteristicas.co/metodo-cientifico/. Accessed: 2019-09-30.

[11] "Ciencia Abierta: definición." http://www.ciecti.org.ar/wp-content/uploads/2016/09/CIECTI-Proyecto-CENIT.pdf. Accessed: 2019-08-13.

[12] "Ciencia Ciudadana: definición." https://www.gbif.es/wp-content/uploads/2017/12/02$_{C}onceptos_{d}e_{C}ienciaCiudadana.pdf.Accessed$: $2019-08-13$.

[13] R. P. A. Miller-Rushing and R. Bonney, *The history of pu- blic participation in ecological research.* Frontiers in Ecology and the Environment, vol. 10, no. 6, pp. 285–290, 2012.

[14] "Ciencia Ciudadana: ejes esenciales." https://blogs.iadb.org/conocimiento-abierto/es/la-ciencia-ciudadana-promueve-conocimiento-abierto/. Accessed: 2019-09-01.

[15] A. Wiggins and K. Crowston, *From conservation to crowdsourcing: A typology of citizen science.* Proceedings of the Annual Hawaii International Conference on System Sciences, pp. 1–10, 2011.

[16] C. C. et al Wilderman, *Models of community science: design lessons from the field.* Cornell Laboratory of Ornithology, 2007.

[17] R. J. E. M. T. P. J. S. R. Bonney, H. Ballard and C. C. Wilderman, *Models of community science: design lessons from the field.* Public Participation in Scientific Research: Defining the Field and Assessing Its Potential for Informal Science Education. A CAISE Inquiry Group Report, 2009.

[18] "Soporte de las tics." http://www.revistacts.net/files/Volumen$_{9}Numero_{2}7/FINALES/FinquelievichFINAL.pdf.Accessed$: $2019-07-05$.

[19] "Aplicación movil: ventajas." https://www.fundeu.es/recomendacion/aplicacion-alternativa-a-app/. Accessed: 2019-08-29.

[20] "Aplicacion móvil: funcionamiento." https://www.consumidor.ftc.gov/articulos/s0018-aplicaciones-moviles-que-son-y-como-funcionan. Accessed: 2019-10-14.

[21] "Android: características." https://webgenio.com/blog/que-es-android-y-que-es-un-telefono-movil-android/. Accessed: 2019-08-27.

[22] "Samplers: definición." https://www.cientopolis.org/samplers/. Accessed: 2019-08-20.

[23] "Django: características." https://developer.mozilla.org/es/docs/Learn/Server-side/Django/IntroducciC3B3n. Accessed: 2019-07-05.

[24] R. M. G. Labrador, "Tipos de licencias de software," 2012.

[25] "Python Social Auth." https://python-social-auth.readthedocs.io/en/latest/. Accessed: 2019-08-29.

[26] D. Robledo Fernández, *Desarrollo de aplicaciones para Android II*. Ministerio de Educación, 2014.

[27] Y. Shulin and H. Jieping, "Research and implementation of web services in android network communication framework volley," in *2014 11th International Conference on Service Systems and Service Management (ICSSSM)*, pp. 1–3, IEEE, 2014.

[28] G. Nudelman, *Android design patterns: interaction design solutions for developers*. John Wiley & Sons, 2013.

[29] F. Michonneau and G. Paulay, "Using inaturalist to learn more about echinoderms," *Reef Encounter*, vol. 30, pp. 29–31, 2015.

[30] J. Piera, L. Ceccaroni, and B. Claramunt, "Natusfera: a new platform to integrate citizen science approaches for monitoring marine ecosystems," 2016.

[31] A. Balsalobre, S. Ceccarelli, M. E. Cano, W. A. Ferrari, J. Cochero, and G. A. Martí, "Apps en el desarrollo de ciencia ciudadana: Geovin," in *I Jornadas de Inclusión de Tecnologías Digitales en la Educación Veterinaria (La Plata, 2018)*, 2018.

[32] S. Kocaman, B. Anbaroglu, C. Gokceoglu, and O. Altan, "A review on citizen science (citsci) applications for disaster management," *Int Arch Photog Rem Sens Spatial Inf Sci*, vol. 42, no. 3, p. W4, 2018.