



UNIVERSIDAD
NACIONAL
DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

TÍTULO: DEHIA: Una plataforma liviana para definir y ejecutar actividades...

AUTORES: Jose Arcidiacono

DIRECTOR: Patricia Bazán

CODIRECTOR: Alejandra B. Lliteras

ASESOR PROFESIONAL:

CARRERA: Licenciatura en Sistemas

Resumen

La recolección de datos por parte de personas ha ido evolucionando desde formularios en papel hasta las aplicaciones móviles de la actualidad. En este trabajo se propone y desarrolla una plataforma que permite a usuarios, sin conocimientos de programación, la creación de actividades de recolección de datos con intervención humana para ser ejecutadas desde una aplicación móvil.

Palabras Clave

Workflows; Actividades; Microservicios; Recolección de Datos, Plataforma Web, Aplicación Móvil. Arquitecturas Distribuidas

Conclusiones

En esta tesina se estudiaron y compararon plataformas de creación de actividades de recolección de datos con intervención humana. Se presentó una arquitectura para este tipo de plataformas y se desarrolló un prototipo llamado DEHIA, que fue instanciado en el proyecto E-Basura. Se llevaron a cabo pruebas de usuario que arrojaron resultados satisfactorios respecto a la usabilidad de la plataforma web.

Tres artículos en el marco de la temática de esta tesina fueron presentados y aprobados para su presentación.

Trabajos Realizados

Se realizó un análisis del estado del arte en cuanto a plataformas de creación de actividades de recolección de datos con intervención humana.

Se diseñó una arquitectura para una plataforma de este tipo. Se desarrolló un prototipo de la plataforma web siguiendo la arquitectura presentada que permite la creación de actividades y un prototipo de una aplicación móvil que permite ejecutarlas.

Se presentó un caso de estudio y pruebas con usuarios.

Tres artículos en el marco de la temática de esta tesina fueron presentados y aprobados para su presentación.

Trabajos Futuros

El prototipo puede extenderse agregando componentes definidos en la arquitectura diseñada, como un componente web para ejecución de actividades, o diseño colaborativo de planificaciones.

Existen mejoras posibles en cuanto a seguridad y escalabilidad de los microservicios entre otros aspectos detallados en el documento de esta tesina.

Universidad Nacional de La Plata
Facultad de Informática

Tesina de grado
Licenciatura en Sistemas

DEHIA: UNA PLATAFORMA LIVIANA
PARA DEFINIR Y EJECUTAR
ACTIVIDADES CON INTERVENCIÓN
HUMANA BASADAS EN WORKFLOWS

Autor: Jose Arcidiacono

Director: Patricia Bazán

Co-director: Alejandra B. Lliteras

Noviembre de 2020

CONTENIDO

Índice de figuras.....	5
Índice de Tablas.....	6
Capítulo 1 - Introducción.....	8
1.1 Motivación.....	8
1.2 Objetivos.....	8
1.3 Sobre la organización de la tesina	9
Capítulo 2 – Marco Teórico.....	10
2.1 Conceptos relacionados con el modelado de actividades.....	10
2.1.1. Tareas.....	10
2.1.2 Planificación	10
2.1.3. Workflow.....	11
2.2 Conceptos relacionados con la arquitectura	14
2.2.1. Sistemas distribuidos	14
2.2.2. HTTP.....	15
2.2.3. REST.....	18
2.2.4 Microservicios	19
2.3 Conceptos relacionados con el ámbito de aplicación de la plataforma propuesta.....	21
2.3.1 Aprendizaje Móvil.....	21
2.3.2. Recolección de datos.....	22
2.3.3. Ciencia Ciudadana.....	23
2.4 Recapitulación	24
Capítulo 3 - Estado del Arte.....	25
3.1 EpiCollect5	25
3.2 Zooniverse.....	29
3.3 Open Data Kit	32
3.4 MoLE.....	35
3.5 Comparación de herramientas.....	37
3.5.1 Herramienta de creación	38
3.5.2 Aplicación móvil	40
3.5.3 Servidor.....	41
3.6 Recapitulación	42
Capítulo 4. Arquitectura propuesta.....	43
4.1 Capas de la Arquitectura	43
4.1.1 Capa de Clientes.....	44
4.1.2 Capa de compuertas.....	45
4.1.3 Capa de Servicios.....	46

4.1.4 Capa de Persistencia.....	47
4.2 Componentes.....	48
4.2.1 Herramienta Web para la definición de workflows.....	48
4.2.2 Resolución web de actividades.....	50
4.2.3 Resultados web.....	51
4.2.4 Aplicación móvil.....	53
4.2.5 API REST.....	54
4.3 Recapitulación.....	54
Capítulo 5. La plataforma DEHIA.....	55
5.1 Alcance.....	55
5.2 Arquitectura.....	55
5.2.1 Capa de clientes.....	56
5.2.2 Capa de compuertas.....	56
5.2.3 Capa de Servicios.....	57
5.2.4 Capa de Persistencia.....	57
5.3 Componentes.....	57
5.3.1 Herramienta web.....	57
5.3.2 Aplicación móvil.....	65
5.2.3 API.....	68
5.4 Modelo de datos.....	69
5.5 Tecnologías utilizadas.....	70
5.5.1 Aplicación móvil.....	70
5.5.2 Aplicación web.....	70
5.5.3 Compuerta.....	70
5.5.4 Servicios.....	71
5.6 Recapitulación.....	71
Capítulo 6. Caso de estudio.....	72
6.1 Problemática a resolver.....	72
6.2 Solución existente.....	72
6.3 Instanciación del prototipo.....	73
6.4 Pruebas de usuario.....	80
6.5 Recapitulación.....	81
Capítulo 7. Conclusiones y trabajos futuros.....	82
7.1 Conclusiones.....	82
7.3 Contribuciones.....	83
7.2 Trabajos futuros.....	84
Referencias bibliográficas.....	86
Anexos.....	89

Anexo A - Protocolo de la API de DEHIA	89
Anexo B - Bibliotecas utilizadas para el prototipo	104
Servicios	104
Aplicación Móvil	105
Aplicación web	106
Compuerta.....	107
Anexo C - Casos de prueba	107
C.1 Test de Unidad	108
C.2 Test de Integración	109
C. 3 Test de Interfaz gráfica.....	110
Anexo D - Planificación del caso de estudio original.....	113
Nombre de la actividad: Entrega de elementos en desuso	113
Objetivo: Notificar a E-Basura de la entrega de equipamiento.....	113
Persona de contacto (1).....	113
Tipo de Donación (2)	113
Sobre el teléfono de contacto (3).....	113
Teléfono de contacto (4)	113
Correo electrónico de contacto (5) - OPCIONAL.....	114
Tipo de Institución (6)	114
Otra Institución (7).....	114
Nombre de la empresa (8).....	114
Provincia (9).....	114
Localidad (10)	114
Dirección Institucional (11).....	115
Teléfono Institucional (12) - OPCIONAL	115
Correo electrónico institucional (13)	115
Actividad (14).....	115
RSE (15)	115
Equipamiento I (16).....	115
Equipamiento II (17)	116
Otros elementos (18) - OPCIONAL.....	116
Foto elementos (19).....	116
Retiro (20).....	116
Ubicación retiro (21)	116
Muchas gracias (22).....	117
Anexo E - Materiales para las pruebas de usuario	117
Caso de uso Donación de Residuos de Aparatos Eléctricos y Electrónicos (RAEE) ..	117
Encuesta post-prueba.....	121

Anexo F - Resultados de las pruebas.....	124
Anexo G - Acerca de los archivos fuente.....	126

ÍNDICE DE FIGURAS

Figura 2.1.1. Posibles tipos de planificación para las tareas.....	11
Figura 2.1.2 Patrón de Secuencia ilustrado con una red de Petri coloreada [Kunze, et al., 2016].....	12
Figura 2.1.3 Patrón de Selección exclusiva ilustrado con una red de Petri coloreada [Kunze, et al., 2016].....	12
Figura 2.1.4 Patrón de fusión simple ilustrado con una red de Petri coloreada	13
Figura 2.1.5 Patrón de bifurcación en paralelo ilustrado con una red de Petri [Kunze, et al., 2016].....	13
Figura 2.1.6 Patrón de fusión de sincronismo ilustrado por una red de Petri coloreada [Kunze, et al., 2016]	14
Figura 2.2.1 Un sistema distribuido organizado como <i>middleware</i> [van Steen, et al., 2016]	15
Figura 2.2.2 Ejemplo de petición/respuesta HTTP [Gourley, et al., 2002]	16
Figura 2.2.3 Partes de una URL [Gourley, et al., 2002].....	17
Figura 2.2.4 Niveles de madurez de acuerdo al modelo de Richardson [Webber, et al., 2010].....	19
Figura 2.2.5 Coordinación orquestada [Newman, 2015]	21
Figura 2.2.6 Coordinación coreografiada [Newman, 2015].....	21
Figura 2.3.1 Google Forms	23
Figura 2.3.2 GeoVIN	23
Figura 2.3.3 Open Data Kit.....	23
Figura 2.3.4 Snapshot Serengeti, uno de los proyectos de Zooniverse	24
Figura 3.1.1 Actividad ejecutándose en EpiCollect [Aanensen, et al., 2009].....	25
Figura 3.1.2 Arquitectura de EpiCollect	26
Figura 3.1.3 Formulario jerárquico en Epicollect+ [Aanensen, et al., 2014].....	27
Figura 3.1.4 Arquitectura de Epicollect+	28
Figura 3.1.5 Herramienta de definición de formularios de EpiCollect5.....	28
Figura 3.1.6 Arquitectura de EpiCollect5.....	29
Figura 3.2.1 Tarea dentro del proyecto Galaxy Zoo	30
Figura 3.2.2 Herramienta de definición de workflows de Zooniverse	31
Figura 3.2.3 Arquitectura de Zooniverse	31
Figura 3.3.1 Planilla de cálculo en formato XLSForm	33
Figura 3.3.2 Editor de formularios de Open Data Kit.....	33
Figura 3.3.3 ODK Aggregate	34
Figura 3.3.4 Ejecución de una actividad en ODK Collect	34
Figura 3.3.5 Arquitectura de Open Data Kit.....	35
Figura 3.4.1 Herramienta de creación de actividades de MoLE.....	36
Figura 3.4.2 Actividad ejecutándose en Resuelvo Explorando [Dal Bianco et al., 2019]	36
Figura 3.4.3 Arquitectura de MoLE	37
Figura 3.5.2 Componentes básicos de estas herramientas [Hartung, et al., 2010].....	38
Figura 4.1.1 Arquitectura general propuesta.....	43
Figura 4.1.1.1 Ejemplos de clientes.....	44
Figura 4.1.2.1 Ejemplos de compuertas (en la fila central).....	46
Figura 4.1.4.1 Ejemplo de mecanismos de persistencia	47

Figura 4.2.1.1 Planificación condicional basada en respuestas.....	49
Figura 4.2.1.2 Planificación condicional basada en el recorrido.....	49
Figura 4.2.1.3 Ejemplo de condición externa en una actividad colaborativa	49
Figura 4.2.2.1 Ejemplo de pantalla con tareas para elegir.....	50
Figura 4.2.2.2 Ejemplo de pantalla con un flujo colaborativo	51
Figura 4.2.3.1 Vista de única entrada	52
Figura 4.2.3.2 Vista de mapa	52
Figura 4.2.4.1 Selección de actividad en la aplicación móvil	53
Figura 4.2.4.2 Tarea de elegir una opción en la aplicación móvil.....	53
Figura 5.2.1 Arquitectura general propuesta.....	55
Figura 5.2.1.1 Clientes en la plataforma DEHIA.....	56
Figura 5.2.2.1 Capa de compuertas en la plataforma DEHIA (fila central)	56
Figura 5.2.4.1 Capa de persistencia en la plataforma DEHIA (última fila).....	57
Figura 5.3.1.1 Pantalla de inicio de sesión.....	58
Figura 5.3.1.2 Menú de la herramienta web	58
Figura 5.3.1.3 Diagrama de estados de una actividad	59
Figura 5.3.1.4 Pantalla de creación de tareas (opción múltiple).....	59
Figura 5.3.1.5 Pantalla de selección de tareas	60
Figura 5.3.1.6 Pantalla de simple de selección de tareas opcionales	61
Figura 5.3.1.7 Editor de workflows.....	61
Figura 5.3.1.8 Menú para agregar conexión hacia una tarea sucesora.....	62
Figura 5.3.1.9 Pantalla de detalles de actividad	63
Figura 5.3.1.10 Pantalla de selección de tareas (clonando planificación).....	64
Figura 5.3.1.11 Conexiones condicionales pendientes de ajuste	64
Figura 5.3.1.12 Pantalla de resultados	65
Figura 5.3.2.1 Selección de actividad.....	66
Figura 5.3.2.2 Lista de tareas en una actividad de planificación libre	66
Figura 5.3.2.3 Pantalla de elegir una opción.....	67
Figura 5.3.2.4 Pantalla de indicar posición	67
Figura 5.3.2.5 Pantalla de grabar audio.....	67
Figura 5.3.2.6 Pantalla de envío de respuestas	68
Figura 5.4.1 Modelo de datos de la plataforma (relacional)	69
Figura 5.4.2 Modelo de datos de la plataforma (no relacional).....	70
Figura 6.2.1 Formulario original	73
Figura 6.3.1 Diseño de la actividad.....	73
Figura 6.3.2 Creación de la actividad.....	74
Figura 6.3.3 Pantalla de asignación de tareas	74
Figura 6.3.4 Creación de tarea tipo texto	75
Figura 6.3.5 Creación de tarea de opción múltiple	75
Figura 6.3.6 Listado de tareas de la actividad.....	76
Figura 6.3.7 Creación de conexión condicional.....	76
Figura 6.3.8 Planificación terminada.....	78
Figura 6.3.9 Pantalla para elegir el tipo de donación.....	79
Figura 6.3.10 Tarea de grabar audio.....	80
Figura 6.3.11 Tarea de sacar una foto	80
Figura 6.3.12 Tarea de localización.....	80

ÍNDICE DE TABLAS

Tabla 3.5.1.1 Comparación de tipos de tareas posibles.....	39
--	----

Tabla 3.5.2 Comparación de características de aplicaciones móviles.....	40
Tabla 3.5.3.1 Comparación de características de las APIs	41

CAPÍTULO 1 - INTRODUCCIÓN

1.1 MOTIVACIÓN

Esta tesina surge como respuesta a una iniciativa del proyecto “Recicla tu Compu-Recicla tu Mundo”¹ [LINTI, 2018], un Proyecto de Extensión de la Facultad de Informática. Ésta tiene como fin generar herramientas basadas en el uso de la tecnología para provocar en niños, jóvenes y adultos la apropiación de los conocimientos fundamentales que ayuden en los procesos educativos y de concientización en la temática abordada en el proyecto.

Para dar respuesta a esta iniciativa se tomaron como base los resultados de dos proyectos de investigación ([Lliteras, 2015], [Lliteras et al., 2018]). En ellos, los autores aplicaron conceptos de Ingeniería de Software en conjunto con la Interacción Humano-Computador (entre otras disciplinas), para proponer diferentes estrategias, que permitieran llevar adelante actividades con intervención humana mediadas por tecnología. En contextos educativos, para conducir lo que se conoce como Aprendizaje Móvil [Traxler, 2009] y, en lo que se conoce como Ciencia Ciudadana, para ser utilizado como soporte en la recolección y análisis de datos [Bonney et al., 2009], [Strasser et al., 2019].

Lo propuesto en [Lliteras, 2015] y [Lliteras et al., 2018] generó un enfoque para trabajar con actividades que implican intervención humana y, en base al mismo, se propuso una herramienta web de autor [Dal Bianco et al., 2019] que permitió realizar la configuración de una aplicación móvil educativa con el objetivo de, en ese caso en particular, concientizar respecto a los Residuos de Aparatos Eléctricos y Electrónicos (RAEE), como experiencia de uso para un Programa de extensión de la UNLP llamado E-Basura.

En el contexto detallado previamente, se observó la necesidad de proponer una nueva herramienta que profundice y mejore lo propuesto en [Dal Bianco et al., 2019], promoviendo mayor robustez para dar soporte a más elementos del enfoque que la misma adopta como base ([Lliteras, 2015], [Lliteras et al., 2018]). Mediante lo propuesto en esta tesina, se espera favorecer a la configuración de aplicaciones más generales y aptas para llevar adelante otro tipo de actividades que impliquen la intervención de personas para realizarlas, como por ejemplo la recolección y análisis de datos [Steinberg et al., 2019], [Sprinks et al., 2017] ya no sólo en contextos educativos. El diseño e implementación de la herramienta aquí propuesta seguirá los conceptos de reuso esperable para la construcción de este tipo de aplicaciones según lo propuesto en [Lliteras et al., 2017] y en particular, ciertos aspectos serán implementados considerando conceptos de workflows [van Der Aalst et al., 2003], [Kunze et al., 2016].

1.2 OBJETIVOS

Esta tesina propone una arquitectura distribuida basada en microservicios que da soporte a una plataforma web la cual permite a usuarios finales definir actividades que requieran de la intervención humana con el fin de realizar recolección y análisis de datos. Estas actividades se ejecutan desde una aplicación móvil.

¹ Aprobado y subsidiado en la convocatoria de Proyectos de Extensión Universitaria 2018, de la UNLP. El mismo está destinado a promover la educación ambiental sobre los RAEE en escuelas de la región en el período 2018 al 2019.

Se presenta un prototipo de la plataforma web para permitir la creación de actividades y una aplicación móvil prototípica desde la cual se ejecutan las actividades creadas desde la plataforma web. En base a lo anterior se propone un caso de estudio donde es posible diseñar y ejecutar una actividad con el fin de concientización en el reciclado de RAEE (Residuos de Aparatos Eléctricos y Electrónicos). Posteriormente, se define un subconjunto del caso de estudio para realizar pruebas con usuarios, lo que permite brindar un primer análisis acerca de la usabilidad de la plataforma web.

1.3 SOBRE LA ORGANIZACIÓN DE LA TESINA

El resto del informe está estructurado de la siguiente forma:

- En el capítulo 2 se presenta el marco teórico que sustenta la tesina. Se refiere a temas relacionados con el enfoque, con la arquitectura y con el área de aplicación
- En el capítulo 3 se discute el estado del arte en lo que respecta a plataformas de creación de actividades de recolección de datos con intervención humana
- En el capítulo 4 se propone una arquitectura para este tipo de plataformas
- En el capítulo 5 se desarrolla la plataforma DEHIA, la implementación actual de la arquitectura propuesta
- En el capítulo 6 se describe el caso de uso que se eligió para instanciar el prototipo: un formulario para el programa E-basura
- En el capítulo 7 se detallan las conclusiones de la tesina donde además se presentan las contribuciones realizadas a partir de esta tesina y se presentan posibles trabajos futuros
- Al final se encuentran los anexos A-G que contienen material adicional de interés (referenciados a lo largo de la tesina)

CAPÍTULO 2 – MARCO TEÓRICO

DEHIA es una plataforma pensada para definir y ejecutar *actividades* con intervención humana, basadas en *workflows*, que puedan servir para *recolección y análisis de datos*. Su arquitectura es *distribuida* y basada en *microservicios*. Cada uno de los conceptos mencionados (y algunos más) serán desarrollados antes de describir la plataforma, con el fin de fundamentar los capítulos posteriores y facilitar su lectura. Éstos se agrupan de la siguiente forma: conceptos relacionados con el enfoque que se toma como base (1), aquellos relacionados con la arquitectura de la plataforma (2), y aquellos vinculados con el área de aplicación de la solución a presentar (3).

2.1 CONCEPTOS RELACIONADOS CON EL MODELADO DE ACTIVIDADES

Esta tesina se basa en el enfoque presentado en [Lliteras, 2015]. En dicho enfoque, el autor toma elementos de la Teoría de la Actividad, que tiene su origen en la ex Unión Soviética en la escuela de constructivismo socio-cultural fundada por Vygotsky y Leontev en los años 20 y que fue extendida en la actualidad por Engeström. La unidad mínima de análisis de esta teoría es “un sistema de actividad colectivo el cual es mediado por herramientas y está orientado hacia objetos, en el contexto de sus relaciones de red con otros sistemas de actividad” [Engeström, 2001]

En otras palabras, el ser humano lleva cabo actividades para lograr un objetivo, mediadas por el uso de herramientas, de manera social. En esta tesina, en particular, se consideran actividades mediadas por tecnología. Si bien en [Lliteras, 2015] se aborda la temática desde el punto de vista de las actividades educativas (caracterizadas por su intencionalidad de aprendizaje, ya sea de instrucción, de conocimiento o de evaluación [Lliteras, 2015]), el framework propuesto por el autor es lo suficientemente general como para aplicarse a otros contextos como, por ejemplo, recolección y análisis de datos.

Se describen a continuación cada uno de los conceptos relacionados al modelado de actividades, caracterizadas según los antecedentes enunciados.

2.1.1. TAREAS

Una actividad está compuesta por tareas [Lliteras, 2015]. Cada tarea tiene una consigna. El propósito de la tarea depende del objetivo de quien la formula, y puede requerir una respuesta o la realización de una tarea práctica.

En el caso de actividades mediadas por tecnología, estas respuestas pueden implicar la generación de contenido, ya sea texto o multimedia, o la selección de opciones de una lista predefinida. De la misma forma, el enunciado de la tarea puede consistir en un texto, una imagen u otro contenido multimedia.

2.1.2 PLANIFICACIÓN

Una planificación de tareas es la estructura de orden y dependencia entre ellas dentro de una actividad.

En [Lliteras et al., 2012] se presenta una posible estructuración del contenido educativo en el marco de juegos educativos móviles basados en posicionamiento. Dada su generalidad, tal como se describe en [Lliteras, 2015], se puede usar para estructurar

actividades educativas, y, en el caso de esta tesina, también actividades con intervención humana de recolección y análisis de datos.

A continuación, se describen tres de las estructuras de planificación mencionadas en [Lliteras et al., 2012], adaptadas al contexto de actividades y tareas:

- Secuencial Lineal: debe haber una tarea inicial, una final, y cada tarea intermedia debe ser precedida y sucedida por exactamente una tarea. En el contexto de esta tesina, cuando se habla de “planificación secuencial”, se hace referencia a esta estructura.
- Secuencial bifurcada: hay una o más tareas iniciales, una o más tareas finales, y las tareas intermedias pueden ser precedidas o sucedidas por más de una tarea. En el contexto de esta tesina también se conoce esta estructura como “grafo”.
- Conjunto: no hay un orden definido para las tareas, se puede iniciar, continuar y terminar por cualquiera de ellas. No hay una relación de antecesor y sucesor entre las tareas. En el contexto de esta tesina también llama a esta planificación “libre”.

En la Figura 2.1.1 se muestran posibles tipos de planificación, basados en las estructuras propuestas por [Lliteras et al., 2012]

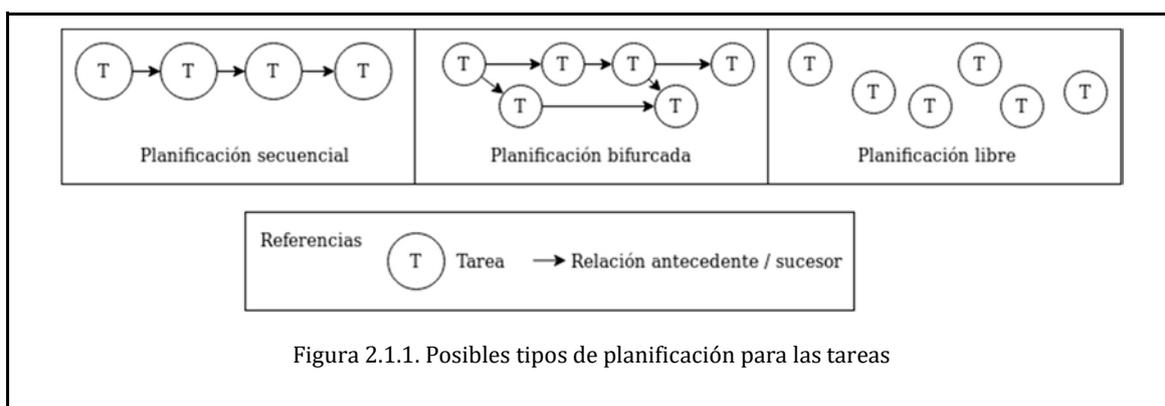


Figura 2.1.1. Posibles tipos de planificación para las tareas

2.1.3. WORKFLOW

La planificación de las tareas puede implementarse desde la perspectiva de lo que se conoce como workflow, y es este el enfoque que se elige en esta tesina. En [Sprinks et al., 2017] se define a un workflow como un conjunto de tareas que deben ser llevadas a cabo para realizar un trabajo, y en donde se plantea que las tareas pueden ser de diferentes tipos e involucrar diferentes formas de interacción por parte de las personas. Un sistema de manejo de workflow permite definir, crear y manejar la ejecución de workflows a través del uso de software.

El uso de workflows para actividades con intervención humana ha sido explorado en diversos dominios, por ejemplo, en análisis de datos [Sprinks et al., 2017] y en recolección de datos por parte de personas y asistidos por tecnología [Steinberg et al., 2019]

Los denominados “patrones de workflow” fueron definidos para representar un conjunto completo de estructuras de control de flujo posibles [van Der Aalst, 2003]. En particular, en [Kunze, et al., 2016] los autores introducen un subconjunto de los patrones

más frecuentemente usados. A su vez, para el desarrollo de esta tesina se eligen algunos de dichos patrones y que se describen en esta sección.

Para desarrollar los ejemplos que ilustran el funcionamiento de sistemas concurrentes, se pueden usar redes de Petri [Kunze, et al., 2016]. Una red de Petri está conformada por un conjunto de *lugares* (representados con círculos), un conjunto de *transiciones* (representadas con rectángulos) y un conjunto de *arcos* que conectan lugares y transiciones, pero nunca dos elementos del mismo tipo. Además, se tienen *fichas* o *tokens* que circulan por la red. Cuando hay tokens en todos los lugares de entrada de una transición, ésta puede dispararse consumiendo los tokens y depositando un token en cada lugar de salida. Pueden extenderse con *condiciones* en los arcos².

2.1.3.1. SECUENCIA

Es la estructura de control más básica. Dos tareas³ A y B tienen que ejecutarse en secuencia si B es causalmente dependiente de A.

En la red de Petri mostrada en la Figura 2.1.2, la transición B no puede ejecutarse hasta que haya un token en el lugar p_1 , es decir, hasta que se haya disparado la transición A.



Figura 2.1.2 Patrón de Secuencia ilustrado con una red de Petri coloreada [Kunze, et al., 2016]

2.1.3.1. SELECCIÓN EXCLUSIVA (*EXCLUSIVE-OR SPLIT*)

Las decisiones en un workflow pueden representarse con el patrón de selección exclusiva (más precisamente, se representa su resultado). La Figura 2.1.3 muestra con una red de Petri coloreada la semántica de ejecución de este patrón. Cuando se dispara la transición A, se evalúa una condición *cond*. Si la condición resulta verdadera, se pone un token en p_1 , y ninguno en p_2 , habilitando así la transición B. Caso contrario, se pone un token en p_2 y ninguno en p_1 , habilitando la transición C.

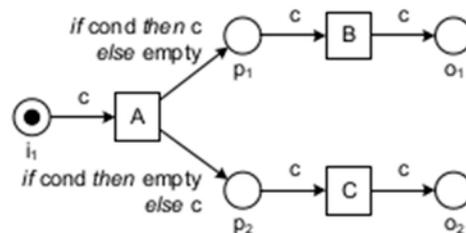


Figura 2.1.3 Patrón de Selección exclusiva ilustrado con una red de Petri coloreada [Kunze, et al., 2016]

2.1.3.1. FUSIÓN SIMPLE (*EXCLUSIVE-OR JOIN*)

² A este tipo de redes se las llama “redes coloreadas”

³ El autor habla de *actividades*, pero en el contexto de workflows son intercambiables. Por lo tanto se usará el término “tarea” para relacionarlo con las tareas que componen una actividad.

El patrón de fusión simple se usa para combinar múltiples flujos alternativos en un workflow. Por lo tanto, se lo puede considerar complementario al patrón de selección exclusiva. La Figura 2.1.4 muestra una red de Petri representando el patrón de fusión simple. Las ramas alternativas se completan con las transiciones A y B. Cuando cualquiera de estas dos transiciones se dispara, se pone un token en p_1 , que realiza la fusión al habilitar la transición C. En el ejemplo sólo A puede dispararse, porque B no está habilitada.

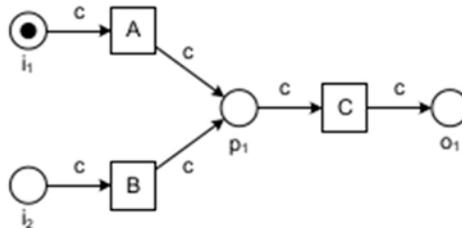


Figura 2.1.4 Patrón de fusión simple ilustrado con una red de Petri coloreada

2.1.3.1. BIFURCACIÓN EN PARALELO (AND SPLIT)

El patrón de bifurcación en paralelo facilita la ejecución concurrente de tareas. La Figura 2.1.5 muestra la semántica de comportamiento de este patrón. Al dispararse la transición A se pone un token en p_1 y un token en p_2 . Como consecuencia, se habilitan las transiciones B y C. En este estado, B y C pueden dispararse independientemente, resultando en su ejecución concurrente. El nombre del patrón puede ser engañoso, porque no obliga a la ejecución concurrente en el sentido de que las tareas sean ejecutadas a la vez en el tiempo. En lugar de eso la palabra “paralelo” representa la ejecución concurrente de tareas.

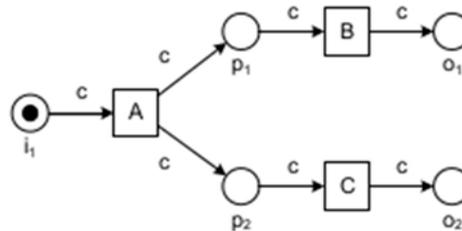


Figura 2.1.5 Patrón de bifurcación en paralelo ilustrado con una red de Petri [Kunze, et al., 2016]

2.1.3.1. FUSIÓN DE SINCRONISMO (AND JOIN)

El patrón de fusión de sincronismo es usado para esperar que se complete la ejecución de ramas concurrentes de ejecución. Este patrón complementa al de bifurcación en paralelo, descrito anteriormente. La red de Petri de la Figura 2.1.6 representa la semántica de ejecución de este patrón. La tarea de sincronización C sólo puede ejecutarse si hay un token en p_1 y un token en p_2 , de forma tal que sincroniza las ramas de ejecución concurrente.

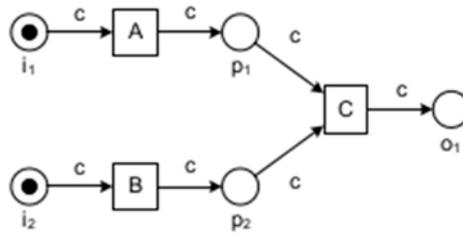


Figura 2.1.6 Patrón de fusión de sincronismo ilustrado por una red de Petri coloreada [Kunze, et al., 2016]

2.2 CONCEPTOS RELACIONADOS CON LA ARQUITECTURA

La arquitectura de un sistema puede definirse como “*El conjunto de estructuras que se necesitan para razonar acerca del sistema, el cual comprende elementos de software, relaciones entre ellos, y propiedades de ambos*” [Bass, et al., 2003].

El estilo de arquitectura elegido para la plataforma es el de *Microservicios*. Es un tipo de arquitectura distribuida que tiene una serie de características interesantes y será explicado en las próximas secciones. Sin embargo, antes de llegar a una definición, es necesario hablar sobre los sistemas distribuidos, el protocolo HTTP y el conjunto de restricciones REST.

2.2.1. SISTEMAS DISTRIBUIDOS

Los sistemas distribuidos pueden concebirse como aquellos cuya funcionalidad se encuentra fraccionada en componentes que al trabajar sincronizada y coordinadamente otorgan la visión de un sistema único, siendo la distribución transparente para quien hace uso del sistema. [Bazán, et al., 2017].

También se puede definir un sistema distribuido como aquel en el cual componentes de software o hardware ubicados en computadoras en red se comunican y coordinan sus acciones sólo por pasaje de mensajes [Colouris, et al., 2000]

Para poder presentar una visión de un sistema único dando soporte a redes y computadoras heterogéneas, los sistemas distribuidos muchas veces se organizan como una capa de software que está ubicada lógicamente entre una capa superior de usuarios y aplicaciones, y una capa inferior que incluye al sistema operativo y mecanismos básicos de comunicación, como se muestra en la Figura 2.2.1. Este tipo de sistema distribuido se denomina *middleware* [Tanenbaum, et al., 2007]

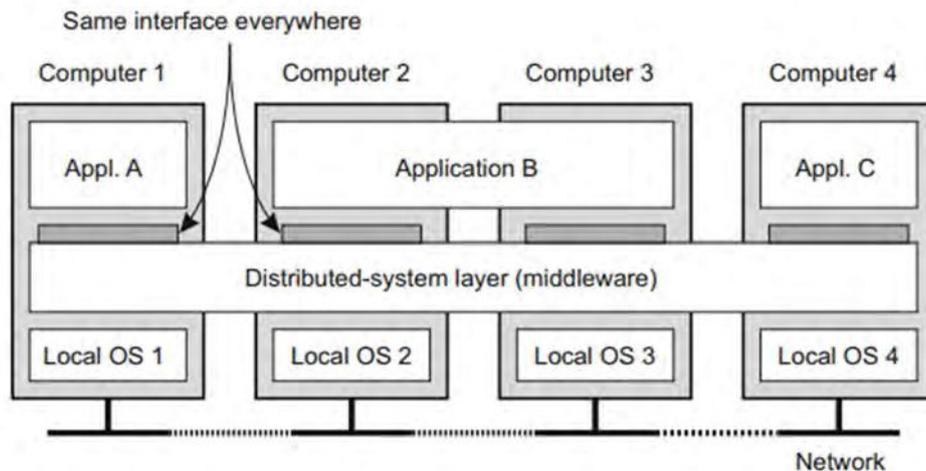


Figura 2.2.1 Un sistema distribuido organizado como *middleware* [van Steen, et al., 2016]

Algunos de los motivos para la distribución incluyen la distribución funcional, donde computadoras con diferentes capacidades cooperan para un fin común; la distribución inherente al dominio de aplicación, donde los componentes se modelan en base a un dominio que ya es distribuido; y el balanceo de carga, que busca mejorar el rendimiento general asignando tareas a distintos procesadores (de hecho, esta alternativa resulta más económica que un solo *mainframe*).

La distribución de un sistema implica que cada componente puede ejecutar tareas de manera concurrente, mejorando el rendimiento, pero a la vez exponiéndolo a la posibilidad de *deadlocks*. La sincronización debe llevarse a cabo a través de pasaje de mensajes, dado que los componentes no pueden sincronizar relojes (añadido el retardo de la red y las llamadas remotas, se tiene retraso o latencia en la comunicación). También se introducen nuevos tipos de falla, como fallas en la red (aunque todos los componentes sigan funcionando no pueden comunicarse) y fallas independientes de componentes (que deben ser detectadas por los demás para actuar en consecuencia).

2.2.2. HTTP

Uno de los mecanismos posibles de pasaje de mensajes (sincrónico) entre componentes de un sistema distribuido es HTTP.

HTTP (*HyperText Transfer Protocol*, protocolo de transferencia de hipertexto) fue presentado como “un protocolo de red de nivel de aplicación con la liviandad y velocidad necesaria para sistemas de información hipermedia distribuidos; un protocolo genérico y sin estado que puede ser usado para diversas tareas, desde servidores de nombres hasta sistemas de objetos distribuidos” [Berners-Lee, et al., 1996].

El protocolo HTTP está basado en un paradigma de petición/respuesta. Un cliente establece una conexión con un servidor y envía una petición comenzando con una línea inicial que contiene *método de petición*, una *URI* y versión del protocolo, seguida por un conjunto de encabezados (*headers*) del cliente y posiblemente un cuerpo con contenido. El servidor responde con una *línea de estado*, incluyendo la versión de protocolo de mensaje y un *código de éxito o error*, seguida por encabezados del servidor y posiblemente un cuerpo con contenido.

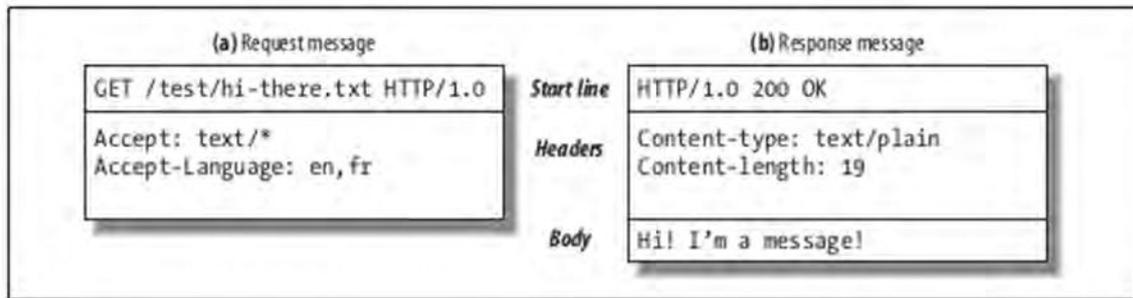


Figura 2.2.2 Ejemplo de petición/respuesta HTTP [Gourley, et al., 2002]

A continuación, se detallan algunos de los componentes de las peticiones y respuestas HTTP.

2.2.2.1. MÉTODOS

HTTP permite diferentes tipos de órdenes en las peticiones, llamados *métodos*. Según el método utilizado, el servidor realizará diferentes acciones sobre el recurso indicado por la URI.

A continuación, se listan los métodos más comunes según [Gourley, et al., 2002]:

- GET: Enviar el recurso indicado del servidor al cliente
- PUT: Guardar datos desde el cliente a un recurso nombrado del servidor
- DELETE: Borrar el recurso nombrado del servidor
- POST: Enviar datos de del cliente al servidor
- HEAD: Enviar únicamente los encabezados (*headers*) de la respuesta para el recurso nombrado

2.2.2.2. URI

En la primera línea de una petición HTTP, después del método, se indica el nombre del recurso sobre el que se quiere operar, por medio de una *URI (uniform resource identifier)*. Las URIs pueden tener varias formas, pero la más utilizada es la de *URL*.

Una URL (*uniform resource locator*) describe la ubicación específica de un recurso en un servidor particular. Está compuesta por un protocolo (1), un servidor (2), y un nombre de recurso local (3), como se muestra en la Figura 2.2.3:

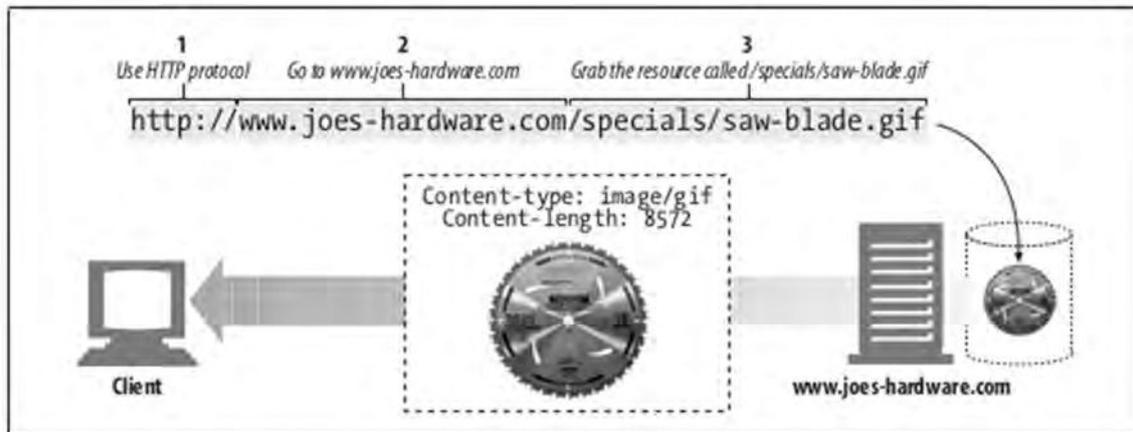


Figura 2.2.3 Partes de una URL [Gourley, et al., 2002]

2.2.2.3. CÓDIGOS DE ESTADO

Cuando el servidor envía una respuesta al cliente, en la primera línea del mensaje HTTP se indica mediante un *código de estado* el resultado de la operación. Los códigos de estado tienen tres dígitos, y se agrupan de acuerdo a su primer dígito en categorías:

- 1xx: Informativo - la petición se recibió, continúa el procesamiento
- 2xx: Éxito - La acción se recibió correctamente, se comprendió y se aceptó
- 3xx: Redirección - Son necesarias acciones adicionales para completar la petición
- 4xx: Error de cliente - La petición contiene sintaxis incorrecta o no puede llevarse a cabo
- 5xx: Error de servidor - El servidor no pudo llevar a cabo una petición aparentemente válida

En la RFC 1945⁴ (HTTP 1.0) y la RFC 2616⁵ (HTTP 1.1) se proponen algunos de estos códigos y el IANA mantiene una lista⁶ de códigos que aparecen en otros documentos⁷. A continuación, se mencionan los que se usan en la plataforma:

- 200 - OK: la petición se recibió satisfactoriamente
- 201 - Creado: se creó un recurso como consecuencia de la petición
- 204 - Sin contenido: No hay cuerpo en la respuesta. Se suele usar en la respuesta a peticiones con el método DELETE
- 400 - Petición mala: Hay un problema de sintaxis u otro problema causado por el cliente en la petición
- 401 - No autorizado: No se recibieron credenciales al acceder a un recurso protegido

⁴ <https://tools.ietf.org/html/rfc1945>

⁵ <https://tools.ietf.org/html/rfc2616>

⁶ <https://www.iana.org/assignments/http-status-codes>

⁷ En la [RFC 2324](#) se agrega en broma el código de error 418 - *Soy una tetera* para indicar que se intentó servir café con una tetera, en un protocolo ficticio para manejar cafeteras basado en HTTP.

- 403 - No permitido: Las credenciales enviadas no son suficientes para efectuar la operación
- 404 - No encontrado: El recurso solicitado no existe
- 500 - Error interno del servidor: La petición es válida, pero ocurrió un error inesperado

2.2.3. REST

En cuanto a la interfaz que exponen los componentes del sistema, una de las posibilidades es usar REST (*REpresentational State Transfer*). Es un conjunto de restricciones que se usó inicialmente para describir el funcionamiento de la Web como una aplicación distribuida en la que *recursos* enlazados se comunican mediante representaciones de su estado [Webber, et al., 2010].

Estas restricciones se mencionan a continuación [Richardson, et al., 2013]:

- Cliente - Servidor: La lógica de la solución está separada en lógica de cliente (o consumidor) y de servidor.
- Sin estado: el servidor no mantiene un estado entre peticiones, esto es responsabilidad del cliente
- Cache: un cliente puede reusar respuestas marcadas como cacheables
- Sistema por capas: se pueden insertar intermediarios de manera transparente entre cliente y servidor
- Interfaz uniforme: cliente y servidor comparten un contrato técnico genérico y limitado. Esto requiere otras restricciones:
 - Identificación de recursos: cada recurso es accesible por una URI estable
 - Manipulación de recursos mediante representaciones: el servidor describe el estado de un recurso enviando representaciones al cliente. El cliente manipula este estado enviando representaciones al servidor
 - Mensajes autodescriptivos: toda la información necesaria para entender una petición o una respuesta está contenida en el mismo mensaje
 - Hipermedia: el servidor manipula el estado del cliente enviando un “menú” de enlaces para que éste elija libremente

2.2.3.1. RECURSOS

Un *recurso* es cualquier objeto, informático o físico, que es de interés y puede ser localizable dentro del sistema. Una URI sirve tanto para localizar un elemento como para identificarlo. La *representación* de un recurso es una vista de cómo es el estado actual del recurso. Una misma URI puede apuntar a distintos *formatos* (JSON, XML, etc) de una representación. Los componentes del sistema se comunican intercambiando estas representaciones. [Webber, et al., 2010]

Para que dos componentes puedan comunicarse son necesarios identificadores (URIs), un tipo de medio (formato) y un conjunto de acciones estándar (métodos HTTP) que describen la acción a realizar.

2.2.3.2. MODELO DE MADUREZ DE RICHARDSON

Es un modelo que clasifica los servicios en la Web en tres niveles, de acuerdo a si soportan recursos, verbos (métodos) HTTP e hipermedia, respectivamente. En [Webber, et al., 2010], se los define como se muestra en la Figura 2.2.4:

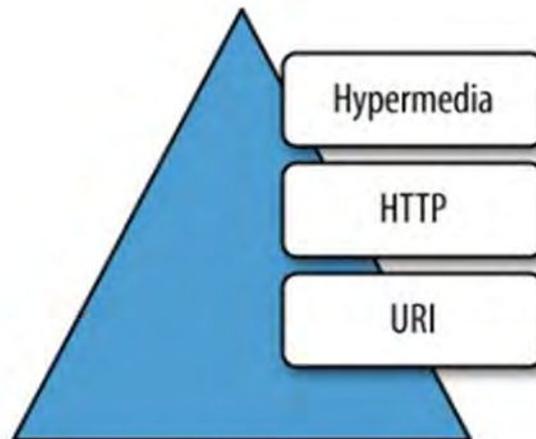


Figura 2.2.4 Niveles de madurez de acuerdo al modelo de Richardson [Webber, et al., 2010]

A continuación, se describe brevemente cada nivel mostrado en la Figura 2.2.4:

Nivel 0: Se tiene una sola URI y un sólo verbo (usualmente POST). Básicamente se usa HTTP a modo de túnel para hacer una llamada de procedimiento remoto.

Nivel 1 - Recursos: Se tiene múltiples URI pero un único verbo, en este caso más usualmente GET. Las operaciones se indican con parámetros en la URI.

Nivel 2 - Verbos HTTP: Se tienen numerosos recursos localizables por URIs. Se soportan varios métodos HTTP para el mismo recurso. Se incluyen en este grupo los servicios de tipo CRUD (alta, lectura, baja y modificación), en los cuales el estado de los recursos se puede manipular a través de la red. La plataforma pretende ubicarse en este nivel.

Nivel 3 - Hipermedia: Se usa hipermedia (links) como el motor para el estado de la aplicación. Esto quiere decir que una representación contiene URIs que apuntan a otros recursos de forma que la aplicación cambia de estado cuando el cliente navega por ellos.

Después de describir sistemas distribuidos, HTTP y REST, se puede pasar a una caracterización de los microservicios.

2.2.4 MICROSERVICIOS

Martin Fowler y James Lewis [Fowler, et al., 2014] describen a los microservicios de la siguiente manera: *“Los microservicios son un patrón o estilo de arquitectura distribuida en el cual se desarrolla una única aplicación como un conjunto de pequeños servicios, cada cual ejecutándose en su propio proceso y comunicándose con mecanismos ligeros, a menudo una API HTTP. Estos servicios se construyen alrededor de capacidades de negocio y se pueden desplegar de manera independiente por mecanismos completamente automatizados. Hay un mínimo de administración centralizada de estos servicios, que*

pueden estar escritos en diferentes lenguajes de programación y usar diferentes tecnologías de almacenamiento”.

Los componentes de este estilo de arquitectura distribuida son los microservicios, componentes autónomos que tienen alta cohesión y bajo acoplamiento. Esto quiere decir que la funcionalidad relacionada está en el mismo servicio, y un cambio en esa funcionalidad no afecta a los demás servicios [Newman, 2015].

En lugar de usar sistemas monolíticos, es decir, conformado por una única pieza de código que resuelve la interfaz, la lógica de negocio y el acceso a los datos, una alternativa es descomponer el sistema en *microservicios*, componentes de software débilmente acoplados y comunicados por pasaje de mensajes. Se vuelve más fácil desarrollar una funcionalidad dentro de un pequeño servicio con un contexto limitado, y no se requiere conocimiento detallado del resto de la aplicación. Cada servicio se puede testear y desplegar de forma independiente, y escalar cierta funcionalidad implica replicar sólo el microservicio que la implementa. Una ventaja interesante de los microservicios (y de los sistemas distribuidos en general) es su heterogeneidad: cada componente puede estar implementado en una tecnología diferente, y, dado su tamaño, reemplazar un microservicio tiene bajo costo. Sin embargo, se agrega la complejidad propia de un sistema distribuido: hay problemas internos de red que antes no existían, se debe coordinar los servicios, y testearlos en su conjunto no es tan fácil.

Como cualquier arquitectura distribuida, los microservicios requieren definir dos aspectos necesarios para su despliegue y funcionamiento: 1 - Cómo dialogan entre sí (comunicación) y 2 - Cómo operan para componer funcionalidad (coordinación).

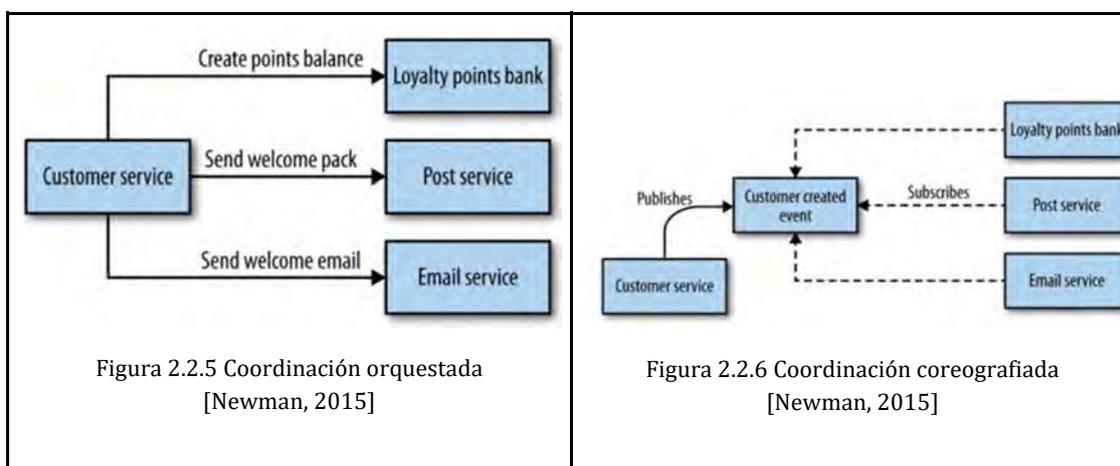
Los servicios deben comunicarse entre ellos mediante pasaje de mensajes. Existen dos formas de hacerlo: sincrónica y asíncrona. En comunicación sincrónica, el servicio que hace la llamada a queda bloqueado hasta que el otro servicio responda. En comunicación asíncrona, el servicio hace la llamada y continúa su ejecución, sin saber el resultado de la operación.

La comunicación sincrónica es más fácil de seguir, y se tiene la confirmación de si fue exitosa o no la operación. Sin embargo, si se tiene en cuenta la latencia, puede llegar a demorar mucho tiempo, dejando la interfaz de usuario en espera. La comunicación asíncrona, por su parte, es más complicada, pero permite la ejecución de operaciones de larga duración, en las cuales no es práctico que el llamante se quede esperando [Newman, 2015].

Esto lleva a dos *modos de colaboración* entre servicios: petición/respuesta o basada en eventos. En el caso de petición/respuesta, un cliente inicia una petición y se queda esperando una respuesta. En el otro caso, un servicio emite un *evento*, que es recibido por las partes interesadas, que actúan en consecuencia (el primer servicio no sabe quién recibirá el evento y no tiene que saber cómo comunicarse con él).

Existen dos formas de coordinar servicios: orquestada o coreografiada. Con orquestación, hay un encargado central de dirigir las acciones de los demás servicios. Cuando se tiene comunicación sincrónica, el servicio que hace de cliente funciona como director, y va llamando a los demás servicios a medida que los necesita. En una coordinación coreografiada, no hay un director central, sino que cada servicio sabe lo que tiene que hacer en base a los eventos que se generan en la red. Esto se alinea más con una comunicación asíncrona.

En las Figuras 2.2.5 y 2.2.6 se observa la diferencia entre estas dos formas de coordinación con un ejemplo:



2.3 CONCEPTOS RELACIONADOS CON EL ÁMBITO DE APLICACIÓN DE LA PLATAFORMA PROPUESTA

La plataforma DEHIA se basa en un framework de modelado de actividades que fue originalmente pensado para actividades educativas. Como se explicó en la sección (1), el framework es lo suficientemente general como para abarcar varios tipos de actividades que impliquen intervención humana, entre ellas aplicadas al Aprendizaje Móvil, y a la recolección y análisis de datos para Ciencia Ciudadana (de acuerdo a lo mencionado en [Lliteras et al., 2019]). En esta sección se describirán muy brevemente estas áreas.

2.3.1 APRENDIZAJE MÓVIL

El Aprendizaje Móvil (*m-learning*) es un caso particular del aprendizaje electrónico (*e-learning*) en el cual el artefacto mediador son los dispositivos móviles [Lliteras, 2015]. Una de las definiciones posibles para este tipo de aprendizaje se da en [O'Malley et al., 2003]:

“Cualquier tipo de aprendizaje que ocurre cuando el alumno no está en una posición fija y predeterminada, o el aprendizaje que ocurre cuando el alumno aprovecha oportunidades de aprendizaje provistas por las tecnologías móviles”

El factor clave que motiva la adopción del Aprendizaje Móvil, según [Cochrane and Bateman, 2010], es el enriquecimiento de la enseñanza y el aprendizaje, que facilita una pedagogía constructivista social centrada en el alumno. De acuerdo a los mismos autores, algunos de los principales beneficios del Aprendizaje Móvil (en nivel secundario) son:

- Explorar prácticas innovadoras de enseñanza y aprendizaje.
- Permitir la realización de “aprendizaje auténtico”, por ejemplo, permitiendo aprendizaje centrado en el alumno en cualquier momento y lugar.
- Involucrar a los estudiantes con los potenciales de las tecnologías móviles 2.0: conectividad, movilidad, geolocalización, redes sociales, etc.

- Acortar la brecha digital proveyendo acceso a contextos de aprendizaje y herramientas de creación de contenido que son accesibles y los alumnos poseen cada vez más.
- Pasar de un modelo de computación fija a un paradigma de computación móvil, inalámbrica, que convierte cualquier espacio en un potencial espacio de aprendizaje.

2.3.2. RECOLECCIÓN DE DATOS

La recolección de datos por parte de personas es una actividad que se usa en diversas disciplinas y que se realiza desde hace ya largo tiempo. En el área de las ciencias sociales, por ejemplo, se suele pedir información a la gente mediante encuestas, experimentos y estudios de campo [Babbie, 2000]. Inicialmente, la recolección de los datos se realizaba empleando medios físicos como papel y lápiz y luego, con el avance de la tecnología, este medio fue reemplazado por otros, como por ejemplo las planillas de cálculo [Morris et al., 2018].

En la actualidad, quienes recolectan datos pueden valerse de software preexistente: plataformas de encuestas como SurveyMonkey⁸, LimeSurvey⁹ o Google Forms¹⁰; blogs; o redes sociales como Twitter [Kim, et al., 2013]. Una alternativa más específica, pero que requiere de conocimientos de programación, es la de desarrollar un sitio o una aplicación móvil para alguna necesidad puntual del proyecto, pero que no pueden ser reusadas en otros dominios. Ejemplos de esto son AppEAR [Cochero, 2018], una aplicación para recolectar datos relacionados con ambientes acuáticos y GeoVIN [Balsalobre, 2018], una aplicación para recolectar datos ayudando a la sociedad a reconocer vinchucas. Una tercera alternativa, más flexible, es la de usar plataformas de generación de proyectos de recolección de datos como Open Data Kit [Hartung et al., 2010]. Estas soluciones generalmente incluyen una aplicación móvil que puede ser reusada con distintas configuraciones de acuerdo a las necesidades de cada proyecto. Herramientas de este tipo serán analizadas en el Capítulo 3. En la Figura 2.3.1, 2.3.2 y 2.3.3 se pueden ver ejemplos de cada alternativa.

Dentro los proyectos que utilizan recolección de datos están aquellos llevados a cabo por un grupo de investigadores que se encarga de la recolección y el análisis. También existen otros tipos de proyectos donde la recolección de datos, su análisis o ambos, es realizado por una comunidad de voluntarios. Dos actividades que usan esta modalidad son el Sensado Participativo (*Participatory Sensing*) y la Ciencia Ciudadana. En el primer caso, presentado en [Burke, et al., 2006], los voluntarios utilizan dispositivos móviles para aportar datos a una investigación llevada a cabo por una entidad central, ya sea un grupo de científicos, un organismo gubernamental, etc. que podría servir según los autores por ejemplo para monitoreo de la salud pública, o para ayudar en la planificación urbana. En el segundo caso, los voluntarios pueden participar tanto de la recolección como del análisis de los datos, como se verá en la próxima sección.

⁸ <https://surveymonkey.com>

⁹ <https://www.limesurvey.org>

¹⁰ <https://www.google.com/forms/about/>



Figura 2.3.1 Google Forms



Figura 2.3.2 GeoVIN

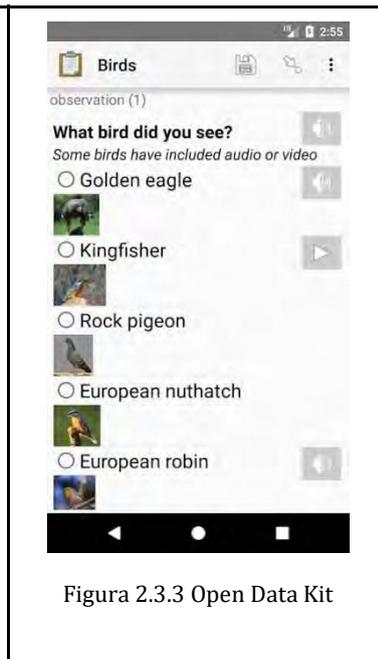


Figura 2.3.3 Open Data Kit

2.3.3. CIENCIA CIUDADANA

Ciencia Ciudadana se refiere a un tipo de investigación en la cual se llama al público general a participar de la recolección de información científica [Bonney et al., 2009]. También se la puede definir como una investigación llevada a cabo, totalmente o en parte, por participantes no profesionales, a menudo usando técnicas de crowdsourcing, según [Sprinks et al., 2017].

En sus inicios, los proyectos de Ciencia Ciudadana contaban con voluntarios tomando notas sobre observaciones de campo, como por ejemplo el conteo de aves de las vacaciones de invierno de Sociedad de Ornitología Audubon¹¹ o los proyectos del Laboratorio Cornell de Ornitología [Bhattacharjee, 2005]. Actualmente los proyectos de Ciencia Ciudadana incluyen tanto a participantes que toman muestras “en la naturaleza” por medio de dispositivos móviles como a aquellos que analizan, por medio de plataformas basadas en Internet, datos recolectados previamente.

Una plataforma (que será analizada en el próximo capítulo) para este tipo de proyectos es Zooniverse [Simpson et al., 2013]. Los investigadores pueden crear su proyecto en el sitio web y suben contenido multimedia para ser analizado por usuarios sin formación científica. Este análisis puede ser marcar estrellas en una foto del espacio, identificar animales en una foto de una reserva usando una guía de campo (Figura 2.3.4), o transcribir cartas manuscritas de un artista. Este tipo de tareas se lleva a cabo a una escala que un grupo de investigadores no podría alcanzar por su cuenta.

¹¹ <https://www.audubon.org/conservation/history-christmas-bird-count>

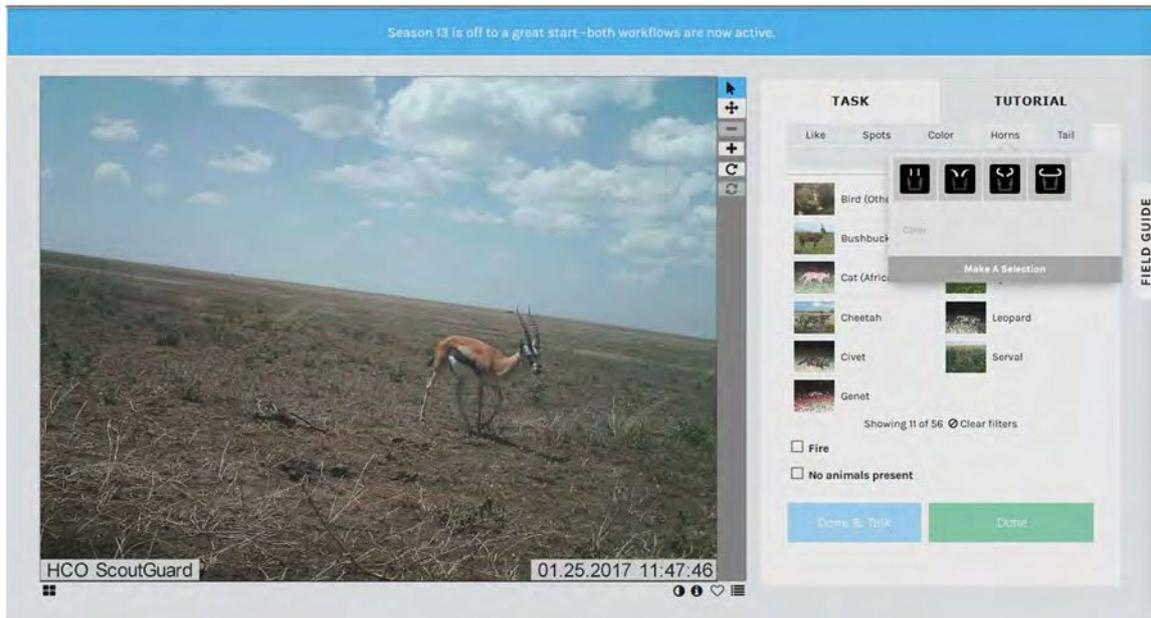


Figura 2.3.4 Snapshot Serengeti, uno de los proyectos de Zooniverse

2.4 RECAPITULACIÓN

En la sección 2.1 se presentaron las actividades. Una actividad tiene un conjunto de *tareas* que se *planifican* usando *workflows*. En la sección 2.2 se explicaron conceptos relacionados con la arquitectura de microservicios: los sistemas distribuidos, el protocolo HTTP y las interfaces REST. Por último, en la sección 2.3 se habló de alguna de las áreas en que es posible la aplicación de la plataforma: Aprendizaje Móvil, recolección de datos, y como caso particular de este último, la Ciencia Ciudadana. En el siguiente capítulo se discutirá el estado del arte en cuanto a la creación de actividades con intervención humana por parte de personas sin conocimientos de programación. Se analizarán algunas de las plataformas existentes y luego se realizará una comparación de sus funcionalidades y arquitecturas.

CAPÍTULO 3 - ESTADO DEL ARTE

Entre las alternativas de soluciones de recolección de datos por parte de personas mencionadas en el Capítulo 2, se destacan por su flexibilidad y potencial de reutilización aquellas que permiten a usuarios finales, sin necesidad de que posean conocimientos de programación, la creación de actividades para múltiples dominios. Algunas de ellas se concentran en un rango de dominios específicos, como la ciencia o la educación, pero permitiendo que diferentes proyectos creen actividades en base a sus necesidades. En este capítulo se analizarán cuatro ejemplos de este tipo de plataformas: EpiCollect5, Zooniverse, Open Data Kit y MoLE. A continuación, se describen las cuatro plataformas mencionadas, para luego abordar la comparación entre ellas.

3.1 EPICOLLECT5

EpiCollect5¹² es una plataforma de creación de formularios para recolección de datos usando smartphones. Fue desarrollada en el Imperial College de Londres, inicialmente para el estudio de epidemiología (de ahí el nombre *EpiCollect*). Tuvo tres versiones publicadas: EpiCollect, presentada en 2009; EpiCollect+, presentada en 2014; y la versión actual, EpiCollect5, que está disponible desde 2017.

En su primera versión, llamada EpiCollect [Aanensen, et al., 2009], la plataforma estaba formada por una aplicación móvil para ejecutar las actividades (formularios) y una aplicación web para diseñarlas y ver los resultados. La comunicación entre la aplicación móvil y el servidor era bidireccional: la aplicación podía descargar formularios, subir los datos recolectados y también descargar datos recolectados por otros en forma de mapas. Al ejecutar una actividad se tomaban las coordenadas del teléfono, para después sacar opcionalmente una foto y/o llenar campos de tipo texto, selección u opción múltiple según se hubiera definido anteriormente. Una actividad ejecutada por EpiCollect puede verse en la Figura 3.1.1.



Figura 3.1.1 Actividad ejecutándose en EpiCollect [Aanensen, et al., 2009]

Se requería conectividad para descargar los formularios y enviar los resultados al servidor, pero no para recolectar los datos (el sensor de GPS tampoco necesita conexión a

¹² <https://five.epicollect.net/>

Internet). Se podía acumular múltiples respuestas para ser enviadas posteriormente al recuperar la conexión. Esto permitía tomar muestras en una zona con poca o nula conectividad.

La visualización de resultados consistía en gráficos y mapas marcando la posición de los teléfonos en el momento de la recolección, pudiéndose filtrar por un rango de fechas.

En cuanto a su arquitectura, como se muestra en la Figura 3.1.2, EpiCollect contaba con un servidor web, *spatialepidemiology.net*, su respectivo sitio web y una aplicación móvil. El servidor web estaba alojado en el servicio Google App Engine, y por lo tanto hacía uso de una base de datos NoSQL de Google (Google Datastore).

El proyecto era de código abierto, aunque actualmente ya no recibe mantenimiento.

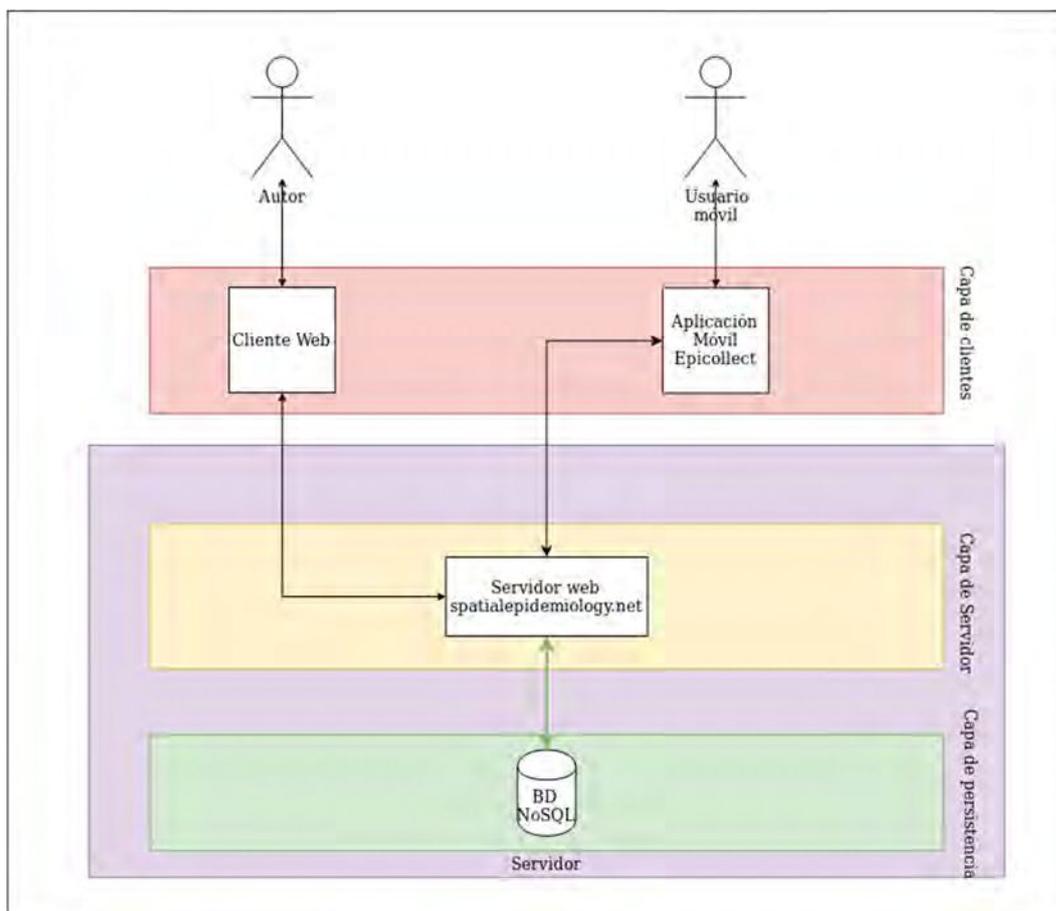


Figura 3.1.2 Arquitectura de EpiCollect

En 2014 los autores de EpiCollect presentan EpiCollect+ [Aanensen, et al., 2014] como una herramienta avanzada para la definición de formularios más complejos que los que eran posibles con la primera versión. Se añadieron nuevos tipos de tarea multimedia, como la toma de videos y la lectura de códigos de barras. En cuanto a las actividades, la principal novedad de esta herramienta fue la capacidad para definir formularios jerárquicos y ramificación (loops). En el primer caso, se podía definir un formulario que dependía de otro: por ejemplo, información sobre una escuela que además contiene información sobre cada clase en formularios separados. En el caso de la ramificación, una misma tarea podía ser ejecutada varias veces con diferentes datos para representar, por ejemplo, diferentes personas de una familia en un relevamiento sobre hogares. También se

agregó la posibilidad de “saltar” de una tarea a otra (planificación), algunas veces de manera condicional, y de validar los campos.

En la Figura 3.1.3 se muestra un formulario jerárquico en Epicollect+.

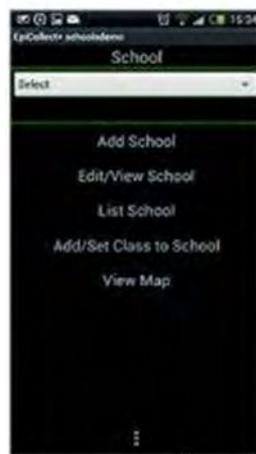


Figura 3.1.3 Formulario jerárquico en Epicollect+ [Aanensen, et al., 2014]

La arquitectura de EpiCollect+ puede verse en la Figura 3.1.4. Consistía en una nueva aplicación móvil, Epicollect+ y un servidor web con su respectivo cliente, además de una API REST para consultar los resultados. La principal diferencia con EpiCollect es que en este caso el servidor puede ser propio. Consistía en una aplicación PHP corriendo sobre Apache con una base de datos MySQL. El proyecto era de código abierto y dejó de mantenerse en 2017 con la aparición de su iteración más nueva: EpiCollect5.

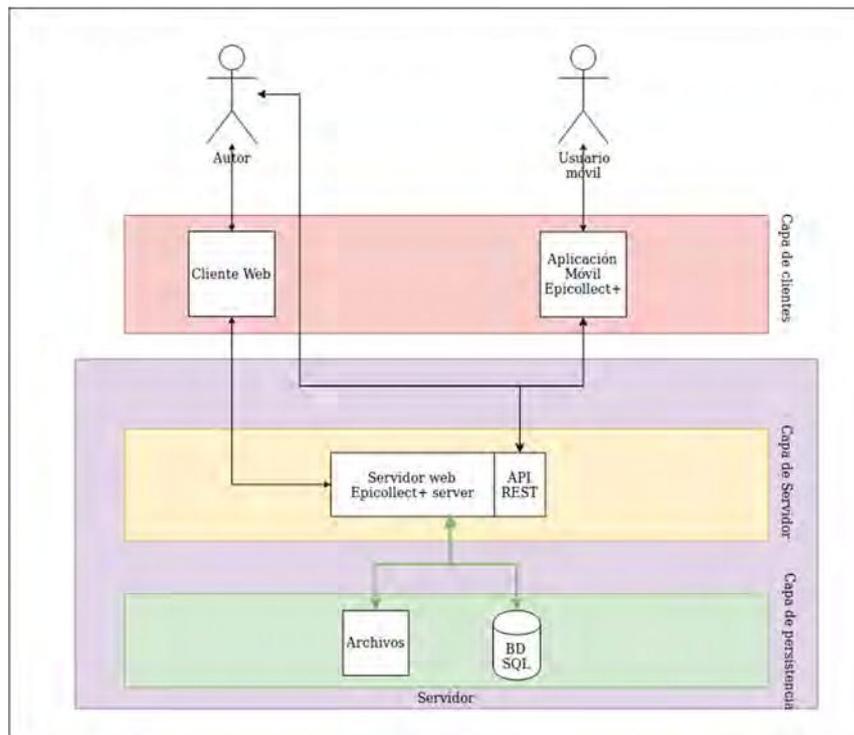


Figura 3.1.4 Arquitectura de Epicollect+

EpiCollect5 cuenta con un sitio web nuevo y nuevos tipos de tareas, como fechas, búsqueda, cajas de texto, etc. Añadió la posibilidad de importar proyectos completos o formularios (actividades completas) a partir de archivos JSON exportados por la misma plataforma. Esto marca una diferencia con la versión anterior, que usaba XML. Además, esta nueva versión está también disponible para iPhone. La versión actual de la herramienta de diseño de formularios aparece en la Figura 3.1.5. El proyecto actualmente es de código cerrado.

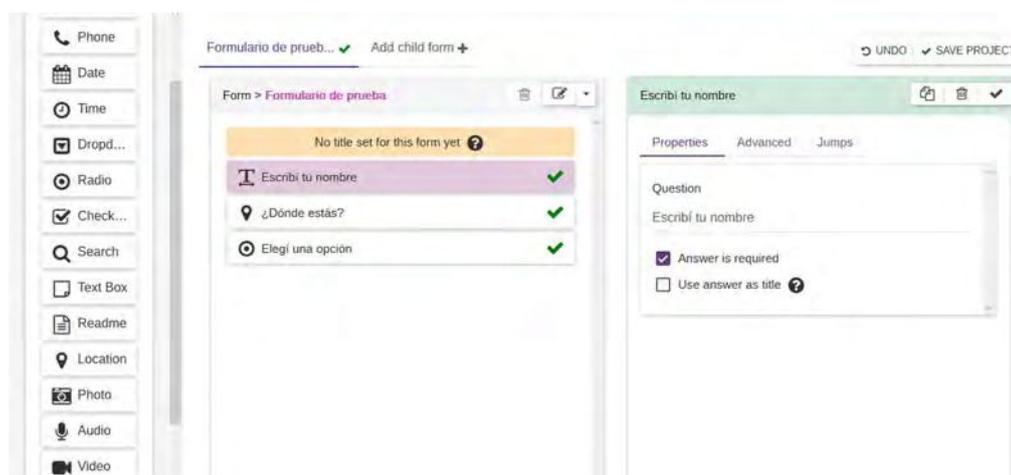


Figura 3.1.5 Herramienta de definición de formularios de EpiCollect5

El inicio de sesión es exclusivamente mediante una cuenta de Google. Desde el sitio web se puede crear una “aplicación” OAuth2.0 para identificarse frente a la API, que sólo permite descargar los formularios y las respuestas (es de sólo lectura).

EpiCollect5 comparte su arquitectura con Epicollect+, con la salvedad de que el servidor es el provisto por *five.epicollect.net*. La Figura 3.1.6 ilustra la arquitectura de EpiCollect 5.

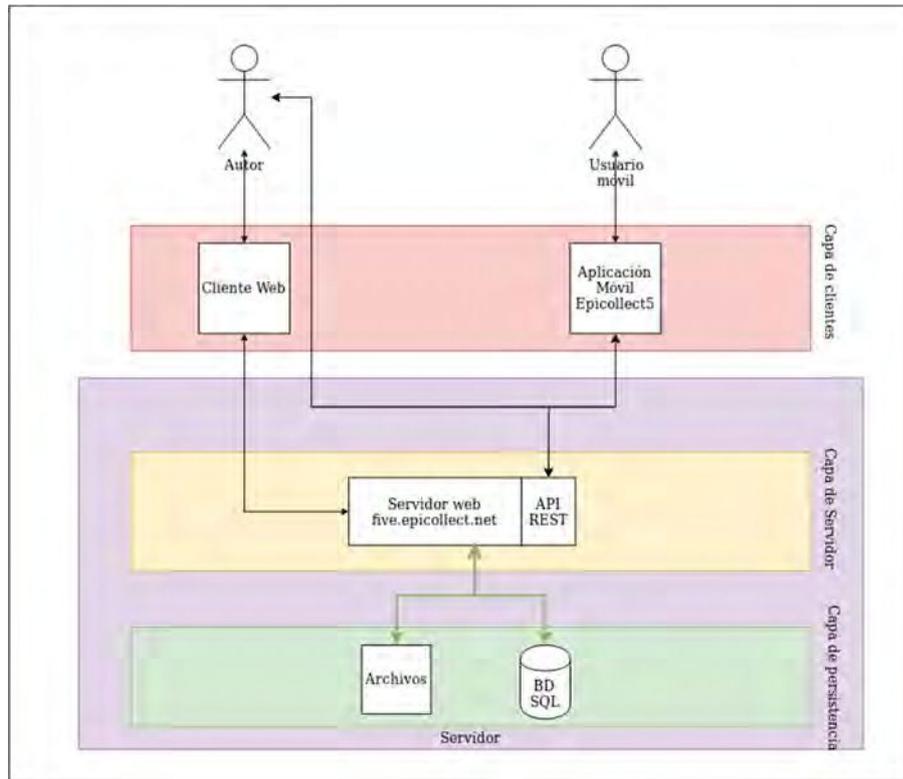


Figura 3.1.6 Arquitectura de EpiCollect5

3.2 ZOONIVERSE

Zooniverse¹³ es una plataforma de código abierto para proyectos de Ciencia Ciudadana, en la cual investigadores recurren a personas comunes para que los ayuden en el análisis de datos a una escala que no podrían por su cuenta [Simpson, et al., 2013], especialmente en tareas resueltas con mayor facilidad por personas que por máquinas, como el reconocimiento de imágenes [Trouille, et al., 2019].

Algunos de los proyectos encontrados en Zooniverse son GalaxyZoo¹⁴ (en la Figura 3.2.1), que sirve para clasificar galaxias de acuerdo a su forma, Snapshot Serengeti¹⁵, que pide clasificar animales de acuerdo a sus características, y Star Notes¹⁶, que busca transcribir cuadernos de notas de ciertas astrónomas. Cada uno de estos proyectos aprovecha un conjunto distinto de herramientas provistas por Zooniverse para la resolución de tareas, que serán detalladas más adelante.

¹³ <https://www.zooniverse.org/>

¹⁴ <https://www.zooniverse.org/projects/zookeeper/galaxy-zoo/>

¹⁵ <https://www.zooniverse.org/projects/zooniverse/snapshot-serengeti>

¹⁶ <https://www.zooniverse.org/projects/projectphaedra/star-notes/>

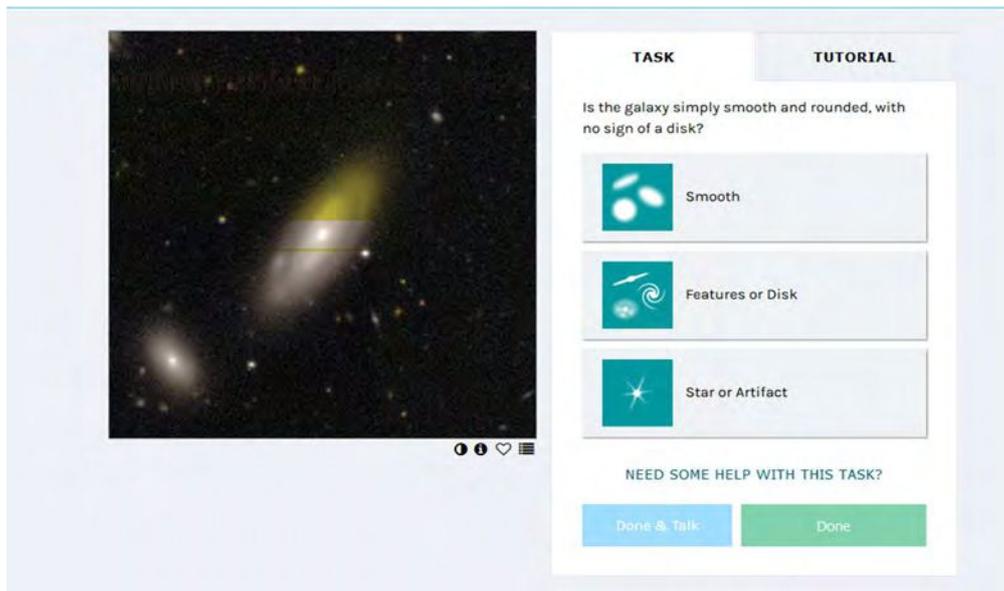


Figura 3.2.1 Tarea dentro del proyecto Galaxy Zoo

En Zooniverse, un investigador sube un conjunto de muestras multimedia (imágenes, audio, video) para analizar, denominado *colección de sujetos*. Además, genera uno o varios *workflows* de tareas, donde indica qué se necesita saber sobre cada sujeto. Cuando un participante interactúa con el sujeto a partir de las tareas que le fueron asignadas, lo que se genera es una *clasificación*, que es la mínima unidad de trabajo humano en la plataforma [Simpson, et al., 2013].

Existen cuatro posibilidades de tareas a crear: hacer una pregunta con opciones, pedir que el participante dibuje sobre una imagen para marcar elementos en ella (permite subtareas), hacer una pregunta para ser respondida con un texto, o crear una guía de identificación para clasificar elementos, como animales o galaxias.

La planificación de las tareas puede hacerse eligiendo cuál será la tarea siguiente en la creación de cada tarea. En el caso de las preguntas con opciones, se puede hacer un salto condicional de acuerdo a la respuesta. El editor permite previsualizar la actividad. También se puede crear un tutorial. Cada proyecto puede tener más de un workflow, pero cada vez que se crea uno hay que crear todas las tareas desde cero, no hay posibilidad de reusar tareas preexistentes.

En la Figura 3.2.2 se puede ver la herramienta de definición de workflows de Zooniverse.

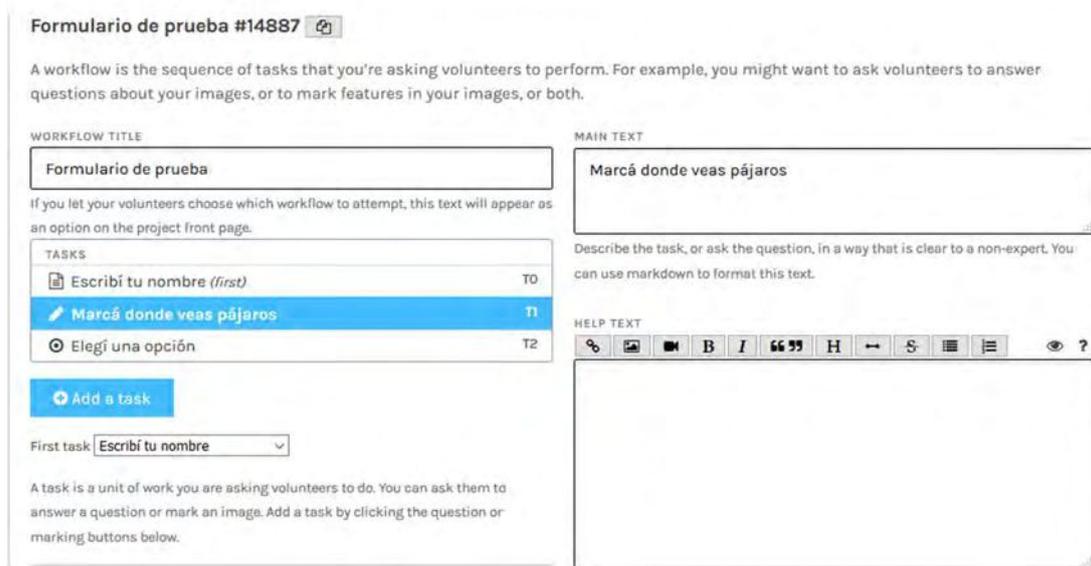


Figura 3.2.2 Herramienta de definición de workflows de Zooniverse

En cuanto a su arquitectura, Zooniverse está compuesto por múltiples sitios web que hacen uso de una librería que les permite acceder a una API REST en común, llamada Panoptes. La API a su vez guarda sus datos en una base de datos SQL y los sujetos en la nube (S3 de Amazon). También existe una aplicación móvil, que puede usarse para resolver workflows de una sola tarea (la resolución se realiza mayormente mediante los sitios web). En la Figura 3.2.3 se puede ver esta configuración.

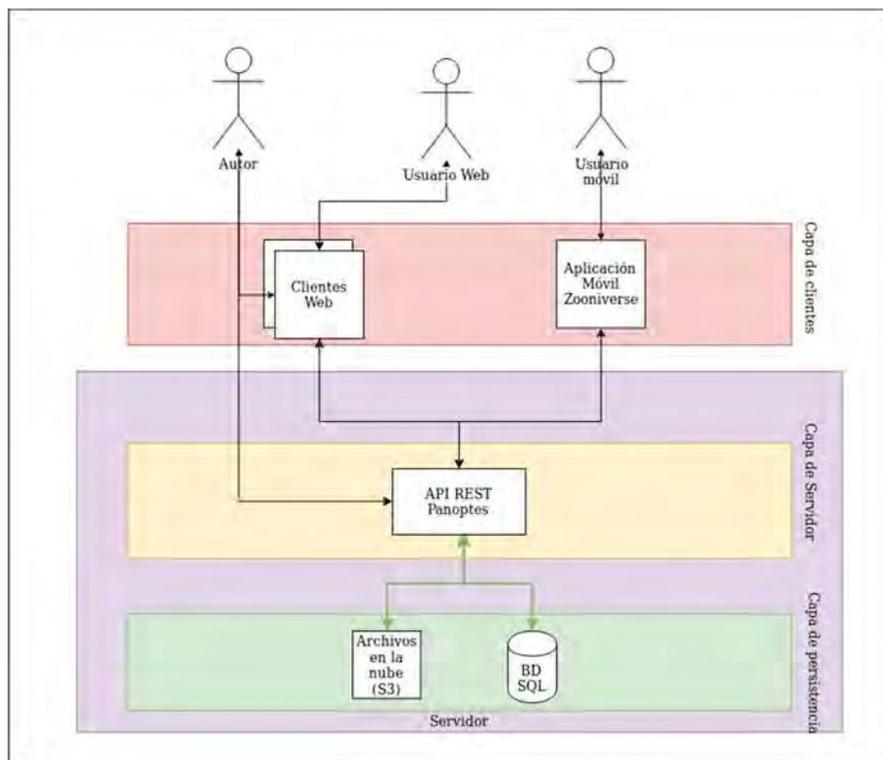


Figura 3.2.3 Arquitectura de Zooniverse

El inicio de sesión es propio de Zooniverse, pero se está trabajando en agregar login desde Facebook y Google. Desde el panel de control de Panoptes se pueden crear

credenciales para acceder a la API mediante OAuth2.0. Esta API permite operar sobre los workflows y las clasificaciones, y obtener las respuestas (serían las clasificaciones).

3.3 OPEN DATA KIT

Open Data Kit¹⁷ (ODK) es una plataforma diseñada para recolectar, administrar y usar datos de las regiones en desarrollo [Hartung, et al., 2010]. Es de propósito general: puede ser usado para censos, actividades educativas, medicina, ecología, etc.

Es un proyecto de software libre pensado de forma modular, de tal forma que sus componentes puedan ser usados juntos o separados en conjunto con otras soluciones de software. Estos módulos son los siguientes:

- ODK Build: un editor *drag-and-drop* para la creación de formularios
- ODK Collect: una aplicación móvil para la resolución de formularios
- ODK Aggregate: un servidor de almacenamiento de respuestas

Adicionalmente, se cuenta con otros módulos, entre ellos:

- ODK Central: un nuevo servidor que reemplaza a ODK Aggregate y tiene una API REST
- ODK Briefcase: una aplicación de escritorio que se conecta con ODK Aggregate y ODK Central para cargar y descargar formularios y exportar los resultados

ODK usa dos estándares para implementar sus formularios: XForm, un estándar XML del W3C¹⁸ y XLSForm, un estándar desarrollado originalmente en el Laboratorio de Ingeniería Sustentable (SEL) de la Universidad de Columbia para representar formularios mediante planillas de cálculo de MS Excel. Para crear un formulario existen dos alternativas: se puede usar el editor ODK Build, tanto en su versión web como de escritorio, que envía los formularios directamente al servidor ODK Aggregate, o crearlo en una planilla de cálculo y luego subirla a ODK Aggregate mediante su API o su interfaz gráfica. Un ejemplo de planilla en formato XLSForm se ve en la Figura 3.3.1

¹⁷ <https://getodk.org/>

¹⁸ <http://w3.org/TR/xforms>

	A	B	C
1	type	name	label
2			
3	start	start_time	
4	end	end_time	
5			
6	begin group	section1	Section 1. Record this information before approaching the household
7	integer	hhid	Please assign a unique household ID.
8	date	date	What is today's date?
9	select_one state	state	What state are you in?
10	select_one geo_list or_ogeo		Is this area rural, urban, peri-urban, or other?
11	text	address	Please enter the address or description of the home location.
12	geopoint	hhgps	Please record the GPS coordinates of this home.
13	photo	hhpicture	If available, you may want to take a photo of the front of the house.
14	end group		
15			

Figura 3.3.1 Planilla de cálculo en formato XLSForm

ODK Build es un editor gráfico de formularios que permite la creación de numerosos tipos de tareas, entre ellas elegir una opción, leer códigos de barras, tomar coordenadas, etc. Las tareas se crean de manera secuencial, pero pueden reordenarse. Se pueden poner condiciones para mostrar o no una tarea en base a lo respondido en las anteriores. Se pueden crear grupos de tareas para mostrarlas en una sola pantalla, y este grupo puede ser respondido varias veces con información diferente, generando múltiples entradas. También se pueden crear traducciones de las tareas, guardar variables, poner restricciones a los valores que se pueden ingresar y previsualizar el formulario. No permite la reutilización de tareas o planificaciones.

En la Figura 3.3.2 puede verse el editor ODK Build, creando una actividad sencilla de tres tareas.



Figura 3.3.2 Editor de formularios de Open Data Kit

ODK Aggregate es un servidor web desarrollado en Java. Tiene una interfaz gráfica que permite administrar los formularios, visualizar las respuestas y exportarlas.

La Figura 3.3.3 muestra una pantalla de ODK Aggregate visualizando los resultados de una investigación sobre bosques en forma de tabla.

The screenshot shows the ODK Aggregate web interface. At the top, there are navigation links for 'Filter Submissions' and 'Exported Submissions'. Below that, a search bar contains 'Forest Plot Survey' and a filter dropdown is set to 'none'. On the right, there are buttons for 'Visualize', 'Enketo Webform', 'Export', and 'Publish'. On the left, there are buttons for 'Save', 'Save As', and 'Delete', along with a 'Filters Applied' section. The main area is a data table with the following columns: PlotLocation Latitude, PlotLocation Longitude, PlotLocation Altitude, PlotLocation Accuracy, PlotType, PlotStatus Status, PlotStatus Logging, PlotStatus Fire, PlotStatus Grazing, PlotStatus Mining, PlotStatus Roads, PlotStatus Farming, and Tree. The table contains 12 rows of data.

PlotLocation Latitude	PlotLocation Longitude	PlotLocation Altitude	PlotLocation Accuracy	PlotType	PlotStatus Status	PlotStatus Logging	PlotStatus Fire	PlotStatus Grazing	PlotStatus Mining	PlotStatus Roads	PlotStatus Farming	Tree
50.8506638	5.7008147	98.83539432523001	24.0			no	no	no	no	yes	no	View
50.8329101	5.6479802	166.78185165597358	24.0			no	no	no	no	yes	yes	View
40.2690104	-7.993991	0.0	34.4			yes						View
33.6230314	35.9192173	0.0	64.1	1								View
												View
-34.58304059	-58.43491873	42.0	7.0	11		no		yes				View
				10		no	yes					View
				5		yes	yes	no	no	no	no	View
13.0531412	77.583671	838.0	15.17	2							yes	View
						yes	yes	yes	yes	yes	yes	View
0.460055	32.6241667	1141.6	3.9	11		yes	no	yes	no	yes	yes	View
-34.14291186666667	18.851965	6.4	5.0	2		no	yes	no	no	no	yes	View

Figura 3.3.3 ODK Aggregate

ODK Collect es la aplicación móvil creada por Open Data Kit para rellenar formularios. Es capaz de descargarlos desde ODK Aggregate y Google Drive. En este último caso los formularios están en formato XLSForm en una planilla de cálculo de Google Sheets. La planilla indica la dirección del servidor donde serán enviados los datos. La aplicación está pensada también para el caso en que los investigadores aporten los dispositivos para la recolección por parte de otras personas, por lo que tiene configuraciones como limitar la funcionalidad y proteger la configuración con una contraseña. La aplicación admite múltiples idiomas tanto para su interfaz como para las preguntas (si han sido traducidas), en base a la configuración del dispositivo. En la Figura 3.3.4 se ejemplifica el uso de la aplicación con una pregunta de elegir una opción, que incluye imágenes y sonidos.



Figura 3.3.4 Ejecución de una actividad en ODK Collect

Open Data Kit tiene una arquitectura distribuida. Sus componentes son ODK Build, ODK Briefcase y ODK Collect como clientes de ODK Aggregate (o Central, que tiene una API REST). La información se guarda en una base de datos SQL, como muestra la Figura 3.3.5:

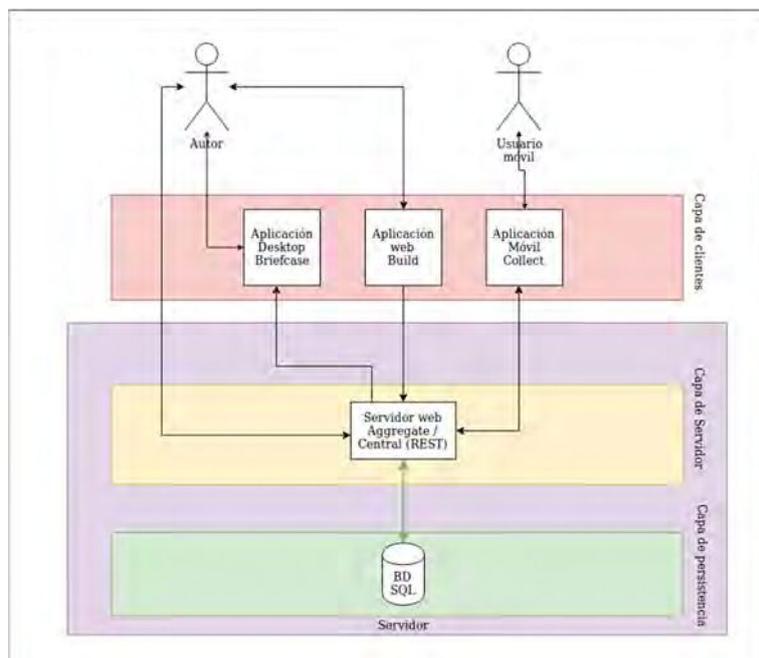


Figura 3.3.5 Arquitectura de Open Data Kit

3.4 MOLE

MoLE (*MOBILE Learning Experiences*) es una herramienta web de autor de código abierto que permite a docentes sin conocimiento sobre desarrollo móvil o programación, la creación de actividades educativas mediadas por tecnología móvil [Dal Bianco et al., 2019],[Lliteras et al., 2019].

La herramienta de creación permite crear actividades en distintos idiomas (actualmente español e inglés). Cada actividad puede tener múltiples tareas, que pueden ser de tipo “elegir una opción”, “generar contenido multimedia”, “respuesta libre”, “recolección” o “depósito”. También permite planificarlas de forma lineal, bifurcada o de conjunto.

En la Figura 3.4.1 se puede ver la herramienta de creación de actividades con múltiples tareas cargadas.

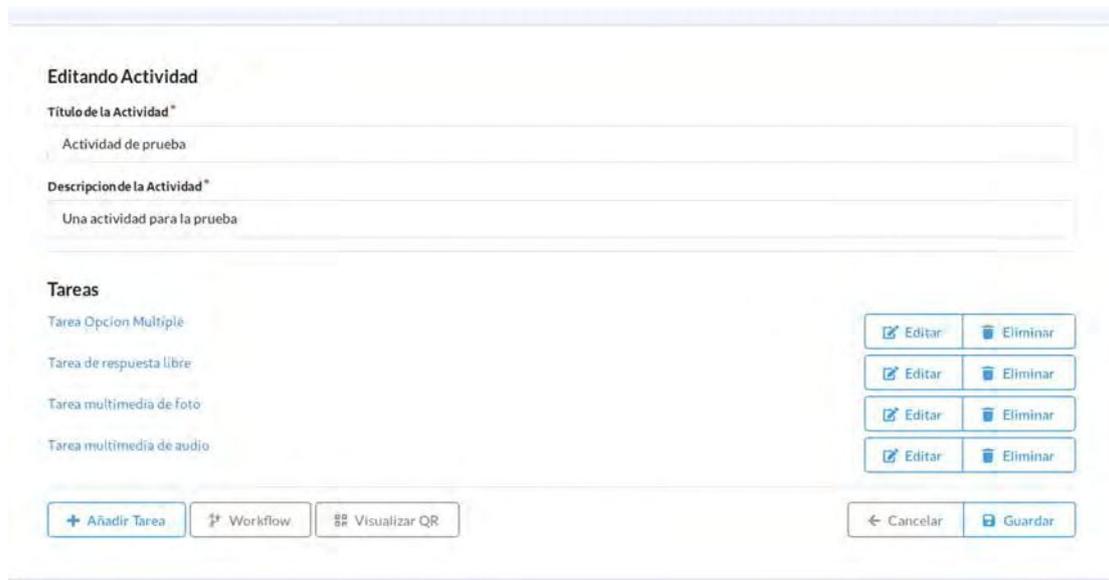


Figura 3.4.1 Herramienta de creación de actividades de MoLE

Para resolver las actividades creadas por el editor, MoLE cuenta con una aplicación móvil llamada “Resuelvo Explorando”. Para descargar una actividad es necesario leer el código QR provisto por el editor. En la Figura 3.4.2 se pueden ver algunas pantallas de una tarea de tipo “recolección”

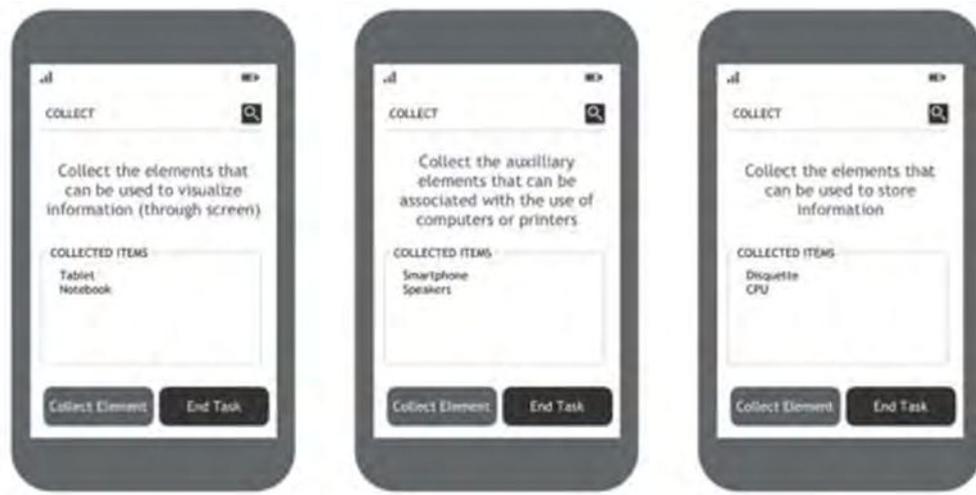


Figura 3.4.2 Actividad ejecutándose en Resuelvo Explorando [Dal Bianco et al., 2019]

La arquitectura de MoLE consiste de un sitio web y una aplicación móvil, que actúan como clientes de una API REST, que persiste los datos en una base de datos de MongoDB (NoSQL), como se ve en la Figura 3.4.3:

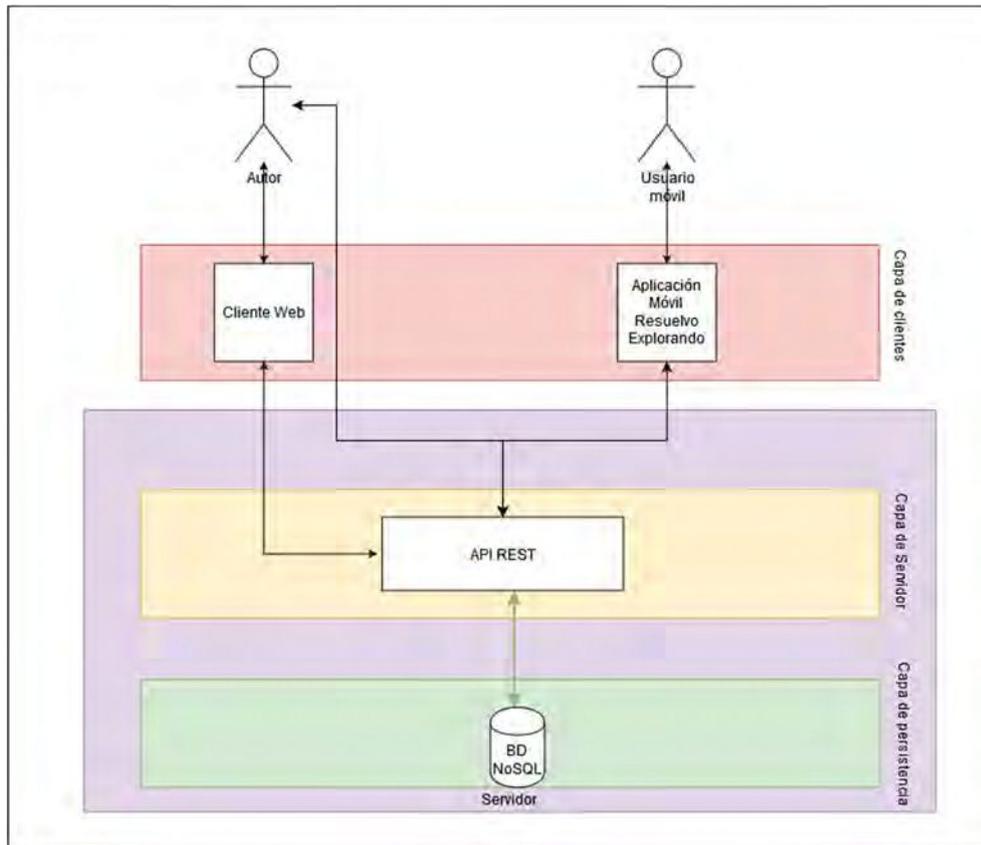
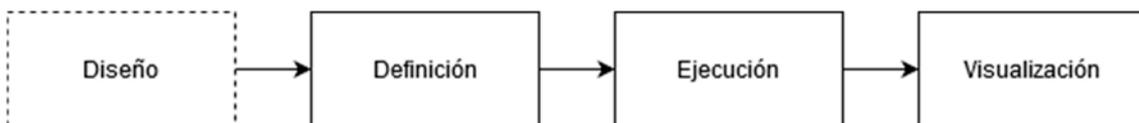


Figura 3.4.3 Arquitectura de MoLE

3.5 COMPARACIÓN DE HERRAMIENTAS

Las herramientas de autor para la generación de actividades para múltiples dominios por parte de usuarios finales comparten un flujo de trabajo en común, como se muestra en la Figura 3.5.1:



1 FIGURA 3.5.1

Etapas para la recolección de datos en estas herramientas

- **Diseño:** *Por fuera de la plataforma* se realiza el diseño conceptual de la actividad que se quiere generar. En esta etapa se deciden los objetivos y alcance del proyecto, las tareas que se van a realizar, su planificación, etc. Con esto definido se pasa a la segunda etapa.
- **Definición:** En esta etapa se usan herramientas provistas por la plataforma para ingresar en el sistema la actividad previamente diseñada. Una vez

definida, la actividad se despliega con algún mecanismo provisto por la plataforma.

- Ejecución: Los participantes ejecutan la actividad cuando llevan a cabo las tareas que se les propone. Esto genera datos que son recolectados por la plataforma.
- Visualización: Los datos recolectados se agrupan y representan de diversas formas (tablas, gráficos...) o se exportan a determinados formatos.

Estas herramientas también comparten una característica de arquitectura: todas cuentan, al menos, con un mecanismo para definir las actividades (generalmente un sitio web), una aplicación móvil para ejecutar las actividades, y un servidor para guardar la información, como se resume en la Figura 3.2.

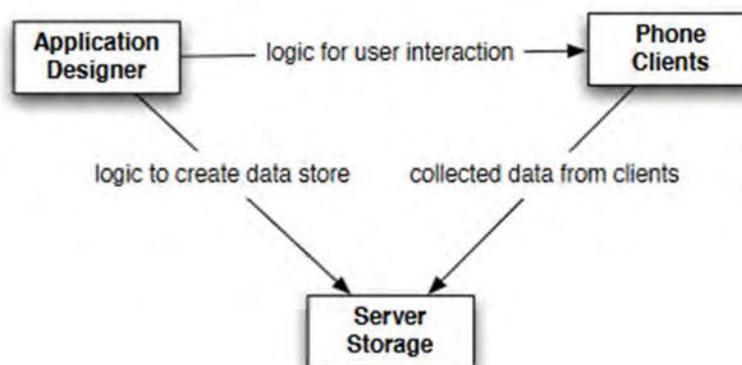


Figura 3.5.2 Componentes básicos de estas herramientas [Hartung, et al., 2010]

A continuación, se hará una comparación de los tres componentes en las cuatro plataformas descritas anteriormente. Todas las afirmaciones se hacen sobre la versión publicada de la herramienta a la fecha de redacción de esta tesina.

3.5.1 HERRAMIENTA DE CREACIÓN

Todas las plataformas cuentan con un editor gráfico para definir actividades junto con sus tareas. En cuanto a la organización de la interfaz, en todos los casos las tareas se crean de manera lineal, es decir, aparecen en forma de lista. EpiCollect5 y ODK Build presentan un mecanismo de drag-and-drop para reordenarlas. La planificación inicial es de acuerdo a esta lista, pero puede alterarse por medio de condiciones en EpiCollect5 y Zooniverse. En el caso de MoLE pueden ordenarse por medio de un grafo en la sección de Workflow.

Ninguna de las herramientas permite el uso de tareas individuales creadas con anterioridad. EpiCollect5 permite reusar actividades completas que hayan sido exportadas desde la herramienta. En EpiCollect5 algunos tipos de tarea pueden marcarse como “obligatorios”, mientras que tanto ODK Build como Zooniverse lo permiten para todas las tareas. EpiCollect5 y ODK Build permiten validar la información ingresada por el usuario.

Zooniverse y ODK Build permiten la previsualización del formulario. Zooniverse además permite la creación de un tutorial.

En cuanto a los tipos de tareas admitidos, todas las herramientas permiten respuestas de texto y selección múltiple. Excepto Zooniverse, todas permiten capturar contenido multimedia, ya sea una foto, audio o video. Zooniverse permite adjuntar contenido multimedia a las consignas. En el caso de Open Data Kit esto es posible pero no mediante el editor gráfico (se hace mediante formularios XForm o XLSForm). Una comparación de los tipos de tarea admitidos por cada editor se puede ver en la Tabla 3.5.1.1. Una cruz indica que la plataforma admite ese tipo de tarea.

Tabla 3.5.1.1 Comparación de tipos de tareas posibles.

Tipos de tarea	EpiCollect	Zooniverse	ODK Build	MoLE
Elegir una opción	X	X	X	
Múltiples opciones	X	X	X	X
Ingresar texto	X	X	X	X
Dibujar sobre una foto		X		
Clasificar elementos		X		
Ingresar un número	X		X	
Ingresar una fecha	X		X	
Ingresar una hora			X	
Ingresar una ubicación	X		X	
Subir una foto	X		X	X
Subir un video	X		X	X
Subir audio	X		X	X
Leer código de barras	X		X	
Recolectar elementos con códigos QR				X
Subir metadatos del teléfono			X	
Instrucciones	X	X	X	

Tipos de tarea	EpiCollect	Zooniverse	ODK Build	MoLE
Señalar en la imagen		X		

3.5.2 APLICACIÓN MÓVIL

Todas las plataformas cuentan con una aplicación móvil para la resolución de actividades. Zooniverse no hace uso de las posibilidades del dispositivo (ver Tabla 3.5.1.1) y limita el uso de la aplicación a aquellos workflows que tengan una sola tarea. Esto se debe a que el foco está puesto en la resolución web por medio de computadoras. En los otros casos, la aplicación móvil es la única forma de resolver las actividades. A diferencia de las otras tres, el funcionamiento de la aplicación de Zooniverse es exclusivamente online. Las demás requieren de acceso a internet para descargar las actividades y enviar los resultados, pero no para la resolución. ODK Collect y EpiCollect permiten acumular formularios respondidos sin enviar para subirlos todos juntos cuando se recupera el acceso a internet. Excepto Zooniverse todas las aplicaciones admiten múltiples idiomas.

Respecto de la obtención de actividades, Zooniverse tiene disponibles los proyectos que cumplen con ciertas condiciones que los hacen usables desde un dispositivo móvil. ODK Collect sólo permite acceso unas pocas actividades de prueba: para obtener más hay que iniciar sesión en un servidor propio de ODK Aggregate o asociar una cuenta de Google Drive para acceder a las planillas en formato XLSForm. También se pueden obtener formularios de cualquier servidor que admita los estándares manejados por Open Data Kit. En el caso de EpiCollect, se puede acceder a cualquier proyecto público. Para acceder a un proyecto privado es necesario iniciar sesión con Google. MoLE obtiene sus actividades a partir de leer un código QR provisto por la herramienta de creación. En todos los casos se puede resolver las actividades de manera anónima. La Tabla 3.5.2.1 resume estas características.

Tabla 3.5.2 Comparación de características de aplicaciones móviles

Una X indica que la aplicación móvil cuenta con esa característica

Característica	EpiCollect	Zooniverse	ODK Build	MoLE
Aprovecha las posibilidades del dispositivo	X		X	X
Resolución offline	X		X	X
Acumular entradas	X		X	
Internacionalización	X		X	X

Proyectos públicos	X	X		
Usa Google Drive			X	
Descarga con QR				X

3.5.3 SERVIDOR

Zooniverse y EpiCollect cuentan con servidores centrales públicos para guardar los proyectos creados por los usuarios. EpiCollect permite proyectos privados en su servidor. Open Data Kit promueve el uso de servidores de los usuarios, ya sea ODK Aggregate u ODK Central (un servidor algo más moderno). MoLE y Zooniverse también permite que los usuarios tengan su propio servidor, ya que son de código abierto.

Todas las plataformas cuentan con una API REST con diversos grados de apertura y funcionalidad. EpiCollect5 tiene una API de sólo lectura: sólo se puede descargar formularios y respuestas de acuerdo a las credenciales provistas (para los proyectos privados). En Zooniverse se pretende que los workflows sean creados desde la herramienta web. El resto de la funcionalidad (administración de sujetos, descarga de clasificaciones) es posible mediante la API por medio de credenciales de OAuth2.0. En Open Data Kit es posible subir formularios completos en formato XForm o XLSForm (creados o no por ODK Build). Es posible descargar las respuestas haciendo uso de credenciales (recordar que el servidor es propio). En el caso de MoLE, prácticamente todo lo que se hace desde el editor puede hacerse desde la API: creación de tareas, creación de actividades, planificación, etc. Estas características se resumen en la Tabla 3.5.3.1

Tabla 3.5.3.1 Comparación de características de las APIs

Una X indica que la API posee esa característica

Característica	EpiCollect	Zooniverse	ODK Build	MoLE
Instancia pública	X	X		
Código abierto		X	X	X
Crear workflows			X	X
Editar workflows			X	X
Obtener workflows	X		X	X

Obtener respuestas	X	X	X	X
--------------------	---	---	---	---

3.6 RECAPITULACIÓN

En este capítulo se analizaron cuatro plataformas que permiten a usuarios sin conocimientos de programación la creación de actividades para recolección y análisis de datos: EpiCollect5 (en sus tres versiones), Zooniverse, Open Data Kit y MoLE. También se hizo una descripción de los aspectos comunes entre ellas para después compararlas de acuerdo a sus tres componentes principales: la herramienta de edición, la aplicación móvil y el servidor.

En el próximo capítulo se presentará una arquitectura pensada para soportar una plataforma de este tipo.

CAPÍTULO 4. ARQUITECTURA PROPUESTA

Se propone una arquitectura para la definición y ejecución de actividades con intervención humana como soporte para proyectos de recolección y análisis de datos. Esta arquitectura es distribuida y está basada en microservicios. La suma de sus componentes permite el flujo de trabajo presentado en el Capítulo 3.

En primer lugar, se presentará una visión en capas de la arquitectura, para luego pasar a una visión por componentes.

4.1 CAPAS DE LA ARQUITECTURA

La arquitectura se estructura en cuatro capas, como se muestra en la Figura 4.1.1:

- **Capa de Clientes:** en esta capa se ubican los componentes que permiten la interacción entre los usuarios y el resto de los componentes. Puede haber varios tipos de clientes.
- **Capa de Compuertas:** en esta capa se encuentran los componentes que dirigen las peticiones de los clientes hacia los diferentes servicios, abstrayendo la implementación de estos últimos en forma de una interfaz uniforme (API), de acuerdo a las necesidades de cada cliente.
- **Capa de Servicios:** en esta capa se ubican los componentes que, de manera conjunta, procesan los pedidos de los clientes para administrar las actividades, con intervención humana como soporte para proyectos de recolección y análisis de datos, y sus respuestas.
- **Capa de Persistencia:** en esta capa se ubican los componentes que hacen permanentes los cambios realizados en los datos que administra el sistema.

Las últimas tres capas mencionadas, se ubican “del lado del servidor” (en contraposición a los clientes) y conforman el componente denominado “API”.

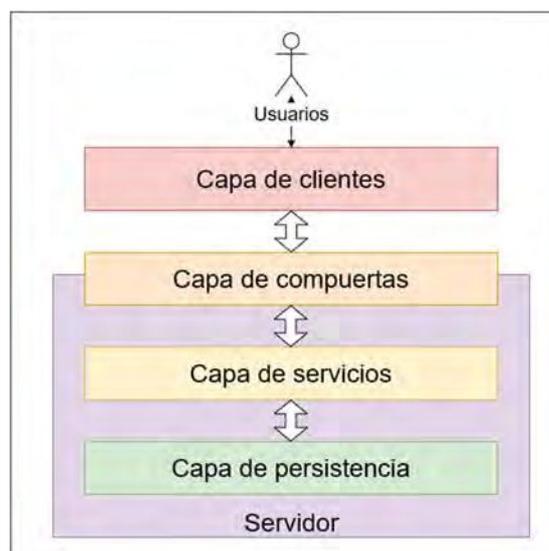


Figura 4.1.1 Arquitectura general propuesta

A continuación, se detalla el contenido de cada una de las capas de la arquitectura propuesta, para luego pasar a una descripción, de alto nivel, de los componentes principales del sistema en la Sección 4.2.

4.1.1 CAPA DE CLIENTES

Dependiendo del tipo de tecnología que se emplea, se pueden identificar cuatro posibles categorías de clientes. A continuación, se describe a cada uno de ellos:

- **Cientes Web:** Presentan una interfaz para ser ejecutada desde un navegador. Ejemplos de clientes web son: herramienta web de creación de actividades y un sitio web para resolver las actividades.
- **Cientes móviles:** Aplicaciones para dispositivos móviles. Por ejemplo, una aplicación móvil para la resolución de actividades que aproveche las posibilidades del dispositivo (por ejemplo, cámara y GPS).
- **Cientes HTTP:** Aplicaciones genéricas que permiten el acceso a APIs HTTP a usuarios avanzados (programadores), como Postman¹⁹.
- **Librerías HTTP:** Los servidores externos que pretendan hacer uso de la funcionalidad de la API pueden conectarse de manera automática mediante librerías, como Axios²⁰ para el caso de Javascript.

La Figura 4.1.1.1 muestra posibles clientes en esta capa, junto con sus posibles usuarios.



Figura 4.1.1.1 Ejemplos de clientes

¹⁹ <https://www.postman.com/>

²⁰ <https://github.com/axios/axios>

4.1.2 CAPA DE COMPUERTAS

La capa de componentes fue definida con dos objetivos:

- 1- Administrar adecuadamente los servicios de dicha capa y
- 2- Proveer interoperabilidad entre clientes y servicios.

Esta capa está motivada en dos aspectos, por un lado, la administración de servicios, y por el otro, la interoperabilidad de los clientes. A continuación, se explica cada uno de los mismos.

En relación a la administración de servicios, para que un cliente pueda acceder a la funcionalidad de la capa de servicios, debería saber a qué servicio quiere llamar y cuál es su interfaz. En una arquitectura con microservicios, el número de servicios que debe conocer el cliente tiende a crecer y ser inmanejable. Para solucionar esto se aplica el patrón de compuertas (*API Gateway*²¹), que designa un componente como intermediario entre el cliente y los servicios que abstrae al cliente del hecho de que existen múltiples servicios con, posiblemente, distintas interfaces. Mediante este componente (la compuerta), el cliente ve una sola API uniforme con la que se comunica directamente, y los servicios reciben los pedidos correspondientes con el protocolo que usen.

En términos de interoperabilidad, como se mencionó en la sección anterior, existe una variedad de clientes, que usan distintas tecnologías, queriendo comunicarse con los servicios. En este caso la interfaz que se quiere abstraer, es la de los clientes. Para lograr esto se aplica una variación del patrón anterior, llamado *Backends For Frontends*²². Este patrón propone definir una compuerta para cada tipo de interacción con el sistema. Por ejemplo, las necesidades de un cliente web y un cliente móvil no son las mismas. En la Figura 4.1.2.1 se muestran posibles tipos de compuertas para los distintos tipos de clientes y cómo los abstraen de la complejidad de los servicios.

²¹ <https://microservices.io/patterns/apigateway.html>

²² <https://microservices.io/patterns/apigateway.html>

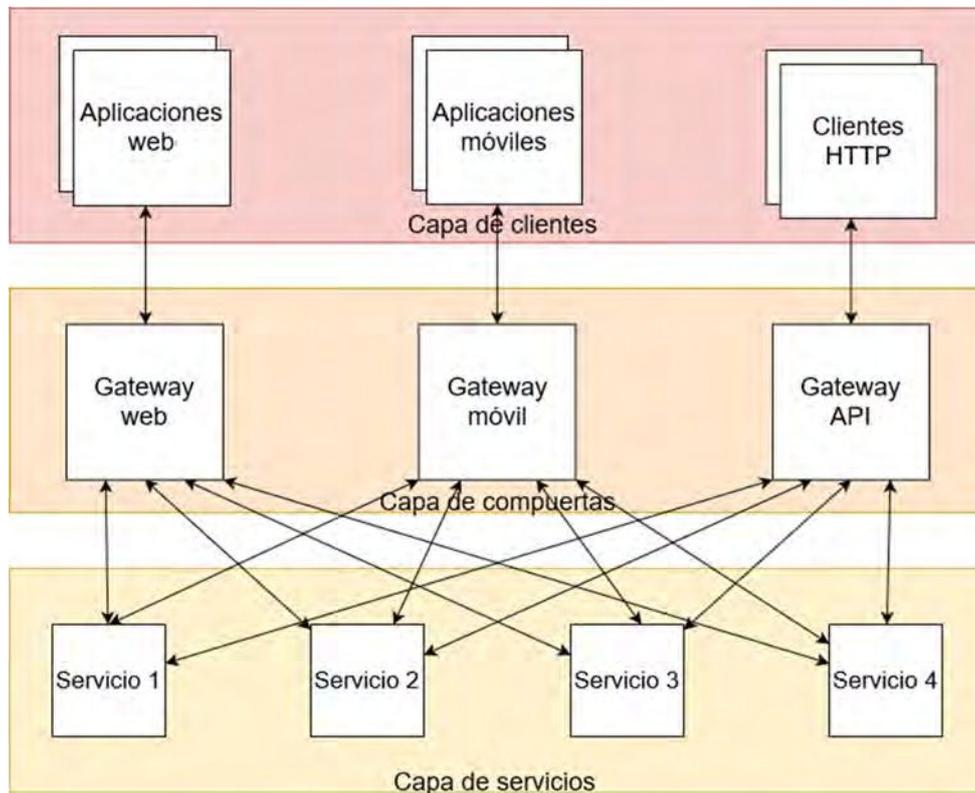


Figura 4.1.2.1 Ejemplos de compuertas (en la fila central)

4.1.3 CAPA DE SERVICIOS

Los servicios son componentes funcionales poco acoplados y altamente cohesivos [Newman, 2015]. Cada servicio tiene un rol en el conjunto y se compone con otros para lograr una funcionalidad más compleja. En la arquitectura propuesta se reconocen cinco servicios básicos:

- **Servicio de definiciones:** permite la generación de actividades
- **Servicio de ejecución:** permite ejecutar una actividad existente a través de la API
- **Servicio de recolección:** recibe las respuestas enviadas al resolver una actividad
- **Servicio de resultados:** permite obtener datos “en crudo” e información procesada acerca de las actividades ejecutadas
- **Servicio de seguridad:** provee mecanismos para identificar a quienes usan la API y autorizarlos (o no) a utilizar los diferentes servicios.

La funcionalidad detallada de estos servicios, se presentará en la Sección 4.2.5. en la que se introduce la API propuesta.

4.1.4 CAPA DE PERSISTENCIA

Los servicios necesitan de mecanismos de persistencia para administrar los datos generados por los usuarios y el mismo sistema. A continuación, se enuncian algunos de los posibles mecanismos a emplear.

- **Archivos en el servidor:** se pueden utilizar para guardar contenido multimedia, ya sea para las actividades como para las respuestas. También pueden usarse para guardar definiciones de actividades, resúmenes de resultados, etc.
- **Base de datos SQL:** se puede usar para guardar la información de definición de actividades, la ejecución de una actividad y los usuarios. Para grandes volúmenes de datos, como es el caso de las respuestas, este mecanismo podría no ser el adecuado. Puede ser una base de datos local al servidor o remota.
- **Base de datos NoSQL:** se puede usar para guardar los resultados de la ejecución de las actividades. También se puede usar para guardar información de definiciones, sin embargo, tiene limitaciones en cuanto a la complejidad de las consultas. En este caso, también se puede tener una base local o remota.
- **Servicio de archivos:** similar al primer mecanismo descrito, pero los archivos son administrados por un servicio autónomo que provee una interfaz para accederlos. Puede ser un servicio local o un servidor remoto.
- **Almacenamiento en la nube:** se puede usar para los casos de archivos, aprovechando algún producto de Infraestructura como Servicio (IaaS) como, por ejemplo, Amazon AWS²³.

En una arquitectura con microservicios, los datos son privados al servicio, es decir que no se comparten bases de datos (en realidad no se comparten *esquemas*, pero sí pueden compartirse *servidores* de bases de datos). Un servidor, puede hacer uso de múltiples sistemas de almacenamiento de acuerdo a sus necesidades. Una posible configuración de servicio y persistencia para un servicio de definiciones, se puede ver en la Figura 4.1.4.1. El servicio cuenta con *conectores*, subcomponentes que le permiten comunicarse con cada tipo de mecanismo.

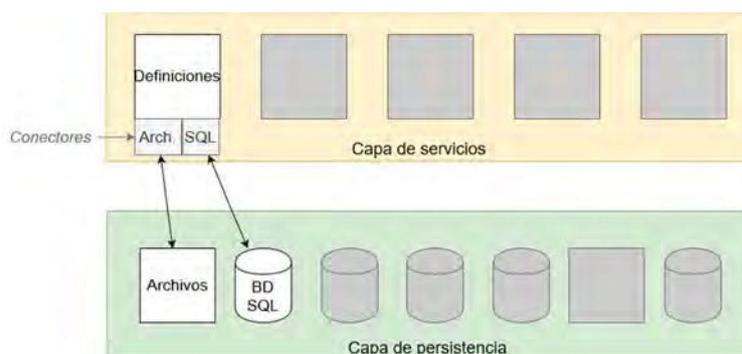


Figura 4.1.4.1 Ejemplo de mecanismos de persistencia

²³ <https://aws.amazon.com/>

4.2 COMPONENTES

Considerando los diferentes tipos de clientes y la abstracción provista por las compuertas, en una visión de alto nivel, se pueden identificar los siguientes componentes:

- Herramienta web para la definición de workflows
- Sitio web para resolver actividades
- Sitio web para ver los resultados
- Aplicación móvil para resolver actividades
- API REST

A continuación, se detalla la funcionalidad esperada de cada uno de ellos:

4.2.1 HERRAMIENTA WEB PARA LA DEFINICIÓN DE WORKFLOWS

La funcionalidad principal de esta herramienta es la de crear actividades de manera gráfica agrupando y planificando tareas. Las tareas y la estructura de las planificaciones son reutilizables, es decir, es posible usar tareas en diferentes planificaciones y planificaciones con otras tareas, de acuerdo a lo propuesto en [Lliteras, 2015]

Tanto a las actividades como a las tareas se les asigna un dominio (área en la que se desarrollan), para facilitar su búsqueda y clasificación, por ejemplo “Ecología” o “Matemáticas”. A las actividades también se les puede asignar un idioma.

Existe la posibilidad de crear actividades colaborativas en las que múltiples participantes cooperan para completar un workflow. Se pueden definir condiciones tanto para habilitar ciertas tareas como para disparar eventos que avisen al autor del estado del workflow (como, por ejemplo, cierta cantidad de resoluciones alcanzada).

Las tareas pueden ser planificadas como se explicó en el Capítulo 2, con el agregado de que se pueden poner condiciones y marcar tareas como opcionales en el marco de una planificación.

En este trabajo, se establecieron tres tipos de condiciones posibles:

1. **Condiciones basadas en la respuesta de la tarea predecesora:** Dada una tarea y un conjunto de tareas destino posibles, lo respondido en la tarea determina a cuál/es de las tareas se puede acceder a continuación. Estas condiciones incluyen haber respondido de determinada forma (o no), y haber respondido de una manera predefinida como “correcta” (o no). Se puede ver un ejemplo en la Figura 4.2.1.1

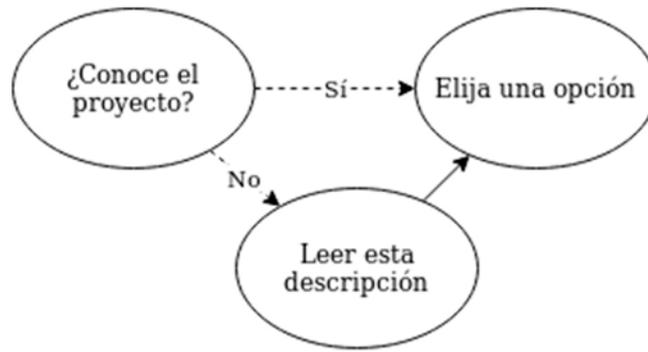


Figura 4.2.1.1 Planificación condicional basada en respuestas

2. **Condiciones basadas en el recorrido:** Dado un grafo de tareas previas y un conjunto de tareas destino posibles, el haber pasado o no por determinada tarea determina cuál/es de las tareas se puede acceder a continuación. Se muestra un ejemplo en la Figura 4.2.1.2

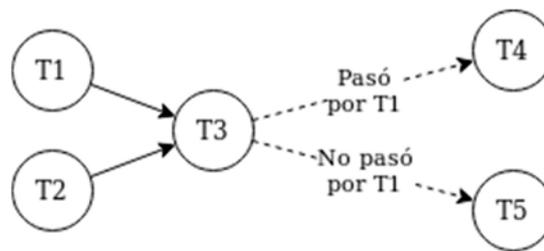


Figura 4.2.1.2 Planificación condicional basada en el recorrido

3. **Condiciones basadas en un evento externo:** Dado un conjunto de tareas posibles, el estado de un flujo colaborativo determina qué tareas se pueden acceder a continuación. Un ejemplo de este tipo de condición son que otra tarea haya alcanzado un mínimo de resoluciones. En la Figura 4.2.1.3 se puede ver un ejemplo, donde la tarea T3 no puede ser resuelta hasta que al menos 5 participantes hayan resuelto la tarea T2.

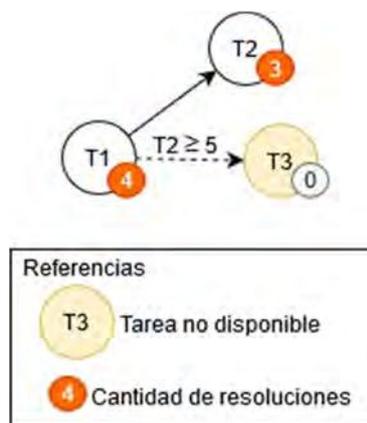


Figura 4.2.1.3 Ejemplo de condición externa en una actividad colaborativa

La plataforma web desde la que se definen los workflows, permite además consultar listados de actividades y tareas creadas previamente, sean del autor o de otros usuarios. En este sentido, tanto las actividades como las tareas pueden ser de acceso público o privado.

Por otro lado, el inicio de sesión en la herramienta, puede ser mediante alguna red social (por ejemplo, Google o Facebook) o bien, de manera anónima (como invitado). En este último caso, sólo se pueden ver los elementos creados por los autores.

4.2.2 RESOLUCIÓN WEB DE ACTIVIDADES

Este componente permite a usuarios (identificados o anónimos) resolver actividades definidas desde la plataforma web de creación.

El usuario tiene un conjunto de actividades públicas disponibles para elegir, o puede buscar una actividad privada específica y acceder a ella mediante un código.

Una vez elegida la actividad, se presenta al usuario la primera tarea a resolver, o un conjunto de tareas iniciales para elegir, como se muestra en la Figura 4.2.2.1

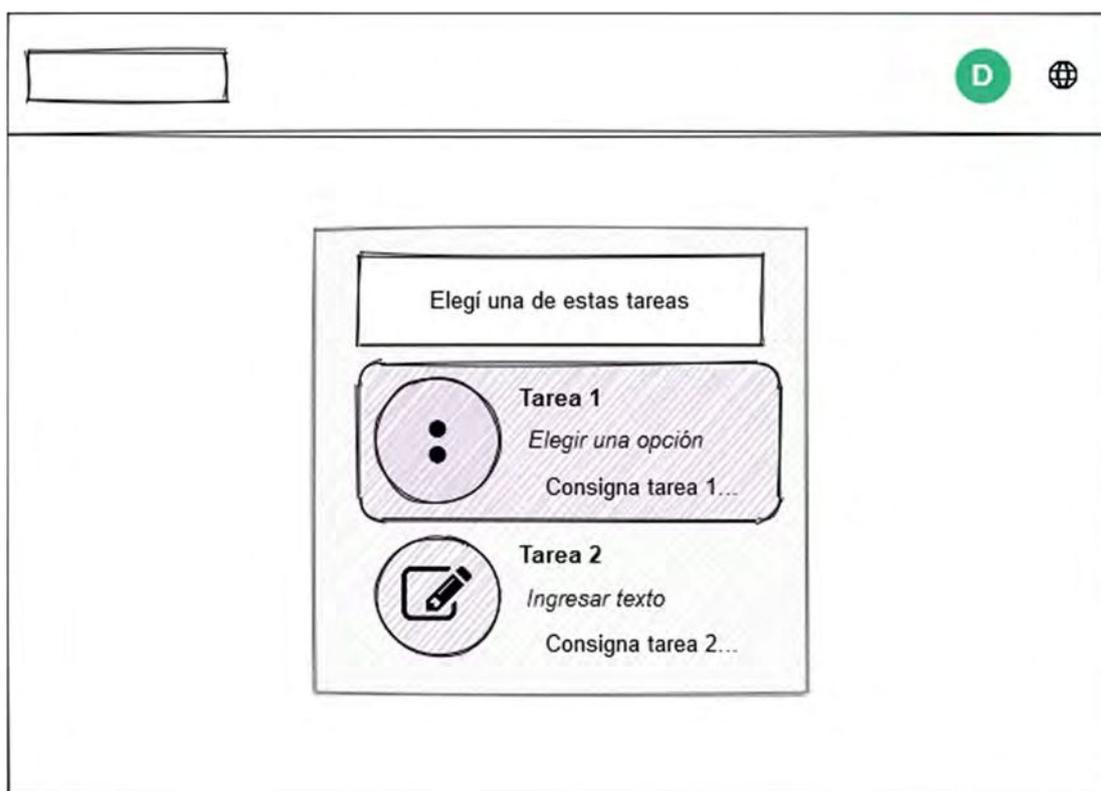


Figura 4.2.2.1 Ejemplo de pantalla con tareas para elegir

Algunos tipos de tarea requieren que el usuario suba un archivo en lugar de capturar la información en el momento (como en el caso de las actividades multimedia), y otras tareas no son posibles de realizar desde la versión web (como las actividades

posicionadas, es decir, aquellas que requieren que el usuario se encuentre en una posición física específica para realizar la tarea).

Si el usuario decide resolver una actividad colaborativa, puede ver un listado de participantes y de usuarios activos, y tiene acceso a la vista del workflow, donde puede ver la planificación de las tareas junto con sus condiciones. Un ejemplo de esto puede verse en la Figura 4.2.2.2:

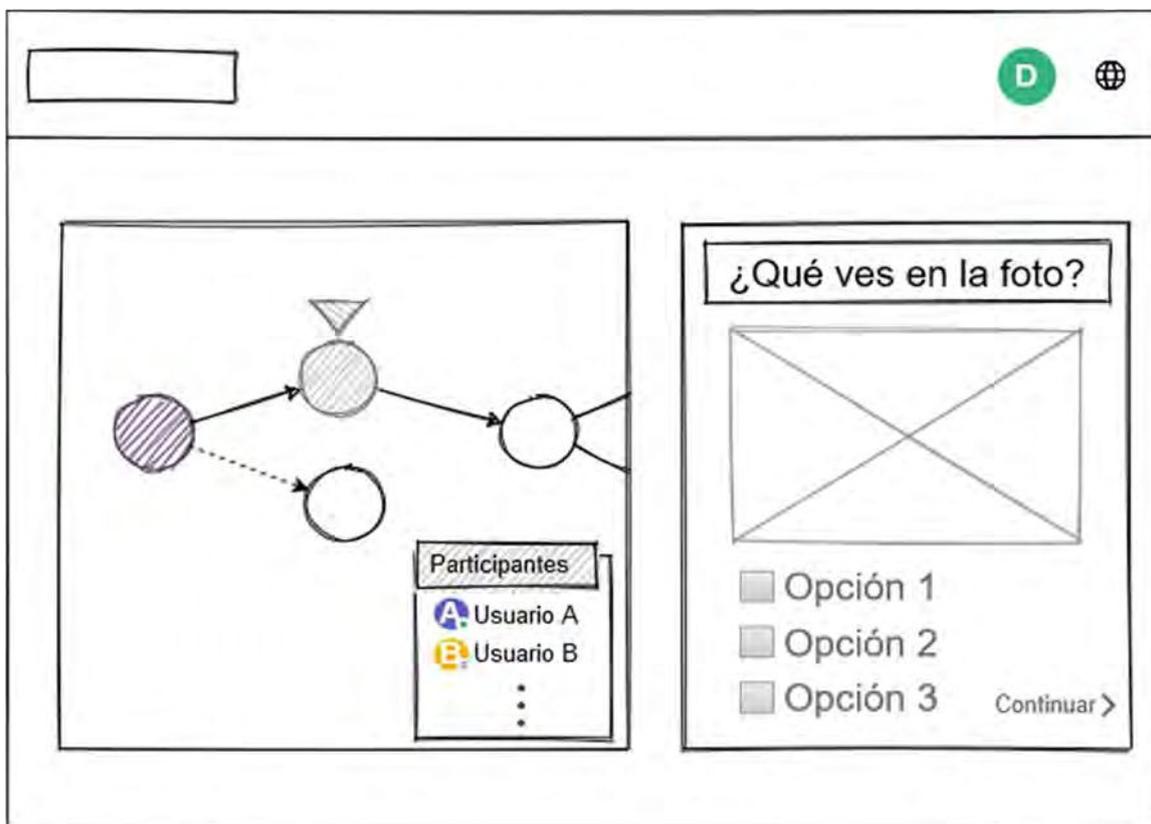


Figura 4.2.2.2 Ejemplo de pantalla con un flujo colaborativo

4.2.3 RESULTADOS WEB

Este componente permite a los autores obtener los resultados de la ejecución de actividades, tanto móvil como web. Estos resultados se pueden exportar a diversos formatos (como CSV, JSON, etc) o mostrarse de diversas formas, entre ellas:

- **Vista individual:** se muestran las respuestas de todas las tareas pertenecientes a una sola entrada
- **Vista de tabla:** se muestran las respuestas de todas las entradas de una actividad en una tabla paginada
- **Vista de gráficos:** se muestra la información de una actividad resumida en gráficos
- **Vista de mapa:** muestra las posiciones reportadas para determinada tarea

También permite revisar o descargar los archivos multimedia recibidos como respuestas.

A continuación, se muestran, en las Figuras 4.2.3.1 y 4.2.3.2, ejemplos posibles de estas vistas:

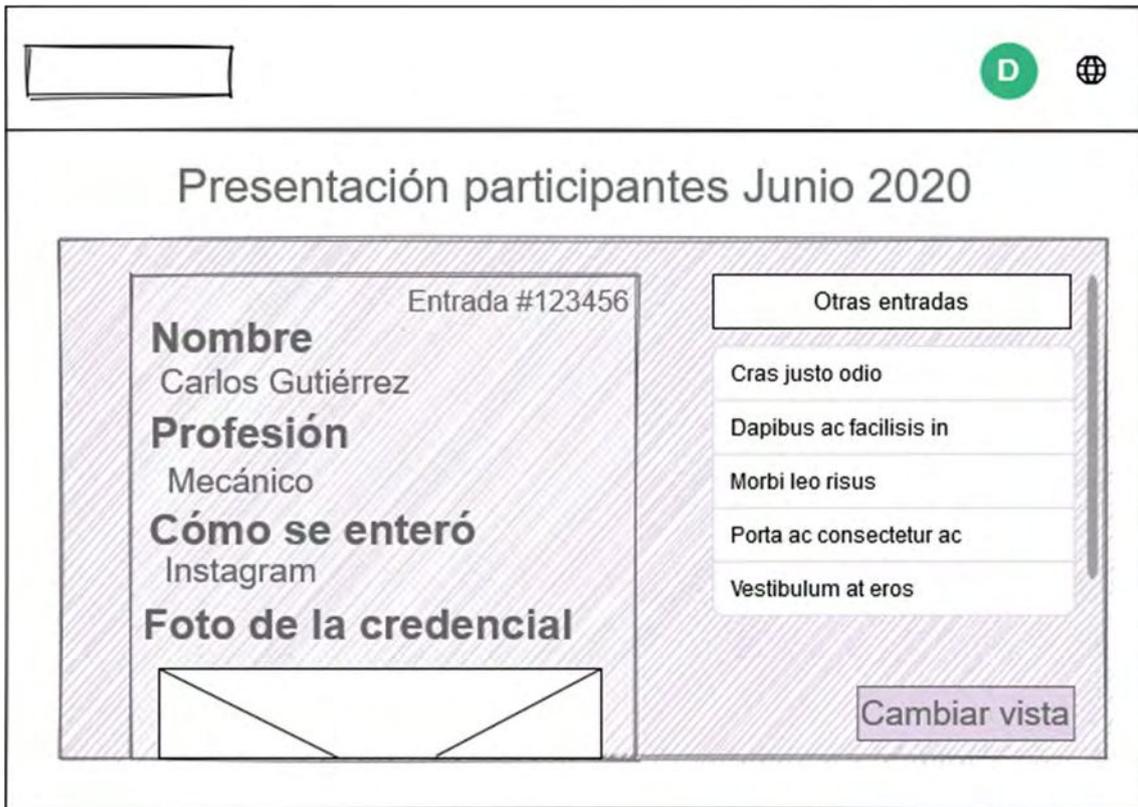


Figura 4.2.3.1 Vista de única entrada



Figura 4.2.3.2 Vista de mapa

4.2.4 APLICACIÓN MÓVIL

Este componente, al igual que el de resolución web, permite al usuario la resolución de actividades. Permite descargar múltiples configuraciones (actividades). El idioma de la interfaz está determinado por el idioma elegido para la actividad en su definición. En la Figura 4.2.4.1 se muestra un ejemplo de selección de actividad:



Figura 4.2.4.1 Selección de actividad en la aplicación móvil

Una vez que el usuario elige la actividad que quiere resolver, se le presenta la primera tarea o el conjunto de tareas iniciales disponibles. De acuerdo al tipo de enunciado y respuesta, en la pantalla se mostrará la consigna y elementos para resolver la tarea. En la Figura 4.2.4.2 se puede ver una tarea de elegir una opción.

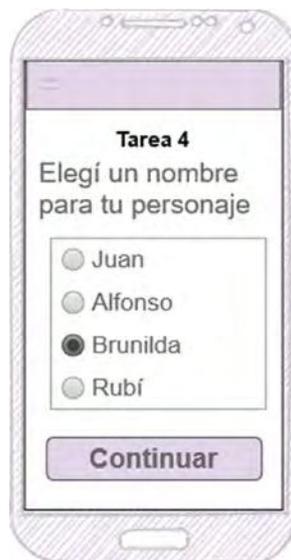


Figura 4.2.4.2 Tarea de elegir una opción en la aplicación móvil

Cuando el usuario termina la actividad puede enviar las respuestas, no siendo necesaria la conexión a internet hasta este punto.

4.2.5 API REST

Este componente tiene tres funciones principales: crear workflows de tareas, recibir las respuestas enviadas por aplicaciones la aplicación móvil o el cliente de resolución web, y devolver los resultados obtenidos. Además, identifica y autoriza usuarios de acuerdo a su nivel de acceso.

En cuanto a la creación de workflows, la API permite crear actividades y tareas, así como los dominios a los que pertenecen ambas. Permite crear planificaciones para las actividades, indicando cuáles son sus tareas sucesoras y las condiciones para acceder a cada una, cuáles son opcionales y cuáles son iniciales. Ofrece listados paginados de actividades y tareas (listado público y del autor actual), y de atributos como idioma, tipo de tarea, dominio, etc. Se puede consultar individualmente actividades y tareas (siempre que sean accesibles por el usuario) y descargar la definición de un workflow en un formato listo para importar desde la aplicación móvil.

4.3 RECAPITULACIÓN

Se presentó la arquitectura propuesta desde el punto de vista de sus capas y el de sus componentes. En el próximo capítulo se pasará a describir la solución prototípica desarrollada.

CAPÍTULO 5. LA PLATAFORMA DEHIA

De acuerdo a lo propuesto en el Capítulo 4, se desarrolló una implementación prototípica llamada DEHIA (por *Define and Execute Human Interaction Activities*). En este capítulo se tratará el alcance del prototipo, la arquitectura específica y su implementación.

5.1 ALCANCE

DEHIA es una solución prototípica que pretende cubrir los requisitos mínimos para crear y ejecutar actividades, y mostrar sus resultados.

La funcionalidad desarrollada consiste en:

- Creación de actividades, dominios, tareas y planificaciones mediante la plataforma web
 - Ejecución de actividades mediante la aplicación móvil
 - Obtención de resultados mediante la plataforma web
- En la sección 5.3 se detalla cada funcionalidad.

5.2 ARQUITECTURA

El prototipo sigue la estructura de capas que puede verse en la Figura 5.2.1 y fue presentada en el Capítulo 4. A continuación se detalla el contenido específico de cada capa.

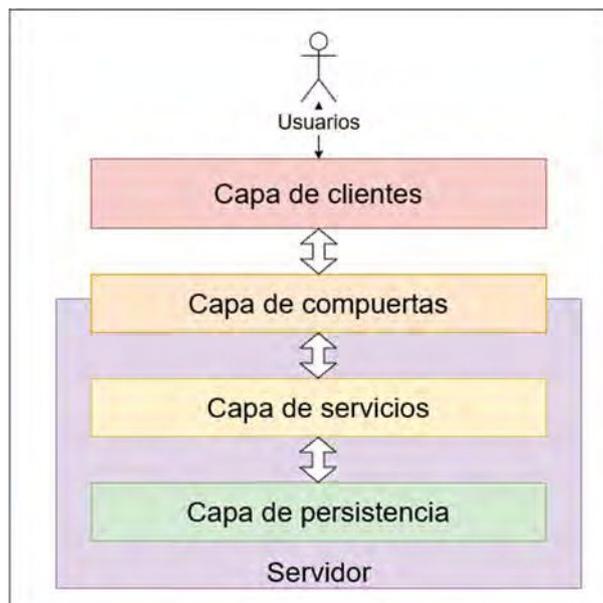


Figura 5.2.1 Arquitectura general propuesta

5.2.1 CAPA DE CLIENTES

En esta capa hay una plataforma web para la creación de actividades y posterior obtención de resultados, así como una aplicación móvil para la resolución de las mismas. En la Figura 5.2.1.1 se muestra esta situación:

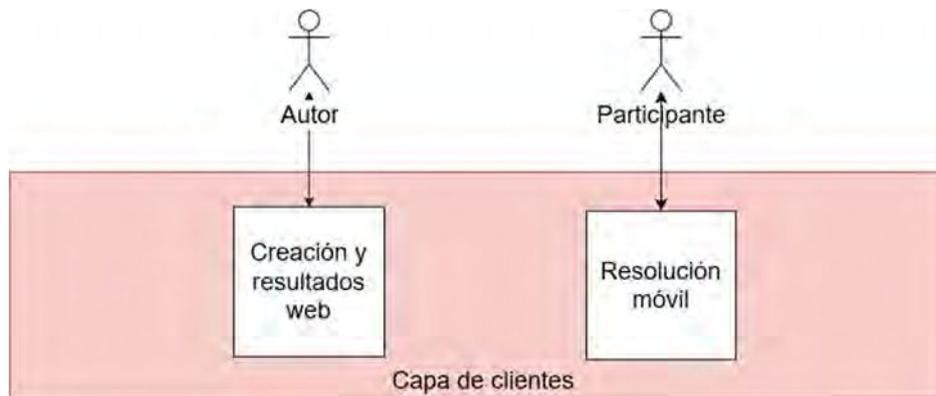


Figura 5.2.1.1 Clientes en la plataforma DEHIA

5.2.2 CAPA DE COMPUERTAS

En esta capa hay una sola compuerta, dado que las diferencias entre la comunicación entre los clientes y los servicios no justifican la creación de una compuerta separada para cada cliente. La Figura 5.2.2.1 ilustra esta disposición:

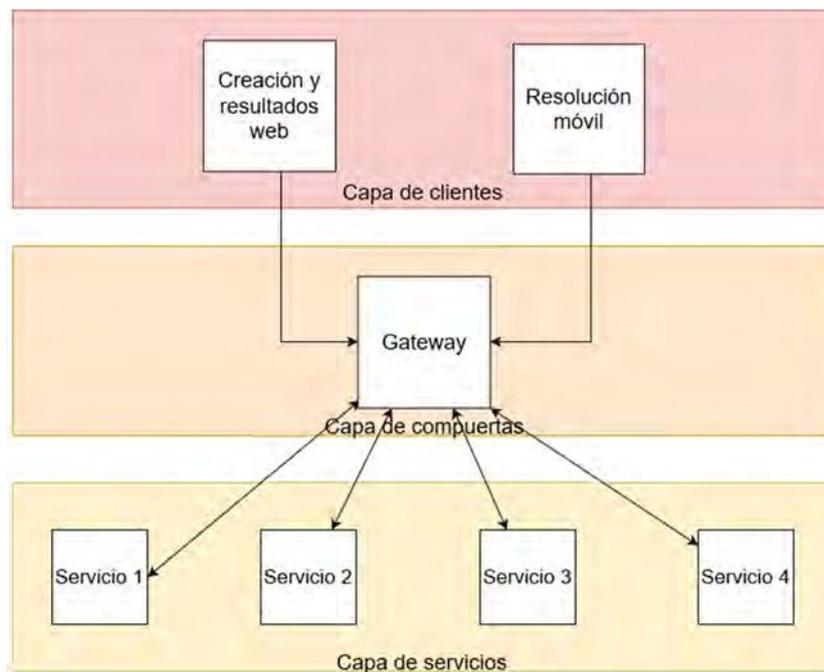


Figura 5.2.2.1 Capa de compuertas en la plataforma DEHIA (fila central)

5.2.3 CAPA DE SERVICIOS

En esta capa se encuentran los servicios de definiciones, de recolección, de resultados y de seguridad (de acuerdo a lo descrito en el capítulo anterior). Como la resolución de actividades es exclusivamente vía la aplicación móvil, el servicio de ejecución no es necesario.

5.2.4 CAPA DE PERSISTENCIA

Los mecanismos elegidos para persistir los datos de cada servicio son los siguientes:

- Para el servicio de recolección, una base de datos SQL (MySQL) y el sistema de archivos del mismo servicio
- Para el servicio de recolección, una base de datos NoSQL (MongoDB) [y el sistema de archivos del mismo servicio]
- Para el servicio de resultados, una base de datos SQL (MySQL)
- Para el servicio de seguridad, una base de datos SQL (MySQL)

En la Figura 5.2.4.1 se refleja esta elección.

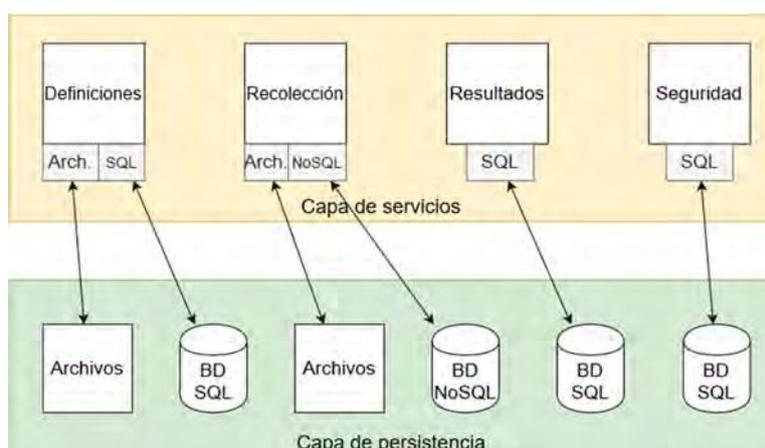


Figura 5.2.4.1 Capa de persistencia en la plataforma DEHIA (última fila)

5.3 COMPONENTES

Considerando los clientes y servicios detallados en la sección anterior, se tienen los siguientes componentes:

- Herramienta web
- Aplicación móvil
- API REST

A continuación, se detalla la funcionalidad de cada uno de ellos:

5.3.1 HERRAMIENTA WEB

La herramienta web del prototipo unifica dos de los componentes descritos en el capítulo anterior: la herramienta web para la definición de workflows y la interfaz de obtención de resultados. Por la parte de definición de workflows comprende tres

funcionalidades: la creación de actividades y tareas a partir de sus atributos, la planificación de tareas indicando su relación y opcionalmente las condiciones para pasar de una tarea a otra, y la clonación de actividades, que consiste en la reutilización de la estructura (planificación, relación entre tareas) para una nueva actividad con tareas compatibles. Por la parte de los resultados, se muestran de manera organizada las respuestas a cada tarea de la actividad que fueron enviadas desde la aplicación móvil.

Un usuario se convierte en autor al iniciar sesión con Google. La pantalla de inicio de sesión aparece en la página de inicio y en todas las secciones protegidas, y puede verse en la Figura 5.3.1.1:



Figura 5.3.1.1 Pantalla de inicio de sesión

Una vez iniciada la sesión, un autor tiene acceso al directorio de actividades públicas, a la lista de sus actividades creadas (lo mismo para las tareas) y a la creación de actividades y tareas. En la Figura 5.3.1.2 se pueden ver las diferentes opciones en el Menú:

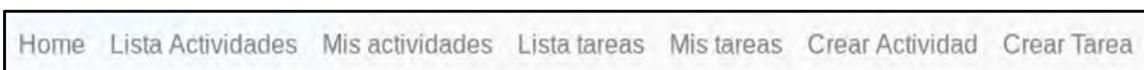


Figura 5.3.1.2 Menú de la herramienta web

Creación de planificaciones con actividades

En el directorio público se muestran las actividades públicas y definitivas. Una actividad se crea como “nueva”. Cuando el autor quiere comenzar a aceptar respuestas debe ponerla como “definitiva”. Si quiere dejar temporalmente de aceptar respuestas, puede ponerla como “cerrada”. Un diagrama de estados mostrando este comportamiento puede verse en la Figura 5.3.1.3:



Figura 5.3.1.3 Diagrama de estados de una actividad

Se pueden crear tareas de antemano, usando la opción de Crear Tarea en el menú. Una tarea tiene un conjunto de atributos comunes, y otro de atributos específicos de acuerdo a su tipo. En la Figura 5.3.1.4 se muestran los atributos extra (elementos y válidos, en este caso) de una tarea de tipo opción múltiple:

Figura 5.3.1.4 Pantalla de creación de tareas (opción múltiple)

Los tipos de tarea disponibles son:

-  Simple
-  Ingresar texto
-  Ingresar número
-  Sacar foto
-  Elegir una opción
-  Opción múltiple
-  Contadores

 *Recolección*

 *Depósito*

 *Localización*

 *Grabar audio*

Para crear una actividad, primero se cargan sus atributos, entre ellos el dominio al que pertenece, el idioma y el tipo de planificación. En una segunda pantalla, se eligen las tareas que van a formar parte de la actividad. Se puede buscar por nombre y filtrar según si tienen o no opciones (esto sirve para poner condiciones). La pantalla de selección de tareas puede verse en la Figura 5.3.1.5



Figura 5.3.1.5 Pantalla de selección de tareas

Una vez elegidas, las tareas pueden reordenarse arrastrándolas hacia arriba o hacia abajo. Este orden indica el lugar en que aparecen en la pantalla de planificación. Según el tipo de planificación, se pasa a una pantalla con creación gráfica del workflow (bifurcada) o una pantalla simple de selección de tareas opcionales (libre, secuencial - véase Figura 5.3.1.6)

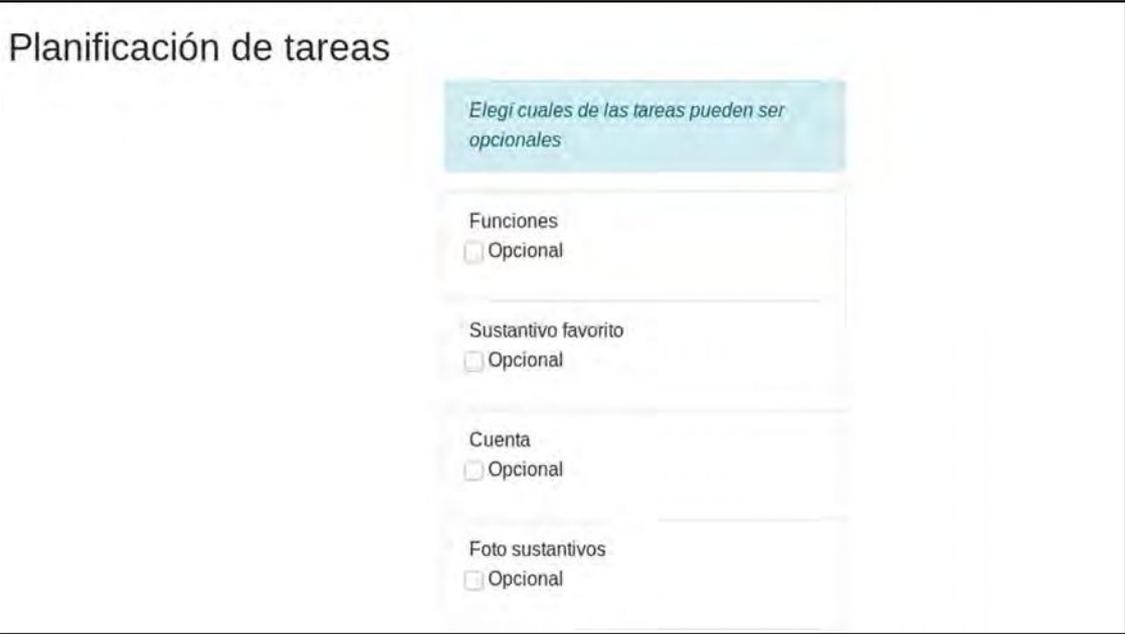


Figura 5.3.1.6 Pantalla de simple de selección de tareas opcionales

En la pantalla de creación de workflows, las tareas se representan con círculos: un círculo con el borde verde representa una tarea inicial, con el borde rojo, una tarea final y caso contrario, una tarea intermedia. Las tareas opcionales aparecen en color gris. Las conexiones entre tareas (flecha) representan la relación de predecesor-sucesor. Si la flecha tiene un círculo, es una conexión directa, si tiene un cuadrado la conexión es condicional. En la Figura 5.3.1.7 se muestra un ejemplo de workflow siendo editado en esta pantalla.

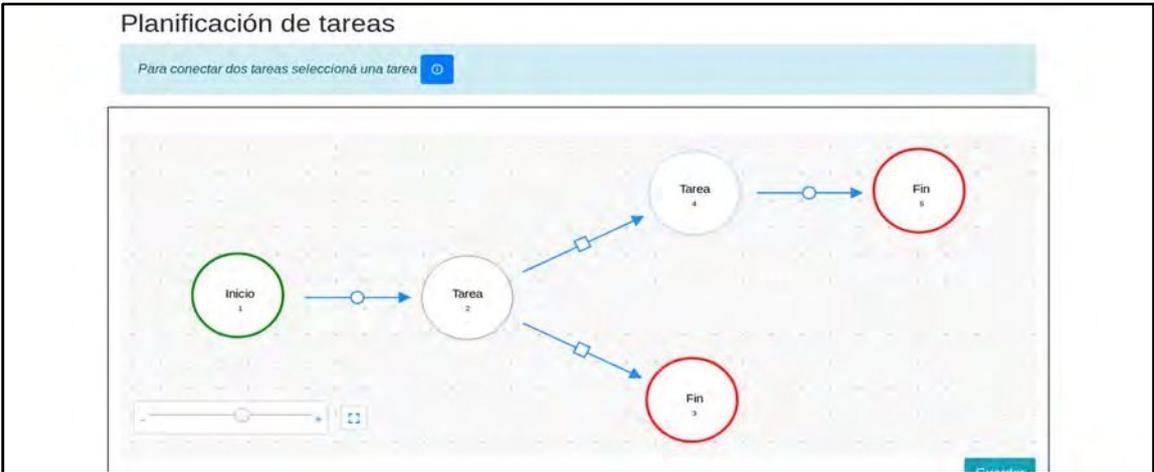


Figura 5.3.1.7 Editor de workflows

Inicialmente todas las tareas figuran como iniciales y obligatorias. Haciendo click en una tarea, se abre un menú desde el cual se puede elegir si la tarea es inicial u opcional, y crear conexiones hacia las tareas sucesoras, indicando la condición. Las condiciones

admitidas son “eligió/no eligió la respuesta X”, “pasó/no pasó por la tarea X” y “eligió/no eligió todas las opciones correctas”. El menú puede verse en la Figura 5.3.1.8

The screenshot shows a mobile application interface with a title bar that reads "Tarea seleccionada: 2. Tipo de Donación" and a close button (X). Below the title bar, there is a section titled "Elegir el tipo de donación" containing two radio button options: "Opcional" and "Inicial". Below this is a section titled "Agregar conexión" which includes a dropdown menu for "Hacia la tarea..." with the selected option "3. Sobre el teléfono". There is a checked checkbox for "Mostrar condición" and a dropdown menu for "Cuando..." with the selected option "Si se elige". Below these is a dropdown menu for "la opción" with the selected option "Institucional". At the bottom of the form is a green button labeled "Agregar conexión" with a mouse cursor pointing to it.

Figura 5.3.1.8 Menú para agregar conexión hacia una tarea sucesora

Cuando se termina la planificación, ésta se persiste y se pasa a la pantalla de detalles de la nueva actividad. Desde esta pantalla se puede poner la actividad como definitiva, cerrarla de manera temporal o cambiar su estado entre Privado y Público. También se puede acceder a la pantalla de resultados y a la definición en formato JSON para ser descargada en la aplicación móvil. Se muestran los atributos de la actividad, así como sus tareas, enlazando a las pantallas de detalles de cada una, y la planificación gráfica, en caso de ser de planificación bifurcada. En la Figura 5.3.1.9 se muestra esta pantalla.

Actividad

Nombre: Entrega de elementos en desuso
Objetivo: Notificar a E-Basura de la entrega de equipamiento
Idioma: Español
Dominio: E-Basura
Tipo de planificación: Bifurcada

[Descargar](#) [Ver resultados](#)

[Hacer privada](#) [Desactivar temporalmente](#)

Actualmente:

- Aparece en el listado público
- Se pueden enviar respuestas

Tareas

- 1. Persona de contacto
- 2. Tipo de Donación
- 3. Sobre el teléfono de contacto
- 4. Teléfono de contacto
- 5. Correo electrónico de contacto
- 6. Tipo de Institución
- 7. Otra Institución
- 8. Nombre de la institución
- 9. Provincia
- 10. Localidad
- 11. Dirección Institucional

Figura 5.3.1.9 Pantalla de detalles de actividad

Por último, en las planificaciones bifurcadas, se muestra un botón para crear actividades nuevas reutilizando la estructura (clonar planificación).

Clonación de planificaciones con actividades

El proceso de clonación es similar al de creación: primero se eligen los atributos (excepto el tipo de planificación, que tiene que ser “Bifurcada”), luego se eligen las tareas. En la pantalla de selección de tareas se listan las tareas que deben reemplazarse indicando si tienen o no opciones (de forma que al reemplazarla se puedan poner condiciones si las había). En particular, si hay una conexión con condición (de tipo “eligió/no eligió...”) que sale desde una tarea, la tarea que la reemplaza va a tener opciones de manera obligatoria. Esta pantalla puede verse en la Figura 5.3.1.10.

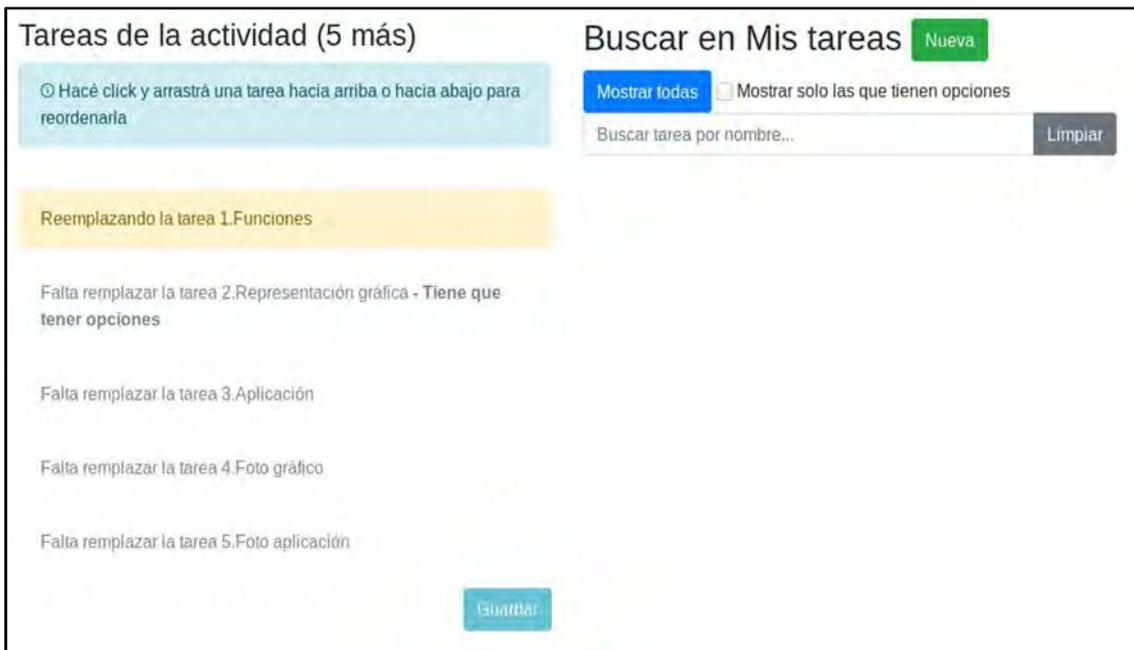


Figura 5.3.1.10 Pantalla de selección de tareas (clonando planificación)

Una vez elegidas todas las tareas, que deben ser exactamente la misma cantidad que en la actividad original, se pasa a la pantalla de planificación para ajustar las condiciones: se conserva el tipo de condición, pero se debe elegir la respuesta o tarea en base a la cual se toma la decisión. Las condiciones pendientes de ajuste aparecen marcadas con una cruz en la conexión, como se muestra en la Figura 5.3.1.11

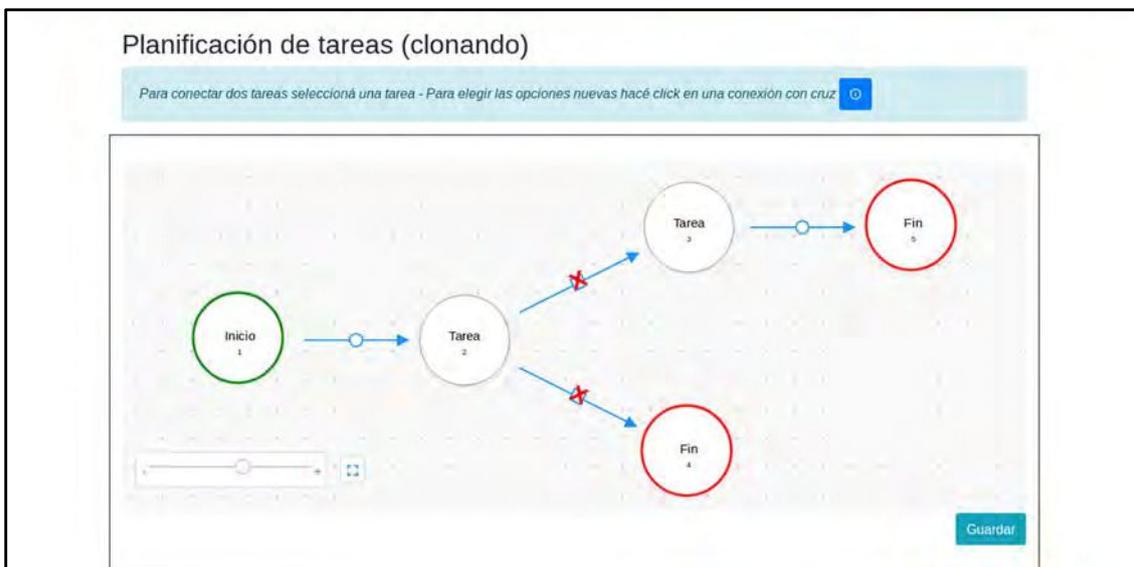


Figura 5.3.1.11 Conexiones condicionales pendientes de ajuste

Una vez que una actividad fue puesta como definitiva, se comienzan a aceptar respuestas. Los resultados pueden verse en la pantalla de resultados, en forma de tabla. Cada respuesta a una actividad ocupa una fila de una tabla, y cada tarea es una columna, como puede verse en la Figura 5.3.1.12.

Respuestas							
Código de respuesta	Marca de tiempo	Nombre	Funciones 1	Funciones 2	Funciones 3	Funciones 4	Funciones 5
eeb9a890a0	2020-05-01 09:20:39	Laura Martínez	$f(x) = 2x + 4y$		9	$f(x) = 2$	
eeb9a890a1	2020-05-01 09:30:17	Carlos Ramírez	$f(x) = 2x + 1$		19	$f(x) = 92$	
eeb9a890a2	2020-05-01 09:32:44	Marta Sánchez	$f(x) = 2x + 5$		42	$f(x) = 29$	

Figura 5.3.1.12 Pantalla de resultados

5.3.2 APLICACIÓN MÓVIL

La funcionalidad de la aplicación móvil consiste en ejecutar las actividades que tiene configuradas, mostrando las distintas tareas y enviando las respuestas a la API. Las definiciones de actividad se cargan en formato JSON a través de una carpeta específica en el almacenamiento interno del teléfono. La actividad se descarga desde la aplicación web y se guarda en el teléfono mediante una conexión USB, o se envía por email, por ejemplo, para luego ser guardada en la carpeta DEHIA del teléfono. En esta carpeta se puede tener múltiples archivos que son reconocidos por la aplicación cuando ésta se inicia. La primera pantalla de la aplicación es un selector que muestra los nombres de las actividades disponibles, como se muestra en la Figura 5.3.2.1.

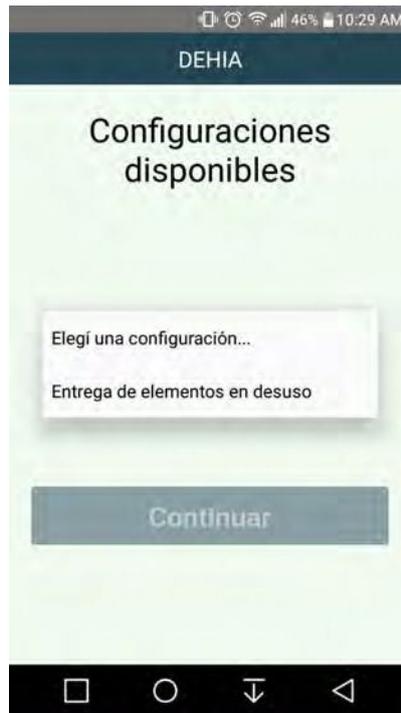


Figura 5.3.2.1 Selección de actividad

Una vez seleccionada una actividad, la interfaz adopta el idioma indicado en el JSON y se muestra una pantalla de bienvenida indicando el nombre y el objetivo de la actividad. Después, de acuerdo al tipo de planificación, se muestra la primera pantalla de tareas. En el caso de las actividades de planificación libre, se muestran todas las tareas disponibles en una lista (Figura 5.3.2.2). En el caso de las actividades secuenciales se muestra directamente la primera tarea a realizar. En el caso de las actividades de planificación bifurcada, en caso de haber más de una tarea inicial se muestra una lista con sus nombres, caso contrario comienza igual que una actividad secuencial.



Figura 5.3.2.2 Lista de tareas en una actividad de planificación libre

La pantalla cambia de acuerdo al tipo de tarea: en una tarea de tipo “simple” simplemente se muestra el nombre de la tarea y una consigna; en una tarea de “introducir texto” además hay una caja de texto para escribir; y tipos de tareas más complejos tienen pantallas específicas. Como ejemplo se muestran las pantallas para elegir una opción (Figura 5.3.2.3), indicar una posición (Figura 5.3.2.4) y grabar audio (Figura 5.3.2.4). En algunos casos hay una devolución, ya sea de cuál era la respuesta correcta o, en el caso del tipo de tarea “contadores”, el total obtenido en base a multiplicar el peso por las cantidades indicadas (por ejemplo, si el peso indica “cantidad de proteínas por gramo” en diferentes alimentos, al indicar los gramos de cada alimento se devuelve el total por alimento y el total de la suma).



En el caso de las actividades de planificación libre, al terminar una tarea siempre se vuelve a la lista, donde se van coloreando las tareas ya realizadas. Cuando se terminaron todas las tareas obligatorias se puede terminar la actividad. Las actividades secuenciales muestran una tarea detrás de otra, y la actividad finaliza cuando se termina la última tarea. En el caso de las actividades bifurcadas, de acuerdo a lo respondido y las condiciones indicadas en su definición, se van mostrando las tareas correspondientes. En caso de haber múltiples tareas sucediendo a cierta tarea, se las muestra en una lista. Cuando se llega completa una tarea final se da por terminada la actividad.

Una vez terminada la actividad (notar que no se requiere conectividad hasta este punto, excepto para mostrar el mapa en las tareas de localización) se pide al participante que envíe la resolución de su actividad, como se muestra en la Figura 5.3.2.6.

Después de enviar las respuestas, se vuelve a la pantalla de selección de actividad.

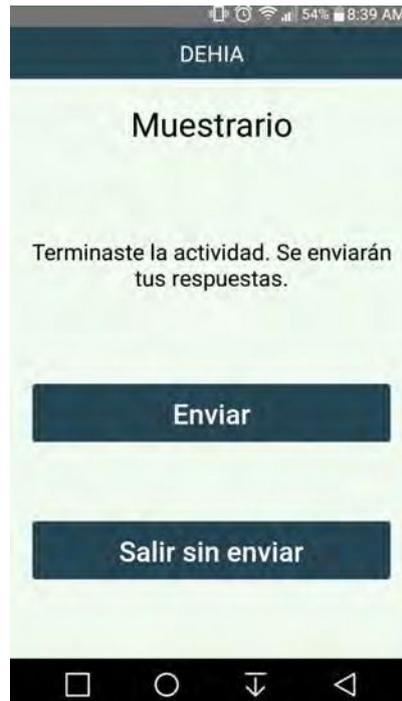


Figura 5.3.2.6 Pantalla de envío de respuestas

5.2.3 API

La API consiste en una interfaz REST que implementa la funcionalidad expuesta por la aplicación web. La definición completa de su protocolo puede verse en el Anexo A. Permite crear actividades en base a sus atributos y tareas con información de acuerdo a su tipo. También permite asociar las tareas con una actividad (las tareas pueden asociarse con múltiples actividades) y planificarlas, indicando cuáles son opcionales y, en el caso de las actividades bifurcadas, indicando también las tareas iniciales y las condiciones. Las condiciones de la planificación se indican mediante “saltos”, que tienen una tarea de origen y una o varias tareas destino, directos o con un tipo de condición. En este último caso de ser necesario hay que indicar el atributo de referencia que determina la condición, por ejemplo, cuál respuesta es la que hay que elegir o no, o por qué tarea hay que pasar (o no). Las condiciones de tipo “todas las respuestas correctas” y “no todas las respuestas correctas” no necesitan una referencia. Cuando se termina de planificar, se debe indicar a la API que la actividad está en su estado definitivo para comenzar a aceptar respuestas. Se puede indicar que la actividad está cerrada para dejar de recibir respuestas temporalmente.

Para evitar repetición de información, la mayoría de los atributos se indican por su código identificador, que se puede obtener consultando los listados de idioma, dominio, etcétera.

Se pueden obtener listados paginados de tareas y actividades, y filtrarlos por nombre. En el caso de las tareas, se puede discriminar si tienen opciones, lo que las hace adecuadas para poner condiciones al planificar.

La clonación de tareas, si bien está soportada, no requiere ninguna implementación, porque basta con copiar una petición de planificación y reemplazar las tareas y las condiciones correspondientes (puede ser algo engorroso sin la interfaz gráfica, pero es posible).

Otra de las funcionalidades de la API es la de recibir las respuestas enviadas por la aplicación móvil. Una respuesta consiste en un código de actividad y una serie de pares (código de actividad, respuesta), donde las respuestas predefinidas se indican mediante códigos para evitar errores y facilitar el procesamiento. La API conoce de antemano las tareas de la actividad, y descarta las respuestas que no correspondan a una tarea válida. Además, sólo se reciben respuestas si la actividad es definitiva y no fue temporalmente cerrada.

En cuanto a la funcionalidad de consulta de resultados, dado un código de actividad devuelve el listado de “entradas” (cada instancia de respuesta a una actividad) que le corresponde, donde cada una consiste en un listado de pares (nombre de tarea, respuesta).

Por último, la API se encarga de identificar a los usuarios mediante un token de identificación de Google enviado desde la aplicación web. Algunas consultas, como los listados de atributos y actividades públicas, son accesibles sin necesidad de autenticarse, mientras que, para las acciones de creación y modificación, u otras consultas restringidas son accesibles sólo para los autores registrados.

5.4 MODELO DE DATOS

En esta sección se presenta el modelo conceptual de datos para la plataforma. Los datos están repartidos entre los diferentes servicios, y en ocasiones hay información redundante para evitar comunicación innecesaria entre servicios. La gran mayoría de las bases de datos utilizadas son relacionales. La excepción es la base de datos de respuestas, que usa una base de datos no relacional, orientada a documentos.

El diagrama de la Figura 5.4.1 muestra una vista general de la organización de los datos.

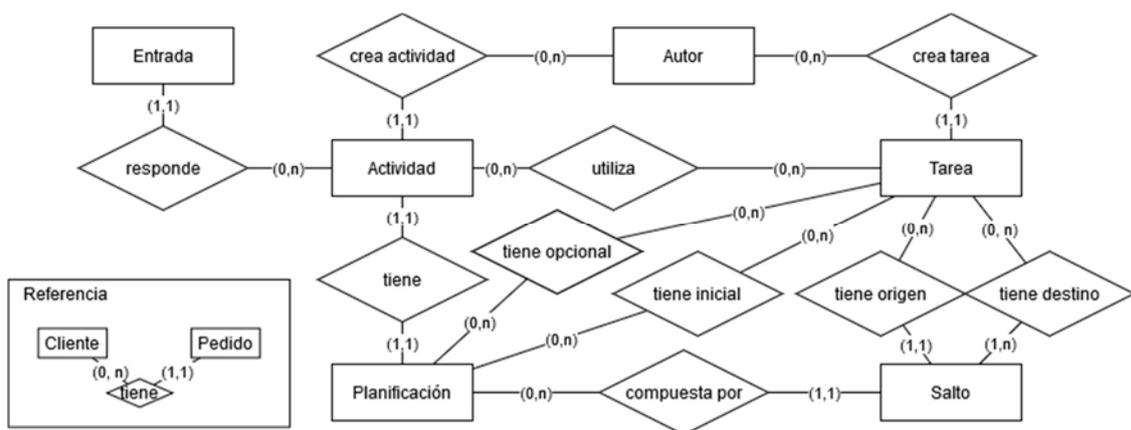


Figura 5.4.1 Modelo de datos de la plataforma (relacional)

En la Figura 5.4.2 se pueden ver los documentos de la base de datos de respuestas.

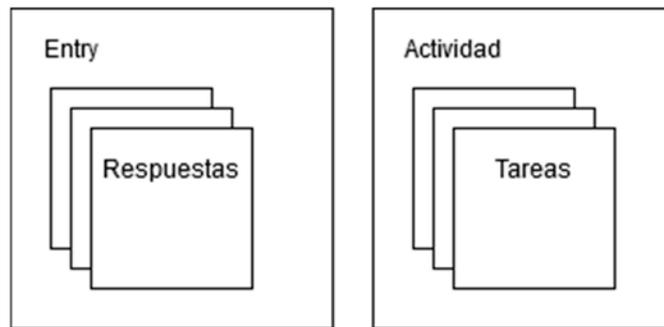


Figura 5.4.2 Modelo de datos de la plataforma (no relacional)

5.5 TECNOLOGÍAS UTILIZADAS

En esta sección se detalla para cada componente propuesto el framework utilizado para su desarrollo. La lista completa de bibliotecas utilizadas para cada uno puede verse en el Anexo B.

5.5.1 APLICACIÓN MÓVIL

La aplicación móvil fue desarrollada con el framework React Native²⁴, que permite la creación de aplicaciones móviles para iOS y Android mediante código javascript. Tiene la ventaja de ser multiplataforma y resulta sencillo de usar para quienes tienen experiencia en React (web) por su similitud y sintaxis compartida.

5.5.2 APLICACIÓN WEB

La herramienta web de creación de actividades, tareas y workflows fue desarrollada con React²⁵, un framework de javascript. Se eligió esta tecnología por su similitud con React Native, que ya se había utilizado para la aplicación móvil. React es una tecnología relativamente moderna y muy usada, y favorece la reutilización de componentes.

Se utilizó una herramienta llamada Cypress²⁶ para realizar tests de interfaz gráfica. El contenido de los tests puede verse en el Anexo C. Esta herramienta permite manipular los elementos del DOM mediante selectores similares a los de CSS y chequear condiciones como la presencia de determinado elemento o la ejecución de un determinado request.

5.5.3 COMPUERTA

La compuerta o *gateway* que comunica los clientes con los servicios fue desarrollada con ExpressJS²⁷, un framework que corre sobre NodeJS que es un entorno

²⁴ <https://reactnative.dev/>

²⁵ <https://reactjs.org/>

²⁶ <https://www.cypress.io/>

²⁷ <https://expressjs.com/>

para ejecutar código JS del lado del servidor. Esta tecnología fue elegida por su capacidad para manejar un gran número de peticiones concurrentes.

5.5.4 SERVICIOS

Todos los servicios fueron implementados utilizando Symfony²⁸, un framework para PHP que mediante el uso de bibliotecas permite la creación de aplicaciones web. En este caso no fue necesario el uso de vistas (dado que es una API) pero también están soportadas. Uno de los beneficios de Symfony es su documentación clara y detallada, y la gran cantidad de bibliotecas que pueden encontrarse.

Tanto para hacer testing de los *controllers* (es decir pruebas “unitarias”) como de endpoints completos (pruebas “de integración”) se utilizó PHPUnit. El contenido de estos tests puede verse en el Anexo C. Esta herramienta permite la creación de casos de prueba en PHP haciendo uso de *mocks* y *stubs*, objetos que reemplazan componentes de la aplicación a fin de aislar el módulo que se quiere probar.

5.6 RECAPITULACIÓN

En este capítulo se describió la arquitectura específica del prototipo desarrollado, para luego detallar la funcionalidad de cada componente, el modelo de datos y las tecnologías utilizadas. En el próximo capítulo se tratará el caso de estudio en el cual se aplicó este prototipo.

²⁸ <https://symfony.com/>

CAPÍTULO 6. CASO DE ESTUDIO

En este capítulo se describe el caso de estudio elegido para instanciar el prototipo: la entrega de RAEE (Residuos de Aparatos Eléctricos y Electrónicos) para el Programa e-basura. En primer lugar, se muestra la solución preexistente, un formulario web, para luego detallar la instanciación con la plataforma DEHIA. Finalmente se abordan las pruebas de usuario realizadas sobre la aplicación móvil a fin de comparar las dos alternativas.

6.1 PROBLEMÁTICA A RESOLVER

Como caso de uso para la arquitectura y componentes propuestos se tomó una necesidad existente en el Programa E-Basura de la UNLP, el cual aborda la problemática de los Residuos de Aparatos Eléctricos y Electrónicos (RAEE) transformándolos en acciones para la inclusión digital, la equidad social y la protección ambiental. Una de las áreas de trabajo del Programa e-basura consiste en la admisión de donaciones de elementos electrónicos en desuso para su posterior reacondicionamiento o descarte seguro²⁹. Para coordinar la donación, es necesario un mecanismo mediante el cual los interesados, ya sean particulares o instituciones, puedan informar el material a entregar, tipos y cantidades, así como un medio de contacto para fijar una fecha y hora de entrega.

6.2 SOLUCIÓN EXISTENTE

Actualmente se cuenta con un formulario web³⁰ basado en LimeSurvey³¹ para la recepción de elementos electrónicos en desuso.

Respecto de la creación del formulario, se realiza mediante la interfaz web de LimeSurvey, que es una herramienta genérica de creación de encuestas. Permite crear preguntas de distintos tipos (ingresar texto, opción múltiple, ingresar una fecha, etc.) y poner condiciones para mostrar u ocultar preguntas de acuerdo a las respuestas anteriores.

El formulario publicado consiste en una serie de campos en una sola página cuyo encabezado puede verse en la Figura 6.2.1.

El formulario se divide en tres secciones: tipo de donación, datos de contacto e información sobre el equipamiento a entregar. De acuerdo al tipo de donación se muestran distintos campos en Datos de Contacto: los particulares deben ingresar nombre, teléfono y correo electrónico mientras que de ser una institución debe ingresarse información acerca de la misma.

²⁹ https://e-basura.unlp.edu.ar/preguntas_frecuentes

³⁰ <https://encuestas.linti.unlp.edu.ar/index.php/369463/>

³¹ <https://www.limesurvey.org/>

Formulario de entrega de elementos

Donación de Equipamiento en Desuso

* Tipo de Donación

📌 Seleccione una de las siguientes opciones

Personal

Institucional

Figura 6.2.1 Formulario original

6.3 INSTANCIACIÓN DEL PROTOTIPO

Se propone una nueva versión del formulario en forma de una actividad que aproveche las capacidades que ofrecen los dispositivos móviles, así como la funcionalidad de salto condicional de la aplicación móvil de la plataforma DEHIA.

En base al formulario preexistente, se diseñó una actividad con 22 tareas, algunas de ellas opcionales (en gris), con conexiones directas (línea llena) y condicionales (línea de puntos). El diagrama de la actividad puede verse en la Figura 6.3.1.

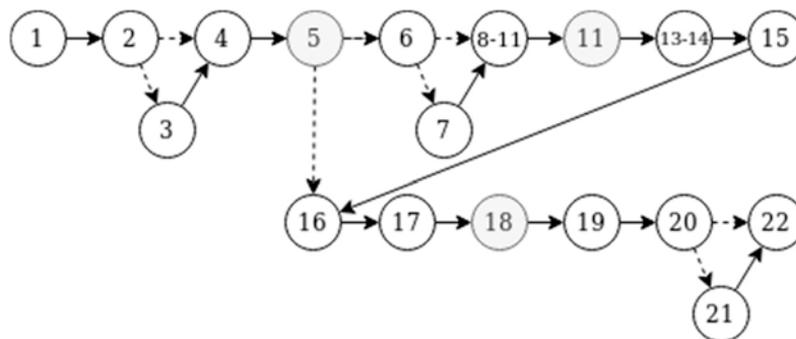
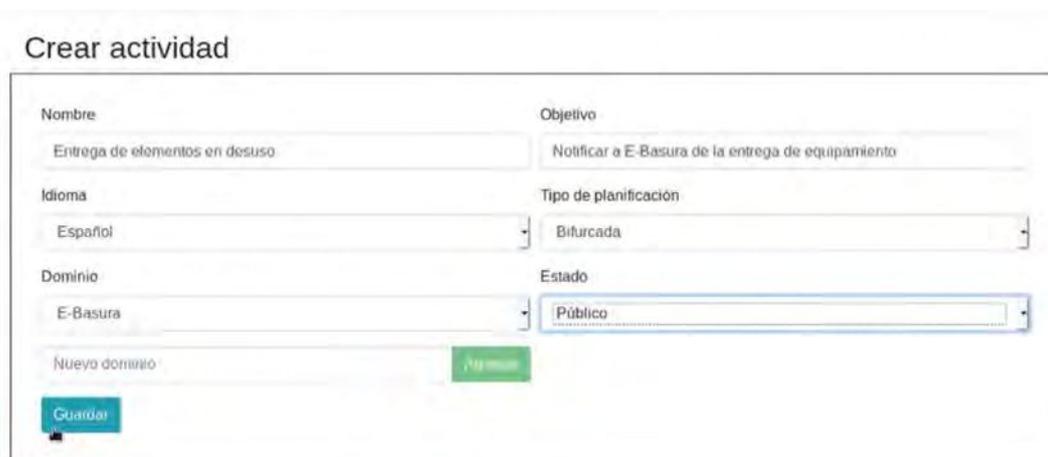


Figura 6.3.1 Diseño de la actividad

Además de las preguntas ya existentes se añadieron tareas de captura de imagen, GPS y audio, a fin de enriquecer la información obtenida por la aplicación. En el Anexo D puede verse el listado completo de tareas con sus respectivos tipos y opciones.

Creación de la actividad

Para crear la actividad se ingresa a la herramienta web de la plataforma y se elige la opción correspondiente desde el menú. Se presentan los campos de la actividad a completar, como se muestra en la Figura 6.3.2. Se elige una planificación Bifurcada a fin de habilitar los saltos condicionales entre tareas.

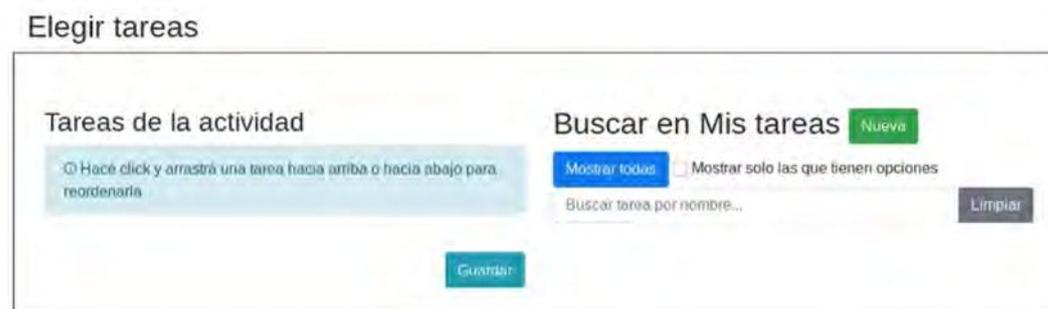


Crear actividad

Nombre	Objetivo
Entrega de elementos en desuso	Notificar a E-Basura de la entrega de equipamiento
Idioma	Tipo de planificación
Español	Bifurcada
Dominio	Estado
E-Basura	Público
Nuevo dominio	<input type="button" value="Nueva"/>
<input type="button" value="Guardar"/>	

Figura 6.3.2 Creación de la actividad

Una vez creada la actividad, se procede a la selección o creación de tareas. En este caso, son todas tareas nuevas y se pueden crear mediante el botón de “Nueva”. La pantalla de asignación de tareas puede verse en su estado inicial en la Figura 6.3.3:



Elegir tareas

Tareas de la actividad	Buscar en Mis tareas <input type="button" value="Nueva"/>
<input type="radio"/> Hacer click y arrastrá una tarea hacia arriba o hacia abajo para reordenarla	<input type="button" value="Mostrar todas"/> <input type="checkbox"/> Mostrar solo las que tienen opciones
	Buscar tarea por nombre... <input type="button" value="Limpiar"/>
<input type="button" value="Guardar"/>	

Figura 6.3.3 Pantalla de asignación de tareas

En la pantalla de creación de tareas, se deben ingresar los datos correspondientes a la tarea que se quiere crear. En el caso de ser una tarea de texto, no es necesario ingresar información adicional. La Figura 6.3.4 muestra esta pantalla.

Crear tarea

Nombre	Consigna
Persona de contacto	Ingresar el nombre de la persona de contacto
Tipo	Estado
Ingresar texto	Público
Dominio	
E-Basura	
Nuevo dominio	<input type="button" value="Agregar"/>
<input type="button" value="Guardar"/>	

Figura 6.3.4 Creación de tarea tipo texto

En el caso de crear una tarea de opción múltiple como la de la Figura 6.3.5, deben agregarse dichas opciones. En este caso no tiene sentido, pero podrían indicarse opciones “correctas”.

Crear tarea

Nombre	Consigna
Equipamiento I	Seleccione los tipos de equipamiento a entregar
Tipo	Estado
Opción múltiple	Público
Dominio	
E-Basura	
Nuevo dominio	<input type="button" value="Agregar"/>
Opción múltiple	
CPUs	<input type="button" value="X Quitar"/>
Monitores CRT (Tubo)	<input type="button" value="X Quitar"/>

Figura 6.3.5 Creación de tarea de opción múltiple

Una vez creadas todas las tareas aparecen listadas en la misma pantalla con su nombre y su tipo. De ser necesario pueden reordenarse. En la Figura 6.3.6 se ve el listado con todas las tareas ya creadas.

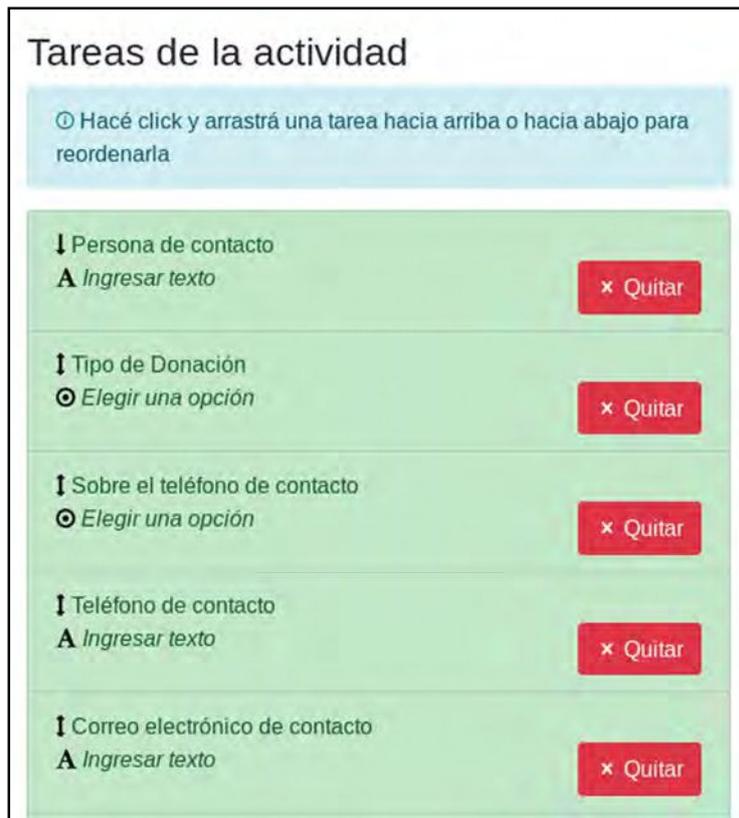


Figura 6.3.6 Listado de tareas de la actividad

Quando están todas las tareas creadas se pasa a la pantalla de planificación. En esta aparecen todas las tareas en forma de círculos. Para conectar dos tareas, se hace click en una de ellas y se especifica la tarea siguiente desde el menú. De ser necesario se elige una condición, como se muestra en la Figura 6.3.7.



Figura 6.3.7 Creación de conexión condicional

Una vez creadas todas las conexiones queda una configuración como la que se puede ver en la Figura 6.3.8.

El último paso consiste en poner la actividad como “Definitiva”, para indicar a la plataforma que puede comenzar a recibir respuestas.

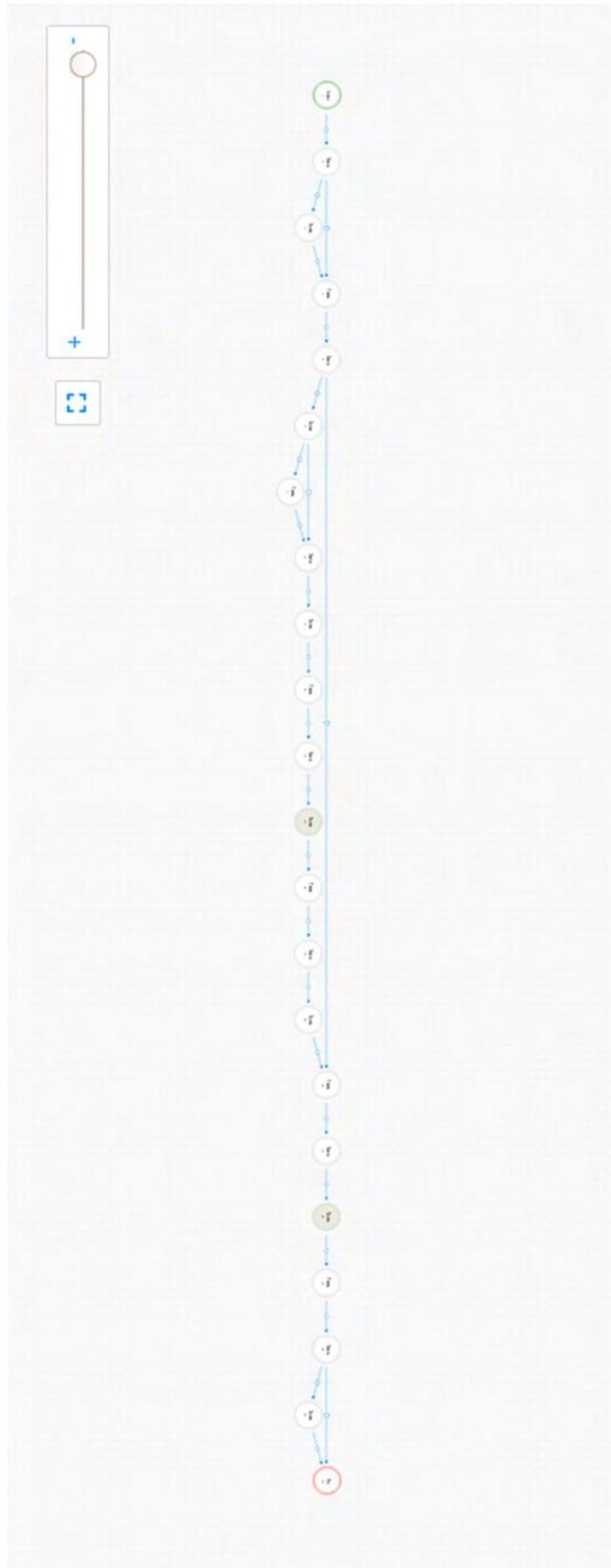


Figura 6.3.8 Planificación terminada

Resolución de la actividad

Una vez que está habilitada la recepción de respuestas, los interesados pueden cargar sus donaciones a través de la aplicación móvil. La definición de la actividad se descarga en formato JSON desde la herramienta web y se ubica en el teléfono en la carpeta DEHIA, desde donde la aplicación escanea las configuraciones disponibles.

Al abrir la aplicación, lo primero que aparece es la lista de actividades encontradas. Se elige la actividad “Entrega de elementos en desuso” para comenzar a resolverla.

De acuerdo con lo planificado, se muestra en primer lugar una tarea para elegir si la donación es particular o institucional, como se muestra en la Figura 6.3.9.

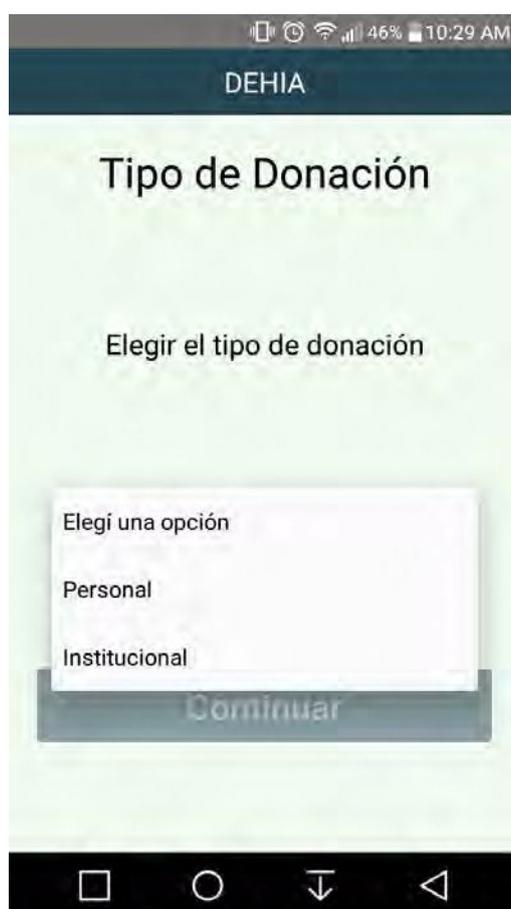


Figura 6.3.9 Pantalla para elegir el tipo de donación

De acuerdo al tipo de donación elegido, se cargan solo datos de contacto o información detallada sobre la institución origen. Entre las nuevas tareas están las de grabar en audio la actividad desarrollada por la institución (Figura 6.3.10), la de tomar una foto de los elementos a entregar (Figura 6.3.11) y la de elegir una ubicación para retirar la donación (Figura 6.3.12).



Figura 6.3.10 Tarea de grabar audio

Figura 6.3.11 Tarea de sacar una foto

Figura 6.3.12 Tarea de localización

Finalmente, cuando se resolvieron todas las tareas presentadas, se puede enviar la información ingresada.

6.4 PRUEBAS DE USUARIO

Dado que el foco de la plataforma web está puesto en que usuarios sin conocimientos de programación puedan acceder a este tipo de herramientas, es importante determinar su nivel de usabilidad. Para tal fin se preparó una prueba en la cual se pide a los participantes creen un conjunto de tareas determinado y luego las secuencien. Se tomó una muestra no probabilística (es decir, se hizo un muestreo por conveniencia) de 5 usuarios. De acuerdo a [Nielsen, 2000]³² son suficientes para una prueba de usabilidad. Se tomaron las pruebas por separado para evitar la interferencia entre los participantes. Se les entregó una guía con los pasos a seguir para crear la actividad de prueba en la plataforma, por un lado, y por otro, una encuesta que incluye un SUS (Simple Usability Scale), “un cuestionario simple de diez preguntas que provee una visión global de aspectos subjetivos de la usabilidad” [Brooke, 1996].

Respecto del caso de uso original, se resumió a seis tareas para que los participantes pudieran realizar la prueba en un tiempo razonable (no más de cuarenta minutos por participante). Estas tareas son representativas de los distintos tipos de tareas. En cuanto a la planificación, se incluyeron tanto conexiones simples como condicionales. La guía completa y el cuestionario pueden verse en el Anexo E.

³² <https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/>

Durante el transcurso de la prueba no se observaron mayores dificultades y el ritmo estuvo dado más por la velocidad de tipeo que por buscar botones en la interfaz. Cada participante tomó un promedio de 16 minutos para realizar la prueba. Si separamos a quienes tenían experiencia en configuración de software de quienes no, se observa que los primeros tardaron en realidad un promedio de 13 minutos contra 19 de los segundos. Se infiere que una mayor experiencia o familiaridad con sistemas de software podría facilitar la resolución del ejercicio.

Respecto al SUS realizado con los usuarios, se obtuvo un puntaje general de 70. Lo que implica un nivel aceptable de usabilidad (a partir de los 68 lo es). En particular, tres de los participantes arrojaron puntajes mayores a los 68 y por debajo en los otros dos (52,5). Si comparamos los usuarios experimentados con los no experimentados, se observa que la mayor tasa de aceptación la tuvieron los no experimentados, que son los usuarios en los cuales se piensa a la hora de diseñar la aplicación.

Descomponiendo brevemente por preguntas, el 60% de los usuarios afirmó que la aplicación era fácil de usar, pero la misma cantidad afirmó no saber si usaría normalmente la aplicación (esto puede deberse a que el objetivo de la aplicación no es de su interés).

Un dato interesante es que los usuarios experimentados creen que es necesaria ayuda de una persona con conocimientos técnicos para utilizar el sistema mientras que los usuarios menos experimentados opinaron lo contrario.

El 80% de los usuarios dijeron sentirse confiados utilizando la aplicación.

En el anexo F pueden verse los resultados de la prueba, los porcentajes y los tiempos.

6.5 RECAPITULACIÓN

En el capítulo se presentó el caso de estudio mediante una actividad diseñada para facilitar la recolección de RAEE (Residuos de Aparatos Eléctricos y Electrónicos). Se llevaron a cabo pruebas de usuario para verificar los niveles de usabilidad del sistema. En el último capítulo, que sigue a continuación, se presentan las conclusiones a las que se llegó con el desarrollo de esta plataforma, así como aspectos que quedaron pendientes y que quedan planteados como trabajos futuros.

CAPÍTULO 7. CONCLUSIONES Y TRABAJOS FUTUROS

En este capítulo, el último de este informe, se presentan las conclusiones y los trabajos futuros que pueden derivar de o mejorar el prototipo y la arquitectura.

7.1 CONCLUSIONES

Se estudió el estado del arte en lo que respecta a plataformas de recolección de datos que permiten la creación de actividades en dominios múltiples por parte de usuarios sin conocimientos de programación. Estas soluciones, al igual que la plataforma propuesta, presentan una herramienta de diseño web, una aplicación móvil para resolver las actividades y un servidor que recibe las respuestas y provee los resultados. Todas comparten un flujo de trabajo que consiste en [Diseñar →] Definir, → Ejecutar → Visualizar. Se analizaron las funcionalidades y características específicas de cada herramienta de definición, aplicación móvil y servidor. Se observó que, si bien cada plataforma tiene sus particularidades, hay una serie de características básicas en común a todas ellas, incluyendo la plataforma propuesta, como tipos de tareas soportadas, facilidades del editor de actividades, usabilidad de la aplicación móvil, etcétera. Cabe destacar que ninguna de las plataformas analizadas presenta la posibilidad de reuso de tareas y actividades (aunque durante la redacción de este informe Epicollect5 agregó una funcionalidad para importar actividades completas) y solo MoLE y la plataforma propuesta permiten la visualización y edición gráfica de la estructura del workflow subyacente a la planificación de las tareas.

Se definió la arquitectura de la plataforma propuesta como un sistema distribuido basado en microservicios. Se analizó la estructura mediante un esquema de capas, partiendo de los clientes hasta llegar a los métodos de persistencia. Se presentó una variedad de componentes posibles de cada capa, de los cuales sólo un subconjunto llegó a formar parte del prototipo. Además, se detalló la funcionalidad esperada para cada componente de alto nivel: Editor Web, Resolución Web, Aplicación Móvil y API (donde esta última abstrae todos los microservicios y compuertas).

Se desarrolló una solución prototípica que implementa una pequeña parte de la plataforma definida. Se detallaron los componentes específicos (clientes, servicios, compuerta, mecanismos de persistencia) que fueron elegidos para su implementación por formar parte del núcleo de la funcionalidad y que permitieran obtener una versión reducida pero completa en cuanto a sus prestaciones funcionales. Se ilustró la explicación de las características implementadas mediante un conjunto de capturas de pantalla del sistema en funcionamiento, que cumple con el flujo de trabajo ([Diseñar →] Definir, → Ejecutar → Visualizar) propio de este tipo de soluciones.

Por último, se aplicó el prototipo al dominio de los RAEE (Residuos de Aparatos Eléctricos y Electrónicos) a través de una actividad para concretar la donación de elementos en desuso para el Programa E-Basura.

Se espera que la plataforma facilite el acceso a la recolección de datos a usuarios sin conocimientos de programación, favoreciendo el reuso de esfuerzo y conocimiento de otros usuarios.

7.3 CONTRIBUCIONES

Durante el transcurso del desarrollo de la tesina se presentaron, y se expusieron a evaluación tres artículos relacionados al tema de la tesina de grado. Los cuales han sido aceptados y luego expuestos por quien presenta esta tesina.:

1. Arcidiacono, J., Lliteras, A. B., & Bazán, P. A. (2020). **Plataforma para la definición y ejecución de actividades orientadas a la recolección y análisis de datos, con intervención humana.** In *XXII Workshop de Investigadores en Ciencias de la Computación (WICC 2020, El Calafate, Santa Cruz)*.

Abstract:

“En este trabajo se propone una plataforma para que usuarios finales definan y ejecuten actividades orientadas a la recolección y análisis de datos. Dichas actividades serán implementadas mediante el concepto de workflow. La plataforma se basará en una arquitectura distribuida basada en microservicios.”

Relación con la tesina:

En este artículo se realiza una presentación de la propuesta de plataforma describiendo el concepto de actividades y la arquitectura diseñada para esta tesina.

2. Arcidiacono Jose, Bazán Patricia y Lliteras Alejandra B. **Arquitectura de microservicios distribuidos para una plataforma que orquesta actividades orientadas a la recolección y análisis de datos, con intervención humana** en JAIIO 2020, Simposio EST (concurso de trabajos de grado).

Abstract:

“Este trabajo presenta la arquitectura distribuida y basada en microservicios, definida para dar soporte a una plataforma para generar y reusar workflows de actividades de recolección de datos, que involucren la intervención humana, y cuya ejecución se realiza desde una aplicación móvil. Tanto la arquitectura como la plataforma conforman la solución propuesta como Tesina de Grado realizada y dirigida por las autoras de este trabajo, siendo el foco principal únicamente la arquitectura sin abordar detalles de la plataforma. Se presentan también las características principales de la recolección de datos con intervención humana”

Relación con la tesina:

Es un resumen en cuanto al estado del arte (Capítulo 3) de las plataformas de creación de actividades de recolección de datos con intervención humana y sus arquitecturas. Comparación y presentación de la arquitectura propuesta para esta tesina (Capítulo 4).

3. Arcidiacono Jose, Lliteras Alejandra B. y Bazán Patricia. **DEHIA, una plataforma para la generación y ejecución de actividades de recolección de datos con intervención humana aplicada en el Programa E-Basura** en JAIIO 2020, Simposio AGRANDA (GRANdes DATos):

Abstract:

“En este trabajo se presenta el desarrollo, en progreso, de DEHIA una plataforma para crear, clonar y gestionar workflows de actividades de recolección de datos, que involucren la intervención humana, y cuya ejecución se realiza desde una aplicación móvil. Para ello, se presenta una arquitectura de solución conformada, por tres componentes: una plataforma Web que permite a usuarios finales crear, clonar y gestionar sus propias actividades de recolección de datos en diferentes dominios, una aplicación móvil para que voluntarios participen de la actividad de recolección de datos y una API responsable de la comunicación entre las dos primeras componentes. Se presenta, además, un caso de uso proponiendo una alternativa tecnológica inclusiva y en este caso particular, con la expectativa de aportar a la labor social que el programa E-Basura realiza desde el año 2009. La propuesta de este trabajo espera abrir un espacio de debate en miras de la democratización y apertura de este tipo de actividad para ser empleado en diferentes disciplinas.”

Relación con la tesina:

Presentación del caso de uso en el programa E-Basura (Capítulo 6) en el cual se aplicó la plataforma DEHIA detallando la interfaz tanto para crear actividades como para ejecutarlas.

Los artículos completos pueden encontrarse junto a la entrega de los archivos del código fuente (véase Anexo G)

7.2 TRABAJOS FUTUROS

En esta sección se plantean distintos aspectos que pueden ser extendidos o implementados en diferentes áreas abarcadas en el informe.

Funcionalidades

Se necesita un buscador de tareas y actividades por autor, así como algún tipo de listado de los autores o tareas más populares para facilitar el compartir conocimiento. En cuanto al editor, necesita una previsualización de las tareas tal como se verían en el dispositivo, para ayudar al autor. Sería deseable poder importar y exportar tareas individualmente, así como reimportar definiciones de actividades ya creadas. También permitir la edición y borrado de tareas. La planificación podría incluir plantillas prediseñadas como por ejemplo “grafo lineal”. La edición del grafo podría ser colaborativa en tiempo real.

Faltaría implementar el componente de ejecución web, que permitiría resolver las actividades de forma individual o colaborativa, mostrando quién está en línea y algunas estadísticas. Podrían agregarse tareas nuevas de análisis de datos pensados para este nuevo componente. Ambos componentes web necesitan internacionalización.

En cuanto a la aplicación móvil, podría preservar el estado de una actividad entre reinicios de la misma, así como permitir varias resoluciones consecutivas antes de enviar los datos

Colaboración / Sincronismo

A nivel arquitectura, podrían reorganizarse los microservicios alrededor de entidades (“actividad”, “planificación”, etc.). La comunicación entre servicios podría mejorarse incorporando mecanismos asíncronos (como colas de eventos). Hace falta un *script* de inicio de los servicios para no tener que levantar la plataforma de forma manual. Para dar soporte a las nuevas funcionalidades, también habría que considerar la incorporación de websockets. Por último, hay que considerar cuestiones de escalabilidad ante una gran cantidad de usuarios o de respuestas, y de resiliencia (resistencia a fallos) como reintentos, disyuntores, etc.

Seguridad

En cuanto a la seguridad de la plataforma, aún no están implementados los permisos para las actividades privadas (sería deseable requerir un código para su ejecución o descarga). Lo mismo ocurre con las imágenes de este tipo de actividades. Para proteger el tráfico de la aplicación también es necesario el uso de HTTPS. Respecto de la autenticación y autorización tanto de usuarios como de microservicios, podrían incorporarse otros proveedores de identidad como Facebook; se podría implementar la creación de credenciales propias de OAuth (existe, pero no tiene interfaz) y por último API keys para que los servicios se identifiquen entre ellos.

Pruebas con usuarios

Sumar pruebas con usuarios tanto de la plataforma web, como de la ejecución de las actividades desde la aplicación móvil. Lo que brindaría feedback para la mejora y adecuación a las necesidades de los usuarios destinatarios.

REFERENCIAS BIBLIOGRÁFICAS

- [Babbie, 2000] Babbie, E., & Martínez, J. F. J. D. (2000). Fundamentos de la investigación social (No. 300.72 B3Y.). México: Thomson
- [Balsalobre, 2018] Balsalobre, A., Ceccarelli, S., Cano, M. E., Ferrari, W. A., Cochero, J., & Martí, G. A. (2018). GeoVin: Ciencia Ciudadana para aprender de las vinchucas de Argentina. In II Congreso Argentino de Ciencia Abierta y Ciudadana (CIACIAR)(Universidad Nacional de San Martín, 2 de noviembre de 2018).
- [Bass, et al., 2003] BASS, Len; CLEMENTS, Paul; KAZMAN, Rick. Software architecture in practice. Addison-Wesley Professional, 2003.
- [Bazán, et al., 2017] Bazán P. et al. (2017). Arquitecturas, Servicios y Procesos Distribuidos. Una Visión desde la construcción del software. Libro de Cátedra. Editado por EDULP (Editorial de la UNLP). ISBN 978-950-34-1520-7
http://sedici.unlp.edu.ar/bitstream/handle/10915/62354/Documento_completo.pdf-PDFA.pdf?sequence=1
- [Berners-Lee, et al., 1996] BERNERS-LEE, Tim; FIELDING, Roy; FRYSTYK, H. RFC 1945: Hypertext Transfer Protocol—HTTP/1.0, May 1996. Status: INFORMATIONAL, 1997, vol. 61.
- [Bhattacharjee, 2005] Bhattacharjee Y. 2005. Citizen scientists supplement work of Cornell researchers. *Science* 308: 1402–1403.
- [Bonney et al., 2009] Bonney, R., Cooper, C. B., Dickinson, J., Kelling, S., Phillips, T., Rosenberg, K. V., & Shirk, J. (2009). Citizen science: a developing tool for expanding science knowledge and scientific literacy. *BioScience*,59(11), 977-984.
- [Bonney et al., 2009] Bonney, R., Cooper, C. B., Dickinson, J., Kelling, S., Phillips, T., Rosenberg, K. V., & Shirk, J. (2009). Citizen science: a developing tool for expanding science knowledge and scientific literacy. *BioScience*,59(11), 977-984.
- [Brooke, 1996] Brooke, J. (1996). SUS: a “quick and dirty usability scale”. Usability evaluation in industry, 189.
- [Burke, et al., 2006] Burke, J. A, Estrin, D., Hansen, M., Parker, A., Ramanathan, N., Reddy, S., & Srivastava, M. B. (2006). Participatory sensing. UCLA: Center for Embedded Network Sensing.
- [Cochero, 2018] Cochero, J. (2018). AppEAR: Una aplicación móvil de ciencia ciudadana para mapear la calidad de los hábitats acuáticos continentales.
- [Cochrane and Bateman, 2010] Cochrane, T., & Bateman, R. Smartphones give you wings: Pedagogical affordances of mobile Web 2.0. *Australasian Journal of Educational Technology*, 26(1). 2010
- [Colouris, et al., 2000] Colouris, G et al. (2000) Distributed Systems Concepts and Design 3e. Addison-Wesley. ISBN- 13: 978-0132143011
- [Dal Bianco et al., 2019] Dal Bianco, P. A., Mozzon Corporaal, F., Lliteras, A. B., Grigera, J., & Gordillo, S. E. (2019). MoLE: A web authoring tool for building mobile learning experiences.

In XXV Congreso Argentino de Ciencias de la Computación (CACIC 2019, Universidad Nacional de Río Cuarto).

[Engeström, 2001] Engeström, Y. Expansive learning at work: Toward an activity theoretical reconceptualization. *Journal of education and work*, 14(1), 133-156. 2001

[Fowler, et al., 2014] Fowler, Martin, and James Lewis. "Microservices a definition of this new architectural term." URL: <http://martinfowler.com/articles/microservices.html> (2014)

[Gourley, et al., 2002] GOURLEY, David, et al. HTTP: the definitive guide. " O'Reilly Media, Inc.", 2002.

[Hartung et al., 2010] HARTUNG, Carl, et al. Open data kit: tools to build information services for developing regions. En *Proceedings of the 4th ACM/IEEE international conference on information and communication technologies and development*. 2010. p. 1-12.

[Kim, et al., 2013] Kim, S., Mankoff, J., & Paulos, E. (2013, February). Sensr: evaluating a flexible framework for authoring mobile data-collection tools for citizen science. In *Proceedings of the 2013 conference on Computer supported cooperative work* (pp. 1453-1462).

[Kunze et al., 2016] Kunze, Matthias, and Mathias Weske. *Behavioural models: From modelling finite automata to analysing business processes*. Springer, 2016.

[LINTI, 2018] Proyecto Recicla tu Compu-Recicla tu mundo. Proyecto de extensión de la UNLP, Facultad de Informática, LINTI.
http://www.extension.info.unlp.edu.ar/articulo/2018/12/18/la_unlp_acredito_cinco_proyectos_de_extension_presentados_por_el_linti_2019

[Llitas et al., 2017] Llitas, A. B., Challiol, C., & Gordillo, S. E. (2017). Location-based mobile learning applications: a conceptual framework for co-design. In *2017 Twelfth Latin American Conference on Learning Technologies (LACLO)* (pp. 1-8). IEEE.

[Llitas et al., 2018] Llitas, A. B., Grigera, J., dal Bianco, P. A., Corporaal, F. M., & Gordillo, S. E. (2018). Challenges in the design of a customized location-based mobile learning application. In *2018 XIII Latin American Conference on Learning Technologies (LACLO)* (pp. 315-321). IEEE.

[Llitas et al., 2019] Llitas, A. B., Grigera, J., Corporaal, F. R. M., Di Claudio, F., & Gordillo, S. E. (2019, October). A Flexible Web Authoring Tool for Building Mobile Learning Experiences. In *Argentine Congress of Computer Science* (pp. 69-83). Springer, Cham.

[Llitas et al., 2012] Llitas, A.B., Challiol, C. y Gordillo, S. *Juegos Educativos Móviles Basados en Posicionamiento: Una Guía para su Conceptualización*. 41 JAIIO. Agosto de 2012. Facultad de Informática, UNLP. Con referato. In *Proceedings of ASSE 2012 Argentine Symposium on Software Engineering*. ISSN: 1850-2792, pp. 164-175. 2012

[Llitas, 2015] Llitas, A. B. (2015). *Un enfoque de modelado de actividades educativas posicionadas que contemplan elementos concretos*. Tesis de maestría. Facultad de Informática, UNLP.

- [Morris et al.,2018] Morris, C. A., Deochand, N., & Peterson, S. M. (2018). Using Microsoft Excel® to build a customized partial-interval data collection system. *Behavior analysis in practice*, 11(4), 504-516.
- [Newman, 2015] Newman, Sam. *Building microservices: designing fine-grained systems*. " O'Reilly Media, Inc.", 2015.
- [Nielsen, 2000] Nielsen, J. (2000). *Why you only need to test with 5 users*.
- [O'Malley et al., 2003] O'Malley, C., Vavoula, G., Glew, J., Taylor, J., Sharples, M., & Lefrere, P. *Guidelines for learning/teaching/tutoring/ in a mobile environment*. Mobilelearn project deliverable. 2003
- [Richardson, et al., 2013] RICHARDSON, Leonard, et al. *RESTful Web APIs: Services for a Changing World*. " O'Reilly Media, Inc.", 2013.
- [Simpson et al., 2013] SIMPSON, Robert; PAGE, Kevin R.; DE ROURE, David. *Zooniverse: observing the world's largest citizen science platform*. En *Proceedings of the 23rd international conference on world wide web*. 2014. p. 1049-1054.
- [Sprinks et al., 2017] Sprinks, J., Wardlaw, J., Houghton, R., Bamford, S., & Morley, J. (2017). *Task Workflow Design and its impact on performance and volunteers' subjective preference in Virtual Citizen Science*. *International Journal of Human-Computer Studies*, 104, 50-63.
- [Steinberg et al., 2019] Steinberg, M., Schindler, S., & Klan, F. (2019). *Software solutions for form-based, mobile data collection—A comparative evaluation*. *BTW 2019—Workshopband*.
- [Strasser et al., 2019] Strasser, B. J., Baudry, J., Mahr, D., Sanchez, G., & Tancoigne, E. (2019). *"Citizen Science"? Rethinking Science and Public Participation*. *Science & Technology Studies*, 52-76.
- [Tanenbaum, et al., 2007] Tanenbaum, A. S., & Van Steen, M. (2007). *Distributed systems: principles and paradigms*. Prentice-Hall.
- [Traxler, 2009] Traxler, J. (2009). *Learning in a mobile age*. *International Journal of Mobile and Blended Learning (IJMBL)*, 1(1), 1-12.
- [Trouille, et al., 2019] TROUILLE, Laura; LINTOTT, Chris J.; FORTSON, Lucy F. *Citizen science frontiers: Efficiency, engagement, and serendipitous discovery with human-machine systems*. *Proceedings of the National Academy of Sciences*, 2019, vol. 116, no 6, p. 1902-1909.
- [van Der Aalst, 2003] van Der Aalst, Wil MP, et al. *"Workflow patterns"*. *Distributed and parallel databases* 14.1. Pág 5-51. (2003)
- [van Steen, et al., 2016] van Steen, M., & Tanenbaum, A. S. (2016). *A brief introduction to distributed systems*. *Computing*, 98(10), 967-1009.
- [Webber, et al., 2010] WEBBER, Jim; PARASTATIDIS, Savas; ROBINSON, Ian. *REST in practice: Hypermedia and systems architecture*. " O'Reilly Media, Inc.", 2010.

ANEXOS

Un compendio de información que excede el alcance de los capítulos particulares.

ANEXO A - PROTOCOLO DE LA API DE DEHIA

En este anexo se presenta la API implementada por la plataforma DEHIA. Para cada funcionalidad se detalla el *endpoint* (método + ruta), si requiere autenticación, los parámetros necesarios, la respuesta devuelta por el servidor y los posibles casos de error.

Listar todas las actividades del usuario

Endpoint	GET /api/v1.0/actividades/user
Requiere token	Sí
Parámetros	-
Respuesta	200 OK <pre>{ "actividades": [<lista de actividades>] }</pre>
Errores	401: falta el token

Crear una actividad

Endpoint	POST /api/v1.0/actividades
Requiere token	Sí
Parámetros	<ul style="list-style-type: none">- nombre- objetivo- codigo- dominio- idioma- tipoPlanificacion- estado

Respuesta	<pre> 201 Created { "nombre": ..., "objetivo": ..., "codigo": ..., "dominio": ..., "idioma": ..., ... } </pre>
Errores	<pre> 401: falta el token 400: el código ya existe </pre>

Editar una actividad (sólo ciertos atributos)

Endpoint	PATCH /api/v1.0/actividades/{id}
Requiere token	Sí
Parámetros	<ul style="list-style-type: none"> - definitiva - cerrada - estado
Respuesta	<pre> 200 OK { "nombre": ..., "objetivo": ..., "codigo": ..., "dominio": ..., "idioma": ..., ... } </pre>

Errores	<p>400: campo inválido o estado inválido</p> <p>401: falta el token</p> <p>403: la actividad no es propia</p> <p>404: actividad no encontrada</p>
---------	---

Obtener una actividad del usuario

Endpoint	GET /api/v1.0/actividades/{id}
Requiere token	Sí
Parámetros	-
Respuesta	<p>200 OK</p> <pre>{ "nombre": ..., "objetivo": ..., "codigo": ..., "dominio": ..., "idioma": ..., ... }</pre>
Errores	<p>401: falta el token</p> <p>403: la actividad es privada y no es propia</p> <p>404: actividad no encontrada</p>

Listar las tareas de una actividad

Endpoint	GET /api/v1.0/actividades/{id}/tareas
Requiere token	Sí

Parámetros	-
Respuesta	200 OK <pre>{ "tareas" : [<lista de tareas>] }</pre>
Errores	401: falta el token 403: la actividad es privada y no es propia 404: actividad no encontrada

Asignar un conjunto de tareas a una actividad

Endpoint	PUT /api/v1.0/actividades/{id}/tareas
Requiere token	Sí
Parámetros	- tareas
Respuesta	200 OK <pre>{ status: "ok" }</pre>
Errores	400: petición errónea 401: falta el token 403: la actividad no es propia, o las tareas son privadas y ajenas

Listar todas las actividades públicas

Endpoint	GET /api/v1.0/public/actividades
Requiere token	No

Parámetros	-
Respuesta	<pre> 200 OK { "results": [<primeras 10 actividades>], "page": 1, "total": 50 } </pre>
Errores	-

Obtener una actividad pública

Endpoint	GET /api/v1.0/public/actividades/{id}
Requiere token	No
Parámetros	-
Respuesta	<pre> 200 OK { "nombre": ..., "objetivo": ..., "codigo": ..., "dominio": ..., "idioma": ..., ... } </pre>
Errores	<p>403: la actividad es privada</p> <p>404: actividad no encontrada</p>

Listar las tareas de una actividad

Endpoint	GET /api/v1.0/actividades/{id}/tareas
Requiere token	Sí
Parámetros	-
Respuesta	200 OK <pre>{ "tareas": [<ids de las tareas>] }</pre>
Errores	401: falta el token 403: la actividad es privada y no es propia

Descargar la definición de una actividad

Endpoint	GET /api/v1.0/public/actividades/{id}/data
Requiere token	No
Parámetros	-
Respuesta	200 OK <definición en JSON>
Errores	404: actividad no encontrada

Listar todas las tareas de un usuario

Endpoint	GET /api/v1.0/actividades/user
Requiere token	Sí

Parámetros	-
Respuesta	200 OK <pre>{ "results": [<primeras 10 tareas>], "page": 1, "total": 32 }</pre>
Errores	401: falta el token

Crear una tarea

Endpoint	POST /api/v1.0/tareas
Requiere token	Sí
Parámetros	<ul style="list-style-type: none"> - nombre - consigna - codigo - tipo - dominio - estado - extraData
Respuesta	201 Created <pre>{ "nombre": ..., "consigna": ..., "codigo": ..., ... }</pre>
Errores	400: el código ya existe 401: falta el token

Obtener una tarea

Endpoint	GET /api/v1.0/tareas/{id}
Requiere token	Sí
Parámetros	-
Respuesta	200 OK <pre>{ "nombre": ..., "consigna": ..., "codigo": ..., ... }</pre>
Errores	401: falta el token 403: la tarea es privada y no es propia 404: tarea no encontrada

Agregar el plano a una tarea posicionada

Endpoint	POST /api/v1.0/tareas/{id}/plano
Requiere token	Sí
Parámetros	- plano (formData - archivo)
Respuesta	200 OK <pre>{ status: "ok" }</pre>

Errores	400: error en el archivo 401: falta el token 403: la tarea no es propia 404: tarea no encontrada
---------	---

Listar todas las tareas públicas

Endpoint	GET /api/v1.0/public/tareas
Requiere token	No
Parámetros	-
Respuesta	200 OK <pre>{ results: [<primeras 10 tareas>], page: 1, total: 32 }</pre>
Errores	-

Obtener una tarea pública

Endpoint	GET /api/v1.0/public/tareas/{id}
Requiere token	No
Parámetros	-

Respuesta	<pre> 200 OK { "nombre": ..., "consigna": ..., "codigo": ..., ... } </pre>
Errores	

Obtener la planificación de una actividad

Endpoint	GET /api/v1.0/planificaciones/{id}
Requiere token	Sí
Parámetros	-
Respuesta	<pre> 200 OK { opcionales_ids: [<ids de tareas>], iniciales_ids: [<ids de tareas>], saltos: [{ origen: <id_tarea>, destino: [<ids_tareas>], respuesta: <id_respuesta>, condicion: <YES/NO/YES_TASK/NO_TASK/ALL/CORRECT> }] } </pre>

Errores	<p>401: falta el token</p> <p>403: la actividad es privada y no es propia</p> <p>404: actividad no encontrada</p>
---------	---

Asignar una planificación a una actividad

Endpoint	PUT /api/v1.0/planificaciones/{id}
Requiere token	Sí
Parámetros	<ul style="list-style-type: none"> - iniciales_ids - opcionales_ids - saltos
Respuesta	<pre> 201 Created { opcionales_ids: [<ids de tareas>], iniciales_ids: [<ids de tareas>], saltos: [{ origen: <id_tarea>, destino: [<ids_tareas>], respuesta: <id_respuesta>, condicion: <YES/NO/YES_TASK/NO_TASK/ALL/CORRECT> }] } </pre>
Errores	<p>400: petición errónea</p> <p>401: falta el token</p> <p>403: la actividad es privada y no es propia</p> <p>404: actividad no encontrada</p>

Mostrar planificación de una actividad pública

Endpoint	GET /api/v1.0/public/planificaciones/{id}
Requiere token	No
Parámetros	-
Respuesta	200 OK <pre>{ opcionales_ids: [<ids de tareas>], iniciales_ids: [<ids de tareas>], saltos: [{ origen: <id_tarea>, destino: [<ids_tareas>], respuesta: <id_respuesta>, condicion: <YES/NO/YES_TASK/NO_TASK/ALL/CORRECT> }] }</pre>
Errores	403: la actividad es privada 404: actividad no encontrada

Crear un dominio

Endpoint	POST /api/v1.0/dominios
Requiere token	Sí
Parámetros	- nombre

Respuesta	201 Created <pre>{ nombre: ... }</pre>
Errores	400: el dominio ya existe, error en la petición 401: falta el token

Listar todos los dominios disponibles

Endpoint	GET /api/v1.0/public/dominios
Requiere token	No
Parámetros	-
Respuesta	200 OK <pre>{ results: [<dominios>] }</pre>
Errores	-

Listar los estados de actividad disponibles

Endpoint	GET /api/v1.0/public/estados
Requiere token	No
Parámetros	-
Respuesta	200 OK <pre>{ results: [<estados>] }</pre>

	}
Errores	-

Listar los tipos de tarea disponibles

Endpoint	GET /api/v1.0/public/tipos-tarea
Requiere token	No
Parámetros	-
Respuesta	200 OK <pre>{ results: [<tipos de tarea>] }</pre>
Errores	-

Listar los tipos de planificación disponibles

Endpoint	GET /api/v1.0/public/tipos-planificacion
Requiere token	No
Parámetros	-
Respuesta	200 OK <pre>{ results: [<tipos de planificación>] }</pre>
Errores	-

Listar los idiomas disponibles

Endpoint	GET /api/v1.0/public/idiomas
Requiere token	No
Parámetros	-
Respuesta	200 OK <pre>{ results: [<idiomas>] }</pre>
Errores	-

Enviar una respuesta

Endpoint	POST /api/v1.0/entries
Requiere token	No
Parámetros	- code - responses
Respuesta	201 Created <pre>{ code: <codigo de actividad>, responses: [<respuestas por tarea>] }</pre>
Errores	400: petición errónea o actividad cerrada 404: la actividad no existe

Obtener las resoluciones para una actividad

Endpoint	GET /api/v1.0/resultados?code={codigo_actividad}
Requiere token	Sí
Parámetros	-
Respuesta	200 OK <pre>{ tareas: <nombres de las tareas>, respuestas: [<respuestas ordenadas por tarea>] }</pre>
Errores	401: falta el token 403: la actividad no es propia 404: la actividad no existe

ANEXO B - BIBLIOTECAS UTILIZADAS PARA EL PROTOTIPO

En este anexo se detallan las bibliotecas utilizadas en cada componente para su desarrollo.

SERVICIOS

FRIENDS-OF-SYMFONY/OAUTH-SERVER-BUNDLE
([HTTPS://GITHUB.COM/FRIENDSOFSYMFONY/FOSOAUTHSERVERBUNDLE](https://github.com/friendsofsymfony/fosoauthserverbundle))

Se usa en la implementación de OAuth2 en el servidor. Hubo que sobrescribir algunas partes para adaptarlas al formato de errores (porque en lugar de lanzar una excepción devolvía una respuesta), y agregarle la posibilidad de identificarse con un id_token de Google (que internamente se intercambia por un client_id+client_secret)

FRIENDS-OF-SYMFONY/REST-BUNDLE
([HTTPS://GITHUB.COM/FRIENDSOFSYMFONY/FOSRESTBUNDLE](https://github.com/friendsofsymfony/fosrestbundle))

Se usa en la creación de una API REST. Es una de las bibliotecas más usadas (en Symfony) y está recomendado por la documentación oficial.

GOOGLE/APICLIENT ([HTTPS://GITHUB.COM/GOOGLEAPIS/GOOGLE-API-PHP-CLIENT](https://github.com/googleapis/google-api-php-client))

Se usa en el login y registro de usuarios, para validar los id_token. Es la biblioteca oficial de Google para interactuar con sus apis (en este caso OpenIDConnect)

JMS/SERIALIZER-BUNDLE ([HTTPS://GITHUB.COM/NEXYLAN/JMS-SERIALIZER/](https://github.com/NEXYLAN/JMS-SERIALIZER/))

Se usa en la serialización y deserialización de objetos en los pedidos y respuestas REST. Funciona bien con el RestBundle.

NELMIO/API-DOC-BUNDLE
([HTTPS://GITHUB.COM/NELMIO/NELMIOAPIDOCBUNDLE](https://github.com/NELMIO/NELMIOAPIDOCBUNDLE))

Se usa para documentar la API. Permite crear la documentación fácilmente con anotaciones y exportarla en formato estándar (Swagger) en JSON

SENSIO/Framework-EXTRA-BUNDLE
([HTTPS://GITHUB.COM/SENSIOLABS/SENSIOFRAMEWORKESTRABUNDLE](https://github.com/SENSIOLABS/SENSIOFRAMEWORKESTRABUNDLE))

Se usa en los controllers, para escribir menos código. Facilita mucho la escritura de controllers y está recomendado por la documentación oficial.

GUZZLEHTTP/GUZZLE ([HTTPS://GITHUB.COM/GUZZLE/GUZZLE](https://github.com/GUZZLE/GUZZLE))

Se usa para conectarse a los otros microservicios vía HTTP. Es un cliente HTTP conocido para PHP.

WHITE-OCTOBER/PAGERFANTA-BUNDLE
([HTTPS://GITHUB.COM/WHITEOCTOBER/WHITEOCTOBERPAGERFANTABUNDLE](https://github.com/WHITEOCTOBER/WHITEOCTOBERPAGERFANTABUNDLE))

Se usa para la paginación de resultados. Una guía recomendada por la página oficial de Symfony lo recomienda.

ONEUP/FLYSYSTEM-BUNDLE ([HTTPS://GITHUB.COM/1UP-LAB/ONEUPFLYSYSTEMBUNDLE](https://github.com/1UP-LAB/ONEUPFLYSYSTEMBUNDLE))

Sirve para abstraer del sistema de archivos. Se usa esta librería para poder cambiar a un almacenamiento en la nube (ej. S3) sin afectar la implementación.

APLICACIÓN MÓVIL

REACT-NATIVE-CAMERA-KIT - COMPONENTE `CAMERAKITCAMERASCREEN`
([HTTPS://GITHUB.COM/WIX/REACT-NATIVE-CAMERA-KIT](https://github.com/WIX/REACT-NATIVE-CAMERA-KIT))

Se usa para capturar códigos QR.

REACT-NATIVE-ELEMENTS - COMPONENTES `BUTTON`, `LISTITEM`, `INPUT`
([HTTPS://GITHUB.COM/REACT-NATIVE-ELEMENTS/REACT-NATIVE-ELEMENTS](https://github.com/REACT-NATIVE-ELEMENTS/REACT-NATIVE-ELEMENTS))

Se usa en botones y otros elementos visuales. Brinda más flexibilidad que los elementos nativos

REACT-NATIVE-FS ([HTTPS://GITHUB.COM/ITINANCE/REACT-NATIVE-FS](https://github.com/ITINANCE/REACT-NATIVE-FS))

Se usa en el acceso al sistema de archivos (para importar JSON)

REACT-NATIVE-IMAGE-PICKER - COMPONENTE IMAGEPICKER
([HTTPS://GITHUB.COM/REACT-NATIVE-COMMUNITY/REACT-NATIVE-IMAGE-PICKER](https://github.com/react-native-community/react-native-image-picker))

Se usa para sacar fotos / elegir imágenes de la galería (pantalla de tareas con foto). Está hecho por ReactNativeCommunity y funcionó para lo que necesitaba.

REACT-NATIVE-VECTOR-ICONS ([HTTPS://GITHUB.COM/OBLADOR/REACT-NATIVE-VECTOR-ICONS](https://github.com/Oblador/react-native-vector-icons))

Se usa en elementos visuales como íconos de check, flechas, etc. Tiene una gran variedad de íconos.

REACT-NAVIGATION ([HTTPS://GITHUB.COM/REACT-NAVIGATION/REACT-NAVIGATION](https://github.com/react-navigation/react-navigation))

Se usa para la navegación entre pantallas. Está recomendada por la documentación oficial. Es sencilla de usar.

REACT-REDUX Y REDUX ([HTTPS://GITHUB.COM/REDUXJS/REACT-REDUX](https://github.com/reduxjs/react-redux) Y [HTTPS://GITHUB.COM/REDUXJS/REDUX](https://github.com/reduxjs/redux))

Se usa para manejar el estado interno de la aplicación. Es una forma ordenada de mantener la información entre pantallas. Es ampliamente utilizado.

APLICACIÓN WEB

BOOTSTRAP Y REACT-BOOTSTRAP - COMPONENTES BUTTON, FORM, CONTAINER, ROW, COL, ETC ([HTTPS://GITHUB.COM/REACT-BOOTSTRAP/REACT-BOOTSTRAP](https://github.com/react-bootstrap/react-bootstrap))

Se usa para dar estilos visuales a la aplicación. Funciona de manera muy similar a la versión HTML, que es conocida y ampliamente utilizada

FORMIK - COMPONENTE FORMIK
([HTTPS://GITHUB.COM/JAREDPALMER/FORMIK](https://github.com/jaredpalmer/formik))

Se usa para generar formularios. Simplifica el manejo del estado de los componentes. Parece ser más sencillo que otras alternativas.

OIDC-CLIENT Y REDUX-OIDC - COMPONENTES CALLBACKCOMPONENT Y MÉTODOS DE LA API ([HTTPS://GITHUB.COM/MAXMANTZ/REDUX-OIDC](https://github.com/maxmantz/redux-oidc) Y [HTTPS://GITHUB.COM/IDENTITYMODEL/OIDC-CLIENT-JS](https://github.com/identitymodel/oidc-client-js))

Se usa para la autenticación OpenIdConnect con Google. Funciona como cliente y permite guardar el estado en redux y LocalStorage/SessionStorage. De los dos métodos, Session Storage es más seguro, pero mantiene la sesión en una pestaña a la vez (al abrir otra no hay sesión). Por eso se eligió trabajar desde Local Storage, aunque las credenciales persisten infinitamente o hasta que el usuario las borre.

QUERY-STRING ([HTTPS://GITHUB.COM/SINDRESORHUS/QUERY-STRING](https://github.com/sindresorhus/query-string))

Se usa para parsear query strings (la parte de la url después del ?). Es sencilla y liviana de usar.

REACT-APP-REWIRED Y REACT-APP-REWIRE-MULTIPLE-ENTRY ([HTTPS://GITHUB.COM/TIMARNEY/REACT-APP-REWIRED](https://github.com/timarney/react-app-rewired))

Se usa para configurar el deploy de la aplicación. Era necesario agregar un segundo entry point a una aplicación creada con create-react-app sin usar eject para usar la funcionalidad de silent renew de oidc-client.

REACT-DIGRAPH - COMPONENTE GRAPHVIEW ([HTTPS://GITHUB.COM/UBER/REACT-DIGRAPH](https://github.com/uber/react-digraph))

Se utiliza para mostrar grafos (pantalla de planificación). No es confiable la creación de aristas así que se depende únicamente del evento de click.

REACT-ROUTER - COMPONENTES ROUTER, ROUTE, ETC ([HTTPS://GITHUB.COM/REACTTRAINING/REACT-ROUTER](https://github.com/reacttraining/react-router))

Se usa para la navegación entre pantallas. Es fácil de usar y tiene mucha documentación. Para la versión hosteada en github.io hay que usar un HashRouter en lugar de un BrowserRouter por cómo funciona el ruteo en ese sitio. Esto le trae problemas también a oidc (porque Google manda información después del hash (#) y no puede haber dos), de modo que no se pueden usar a la vez.

REDUX Y REACT-REDUX

ver bibliotecas de la app móvil

COMPUERTA

AXIOS ([HTTPS://GITHUB.COM/AXIOS/AXIOS](https://github.com/axios/axios))

Se usa en los pedidos HTTP a los microservicios.

EXPRESS-FILEUPLOAD ([HTTPS://GITHUB.COM/RICHARDGIRGES/EXPRESS-FILEUPLOAD](https://github.com/RichardGirges/express-fileupload))

Se usa para subir archivos recibidos desde el cliente. Después de un tiempo de prueba y error, fue la única alternativa que funcionó para este caso.

FORM-DATA ([HTTPS://GITHUB.COM/FORM-DATA/FORM-DATA](https://github.com/form-data/form-data))

Se usa para subir archivos recibidos desde el cliente. En combinación con la biblioteca anterior, es la única alternativa de las probadas que funcionó.

SWAGGER-UI-EXPRESS ([HTTPS://GITHUB.COM/SCOTTIE1984/SWAGGER-UI-EXPRESS](https://github.com/Scottie1984/swagger-ui-express))

Se usa para documentar la API. Con una configuración mínima, brinda una UI completa a partir de un archivo del estándar (Swagger)

ANEXO C - CASOS DE PRUEBA

En este anexo se describen los distintos casos de prueba utilizados en el desarrollo del prototipo. Se realizaron tres tipos de test: 1- Test de unidad de controllers y otras clases en la API 2- Test de integración de endpoints en la API 3- Test punta-a-punta (*end-to-end*, E2E) en la interfaz gráfica del editor web.

C.1 TEST DE UNIDAD

En el patrón MVC (*model-view-controller*) el controller es el componente que contiene la lógica de la aplicación. En este caso se quiso testear mediante *mocks* si los objetos son persistidos o no según corresponda (por ejemplo, ante información errónea no debería persistirse una actividad).

Los casos de test son los siguientes:

- Controller de dominios
 - Creación exitosa
 - Creación con nombre repetido
 - Creación enviando json malformado
 - Creación sin enviar un nombre
- Controller de actividades
 - Creación exitosa
 - Creación enviando json malformado
 - Creación con código repetido
 - Creación sin enviar un nombre
 - Creación con un dominio inválido
 - Modificación exitosa
 - Modificación enviando json malformado
 - Modificación intentando alterar el código
 - Modificación de actividad no encontrada
 - Obtención exitosa
 - Obtención de actividad no encontrada
- Controllers de listados (idiomas, tipos de tarea, etc)
 - Obtención exitosa del listado

Además, se dispone de dos mecanismos de seguridad propios de Symfony: un autenticador, que obtiene un usuario a partir de un token, y varios *voters*, que deciden si se tiene acceso a cierta funcionalidad en base a un usuario y un objeto con autor (actividades y tareas).

Los casos de prueba fueron los siguientes:

- Autenticador:
 - Verificar token presente
 - Verificar token ausente
 - Verificar formato inválido de token
 - Obtener token a partir de un header
 - Obtener autor a partir de un token
 - Obtener autor con un token de usuario
 - Obtener autor con token desconocido
- *Voter*
 - Acceso a una actividad pública propia
 - Acceso a una actividad pública definitiva ajena
 - (no) acceso a una actividad pública no definitiva ajena
 - Posesión de una actividad pública propia

- (no) posesión de una actividad pública ajena
- Acceso a una actividad privada propia
- Posesión de una actividad privada propia
- (no) acceso a una actividad privada ajena
- (no) posesión de una actividad privada ajena
- Acceso a una tarea pública propia
- Acceso a una tarea pública ajena
- Posesión de una tarea pública propia
- (no) posesión de una tarea pública ajena
- Acceso a una tarea privada propia
- Posesión de una tarea privada propia
- (no) acceso a una tarea privada ajena
- (no) posesión de una tarea privada ajena

En la Figura C.1 se muestra el resultado de correr la *suite* de tests de unidad:

```
I have no name!@4755fd8bf8e5:/usr/src/app$ bin/phpunit --testsuite unit
PHPUnit 7.5.20 by Sebastian Bergmann and contributors.

Testing
.....
Time: 480 ms, Memory: 32.00 MB
OK (41 tests, 134 assertions)
```

Figura C.1 Tests de unidad para la API

C.2 TEST DE INTEGRACIÓN

En los tests de integración se busca testear el funcionamiento de la aplicación a través de varios componentes. En este caso se quiso probar el funcionamiento de la aplicación cuando se usa una base de datos real (de prueba) y se autentican usuarios. Los casos que se probaron son los siguientes:

- Dominios
 - Guardar un dominio cuando la petición es válida
 - No guardar un dominio si ya existe el nombre
 - No guardar un dominio si no hay token presente
 - No guardar un dominio si el usuario no es un autor
 - No guardar un dominio si el token es inválido
 - No guardar un dominio si no se envió un json en la petición
 - No guardar un dominio si no se envió un nombre
- Actividades
 - Guardar una actividad cuando la petición es válida
 - No guardar una actividad si el dominio es inválido

- No guardar una actividad si el código ya existe
- No guardar una actividad si no hay token presente
- No guardar una actividad si el usuario no es un autor
- No guardar una actividad si el token es inválido
- No guardar una actividad si no se envió un json en la petición
- No guardar una actividad si no se envió un código
- Listados
 - Obtener todos los dominios
 - Obtener todos los dominios, filtrando por nombre
 - Obtener los estados posibles
 - Obtener los idiomas posibles
 - Obtener los tipos de planificación posibles
 - Obtener los tipos de tarea posibles

En la Figura C.2 se muestra el resultado de ejecutar los tests de integración para la API. Notar que tarda unas 15 veces más que los tests de unidad, con la mitad de los chequeos. Esto se debe a la interacción con la base de datos.

```
I have no name!@4755fd8bf8e5:/usr/src/app$ bin/phpunit --testsuite integration
PHPUnit 7.5.20 by Sebastian Bergmann and contributors.

Testing
.....
21 / 21 (100%)

Time: 6.82 seconds, Memory: 34.50 MB

OK (21 tests, 69 assertions)
```

Figura C.2 Tests de integración para la API

C. 3 TEST DE INTERFAZ GRÁFICA

Para testear la interfaz gráfica de la aplicación web se utilizó una herramienta llamada Cypress, que permite hacer afirmaciones respecto de elementos del DOM y peticiones HTTP usando selectores similares a los de CSS. En este caso, con estos tests se pretende validar que se muestren los títulos y campos correspondientes, que se puedan ingresar correctamente los datos y que funcione la validación de campos. También se valida que respondan correctamente los controles del editor.

Los casos de prueba son los siguientes:

- Menú
 - Mostrar botón de iniciar sesión con sesión cerrada
 - Mostrar botón de cerrar sesión con sesión iniciada
- Crear actividad
 - Mostrar correctamente botones y etiquetas
 - Ingresar texto en todos los campos
 - Seleccionar opciones
 - Enviar un formulario sin los campos obligatorios

- Crear un nuevo dominio
- Crear un dominio con un nombre repetido
- Enviar un formulario correcto
- Agregar tareas a una actividad
 - Mostrar correctamente botones y etiquetas
 - Enviar un formulario sin tareas
 - Agregar y quitar tareas, enviar el formulario
- Agregar tareas a una actividad (clonando)
 - Mostrar correctamente botones y etiquetas
 - Enviar un formulario sin tareas
 - Agregar y quitar tareas, enviar el formulario
- Actividades públicas / Mis actividades
 - Mostrar correctamente los títulos
 - Mostrar la primera página del listado de actividades
 - Comprobar link a los detalles de la actividad
- Detalles de actividad
 - Mostrar correctamente títulos y atributos
- Crear una tarea
 - Mostrar correctamente botones y etiquetas
 - Enviar un formulario sin los campos obligatorios
 - Ingresar texto en todos los campos
 - Seleccionar opciones
 - Crear un nuevo dominio
 - Crear un dominio con nombre repetido
 - Enviar un formulario completo
 - Enviar un formulario de opción múltiple sin opciones
 - Crear una tarea posicionada con mapa
- Editar planificación
 - Mostrar correctamente títulos y botones
 - Mostrar menú de tarea
 - Mostrar menú de conexión
 - Mostrar las tareas disponibles
 - Conectar nodos
 - Generar el grafo de prueba 1 (Figura C.3.1)
 - Generar el grafo de prueba 2 (Figura C.3.2)

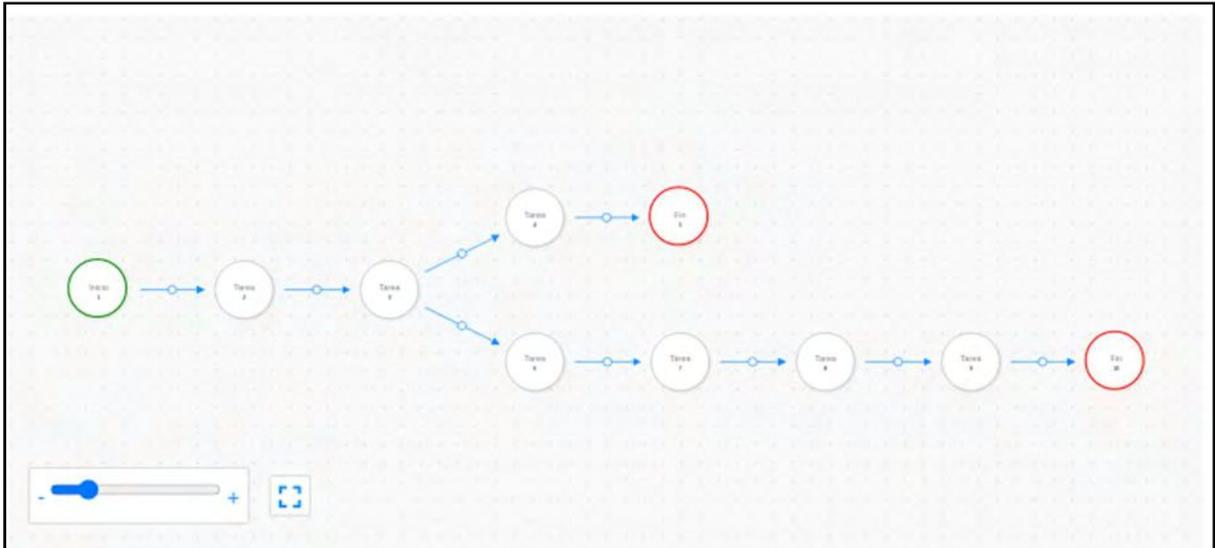


Figura C.3.1 Grafo de prueba 1

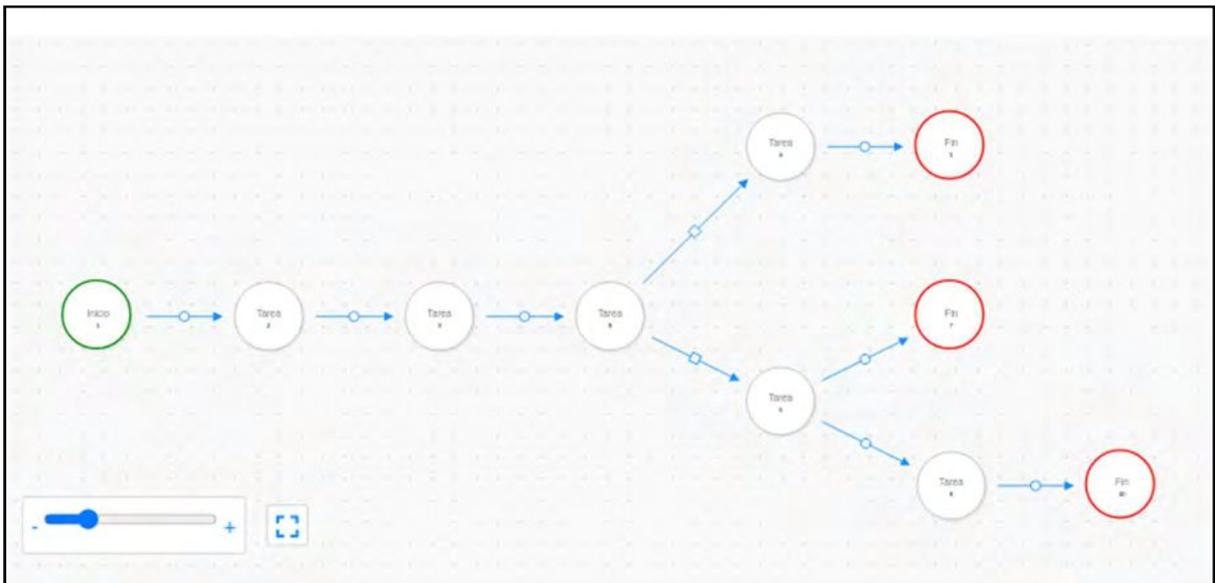


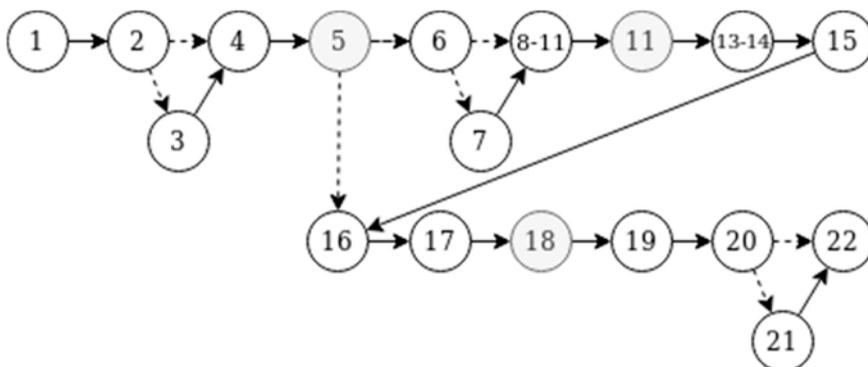
Figura C.3.2 Grafo de prueba 2

ANEXO D - PLANIFICACIÓN DEL CASO DE ESTUDIO ORIGINAL

En este anexo se incluye el caso de estudio original en el que fue instanciada la plataforma.

NOMBRE DE LA ACTIVIDAD: ENTREGA DE ELEMENTOS EN DESUSO

OBJETIVO: NOTIFICAR A E-BASURA DE LA ENTREGA DE EQUIPAMIENTO



PERSONA DE CONTACTO (1)

Consigna: Ingresar el nombre de la persona de contacto

Tipo: Ingresar texto

TIPO DE DONACIÓN (2)

Consigna: Elegir el tipo de donación

Tipo: Elegir una opción

Elementos:

- Personal
- Institucional

Si se eligió "Personal", pasar a (4). Si se eligió "Institucional", pasar a (3)

SOBRE EL TELÉFONO DE CONTACTO (3)

Consigna: El teléfono que se va a ingresar, ¿es particular o institucional?

Tipo: Elegir una opción

Elementos:

- Particular
- Institucional

TELÉFONO DE CONTACTO (4)

Consigna: Ingresar el teléfono de contacto

Tipo: Ingresar texto

CORREO ELECTRÓNICO DE CONTACTO (5) - OPCIONAL

Consigna: Ingrese un correo electrónico de contacto

Tipo: Ingresar texto

Si se pasó por la tarea (3) pasar a (6). Si no, pasar a (16)

TIPO DE INSTITUCIÓN (6)

Consigna: Elegir el tipo de empresa / organización

Tipo: Elegir una opción

Elementos:

- Empresa privada
- Empresa pública
- Organismo público nacional
- Organismo público provincial
- Organismo público municipal
- OSC/ONG
- Otro

Si se elige "Otro" pasar a la tarea (7). Si no, pasar a la tarea (8)

OTRA INSTITUCIÓN (7)

Consigna: Escribir el tipo de institución

Tipo: Ingresar texto

NOMBRE DE LA EMPRESA (8)

Consigna: Ingresar el nombre de la empresa

Tipo: Ingresar texto

PROVINCIA (9)

Consigna: Elegir una provincia

Tipo: Elegir una opción

Elementos:

- (las 23 provincias y CABA)

LOCALIDAD (10)

Consigna: Escribir la localidad

Tipo: Ingresar texto

DIRECCIÓN INSTITUCIONAL (11)

Consigna: Escribir la dirección

Tipo: Ingresar texto

TELÉFONO INSTITUCIONAL (12) - OPCIONAL

Consigna: Ingresar un teléfono institucional

Tipo: Ingresar texto

CORREO ELECTRÓNICO INSTITUCIONAL (13)

Consigna: Ingresar un correo electrónico institucional

Tipo: Ingresar texto

ACTIVIDAD (14)

Consigna: Grabar un audio explicando la actividad desarrollada

Tipo: Grabar audio

RSE (15)

Consigna: ¿Posee Departamento de Responsabilidad Social Empresaria (RSE)?

Tipo: Elegir una opción

Elementos:

- Sí
- No

EQUIPAMIENTO I (16)

Consigna: Seleccione los tipos de equipamiento a donar

Tipo: Opción múltiple

Elementos:

- CPUs
- Monitores CRT (Tubo)
- Monitores LCD
- Mouses
- Teclados
- Impresoras

- Notebooks
- Celulares
- Otros

EQUIPAMIENTO II (17)

Consigna: Ingrese cuántos elementos tiene para entregar de cada categoría

Tipo: Contadores

- Criterio: Elementos a entregar
- Mensaje: Cantidad de elementos a entregar
- Elementos: (nombre: coeficiente)
 - CPUs: 1
 - Monitores CRT (Tubo): 1
 - Monitores LCD: 1
 - Mouses: 1
 - Teclados: 1
 - Impresoras: 1
 - Notebooks: 1
 - Celulares: 1

OTROS ELEMENTOS (18) - OPCIONAL

Consigna: Ingresar, si los hubiese, nombre y cantidad de otros elementos

Tipo: Ingresar texto

FOTO ELEMENTOS (19)

Consigna: Sacar una foto de los elementos a entregar

Tipo: sacar foto

RETIRO (20)

Consigna: Elegir la opción que corresponda

Tipo: Elegir una opción

Elementos:

- Necesito que retiren el equipamiento
- No necesito que pasen a retirarlo

Si se elige "Necesito..." pasar a (21), si no, pasar a (22)

UBICACIÓN RETIRO (21)

Consigna: Ingresar la ubicación de retiro

Tipo: Localización

MUCHAS GRACIAS (22)

Consigna: Muchas gracias por su donación. Recibirá nuestra llamada en breve.

Tipo: Simple

ANEXO E - MATERIALES PARA LAS PRUEBAS DE USUARIO

A continuación, se reproducen la guía para resolver las pruebas de usuario y la encuesta incluyendo el SUS mencionados en el Capítulo 6.

CASO DE USO DONACIÓN DE RESIDUOS DE APARATOS ELÉCTRICOS Y ELECTRÓNICOS (RAEE).

Antes de comenzar, queremos agradecer su interés en participar en este ejercicio para probar en conjunto la plataforma DEHIA, que permite crear, de manera simplificada, una actividad para luego usarla desde una Aplicación Móvil (APP)

Le contamos que los Residuos de Aparatos Eléctricos y Electrónicos, representan un gran problema ambiental y desde ya hace varios años se creó un programa en la Universidad Nacional de La Plata que, entre otras actividades, recolecta este tipo de residuos para intentar recuperarlos y entregarlos a instituciones sin fines de lucro.

En este ejercicio que llevaremos adelante, vamos a suponer que queremos diagramar una actividad para que sea realizada más adelante por diferentes personas. Esta actividad tendrá un conjunto de tareas que serán organizadas de una manera determinada.

Para lograr el objetivo, vamos a establecer el ejercicio en 3 pasos:

Paso 1) Crear una actividad usando un conjunto de datos

Paso 2) Para la actividad creada en el paso anterior, crear un conjunto de tareas de acuerdo a una lista definida

Paso 3) Organizar las tareas de la actividad de acuerdo a ciertas condiciones que le serán explicadas.

¿Iniciamos?

Paso 1) Crear una actividad

- En el menú de la parte superior, elija la opción **Crear Actividad**
- La actividad a crear, lleva el nombre de **Recolección de RAEE**.
- El idioma en el cual se redactarán las tareas posteriormente es el **ESPAÑOL**.
- El dominio de trabajo es el de **E-BASURA**.
- El objetivo consiste en: **Recolectar RAEE para determinar cuál o cuáles de ellos pueden ser reutilizados para donaciones**.
- El tipo de planificación a usar es **BIFURCADA**, ya que esto permite organizar las tareas con ciertas condiciones que definiremos más adelante.

- El estado de esta actividad, será **PRIVADO**, ya que, en principio, no lo dejaremos disponible para otros creadores que accedan a DEHIA.
- **Guardar** la actividad creada.

Una vez guardada la actividad que acaba de crear, ya estará listo para **Continuar**

Paso 2) Crear las tareas a ser realizadas para la actividad creada en el paso 1

Tarea	1	2	3	4	5	6
Tema	Tipo Donación	teléfono	email	Equipamiento	Foto	Muchas gracias
Tipo	opciones	texto	texto	múltiple	foto	simple

Paso 2.1)

- Para crear la primera de las tareas deberá elegir la opción de **Nueva**
- A continuación, se dan los datos para crear la primera de las tareas, en la que el usuario posteriormente, deberá cargar:
 - Nombre de la tarea **Tipo de Donación**
 - Consigna **Elegir el tipo de donación**
 - El tipo de la tarea será: **Elegir una opción**
 - El estado de la tarea es **Privado**
 - El dominio, es **E-BASURA**
 - A continuación, se pondrán las opciones posibles de elección, a las que llamaremos elementos
 - Personal
 - Institucional
- Para terminar, elija la opción de **Guardar**.

Paso 2.2)

- Para crear la segunda tarea, elija la opción de **Continuar**
- Al cambiar de pantalla, elija nuevamente la opción de **Nueva**
- A continuación, se dan los datos para la segunda tarea:
 - Nombre: **Teléfono de contacto**
 - Consigna: **Ingresar el teléfono de contacto**
 - El tipo de tarea es **Ingresar texto**
 - El estado de la tarea es **Privado**
 - El dominio es **E-BASURA**
- Para terminar, elija **Guardar**

Paso 2.3)

- Para crear la tercera tarea, elija la opción de **Continuar**
- Al cambiar de pantalla, elegir la opción **Nueva**

- Los datos para la tercera tarea son:
 - Nombre **Correo electrónico de contacto**
 - Consigna **Ingrese un correo electrónico de contacto**
 - El tipo de tarea es **Ingresar texto**
 - El estado de la tarea es **Privado**
 - El dominio es **E-BASURA**
- Para terminar, elija **Guardar**

Paso 2.4)

- Para crear la cuarta tarea, elija la opción **Guardar**
- Al cambiar de pantalla, elegir la opción **Nueva**
- Los datos para la cuarta tarea son:
 - Nombre **Equipamiento**
 - Consigna **Seleccione los tipos de equipamiento a donar**
 - Tipo de tarea **Opción Múltiple**
 - Estado **Privado**
 - Dominio **E-BASURA**
 - A continuación, crear los elementos:
 - Mouses
 - Teclados
 - Impresoras
 - Celulares
- Para terminar, elija **Guardar**

Paso 2.5)

- Para cargar la quinta tarea, primero elija la opción **Continuar**
- En la nueva pantalla, elija la opción **Nueva**
- Los datos para cargar la tarea son:
 - Nombre **Foto elementos**
 - Consigna **Sacar una foto de los elementos a entregar**
 - Tipo de tarea **Sacar foto**
 - Estado **Privado**
 - Dominio **E-BASURA**
- Para terminar, elija la opción **Guardar**

Paso 2.6)

- Para cargar la sexta tarea, primero elija la opción **Continuar**
- En la nueva pantalla, elija la opción **Nueva**
- Los datos para esta tarea son:
 - Nombre **Muchas gracias**
 - Consigna **Muchas gracias por su donación. Recibirá nuestra llamada en breve.**
 - Tipo de tarea **Simple**

- Estado **Privado**
- Dominio **E-Basura**
- Para terminar, elija la opción **Guardar**, y luego **Continuar**
- Para concluir la carga de tareas, elegir la opción **Guardar** y luego **Continuar**

Paso 3) Organizar las tareas creadas en el paso 2

Paso 3.1)

- Elegir el primer círculo contando desde arriba, que dice **Inicio 1**
- En el menú que se abre, elegir **agregar conexiones**
- Donde dice hacia la tarea... Elegir siguiente elegir **2 - Teléfono de contacto**
- Hacer click en (tildar) el cuadrado donde dice Mostrar condición
- Donde dice Cuando... Elegir... elegir **Sí se elige**
- Donde dice la opción... Elegir... elegir **Institucional**
- Elegir **Agregar Conexión**

Paso 3.2)

- Elegir el primer círculo verde, que dice **Inicio 1**
- En el menú que se abre, elegir **agregar conexiones**
- Donde dice hacia la tarea... Elegir siguiente elegir **3 - Correo electrónico de contacto**
- Hacer click en (tildar) el cuadrado donde dice Mostrar condición
- Donde dice Cuando... Elegir... elegir **Sí se elige**
- Donde dice *la opción... Elegir...* elegir **Personal**
- Elegir **Agregar Conexión**

Paso 3.3)

- Si los círculos quedan fuera de pantalla, se puede mover el diagrama arrastrando el fondo (fuera de los círculos)
- Elegir el círculo de más arriba a la derecha, rojo que dice **Fin 2**
- En el menú que se abre, elegir **agregar conexiones**
- Donde dice *hacia la tarea... Elegir siguiente* elegir **4 - Equipamiento**
- Elegir **Agregar Conexión**

Paso 3.4)

- Elegir el círculo de rojo que dice **Fin 3** (el de más bajo)
- En el menú que se abre, elegir **agregar conexiones**
- Donde dice *hacia la tarea... Elegir siguiente* elegir **4 - Equipamiento**
- Elegir **Agregar Conexión**

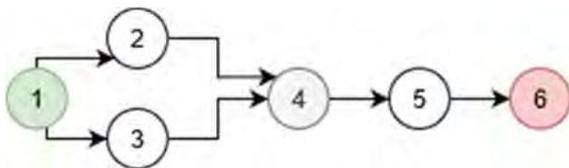
Paso 3.5)

- Elegir el círculo de más a la derecha, rojo, que dice **Fin 4**
- En el menú que se abre, hacer click en (tildar) el cuadrado donde dice **Opcional** (es la opción de más arriba)
- Elegir **Agregar conexiones**
- Donde dice *hacia la tarea...* Elegir siguiente elegir **5 - Foto elementos**
- Elegir **Agregar Conexión**

Paso 3.6)

- Elegir el círculo de más a la derecha, rojo, que dice **Fin 5**
- En el menú que se abre, elegir **agregar conexiones**
- Donde dice *hacia la tarea...* Elegir siguiente elegir **6 - Muchas gracias**
- Elegir **Agregar conexión**

El dibujo debería haber quedado más o menos así:



Paso 3.10)

- Debajo del gráfico hay un botón de **Guardar**, presionarlo
- Finalmente elegir **Continuar**

Con esto termina la prueba. ¡**Muchas gracias!**

ENCUESTA POST-PRUEBA

DEHIA - Pruebas de usuario

Encuesta para usuarios de prueba

*Obligatorio

¿Alguna vez modificó una aplicación que usa regularmente? (por ejemplo una macro de excel, un filtro de correo)

Sí

No

Por favor elija la opción que más represente su opinión sobre cada afirmación
Las opciones van desde En completo desacuerdo (NO) hasta Completamente de acuerdo (SÍ). Son 10 afirmaciones.

1 - Creo que usaría esta aplicación frecuentemente *

- En completo desacuerdo
- En desacuerdo
- Neutral
- De acuerdo
- Completamente de acuerdo

2 - Encuentro esta aplicación innecesariamente compleja *

- En completo desacuerdo
- En desacuerdo
- Neutral
- De acuerdo
- Completamente de acuerdo

3 - Creo que la aplicación fue fácil de usar *

- En completo desacuerdo
- En desacuerdo
- Neutral
- De acuerdo
- Completamente de acuerdo

4 - Creo que necesitaría ayuda de una persona con conocimientos técnicos para usar esta aplicación *

- En completo desacuerdo
- En desacuerdo
- Neutral
- De acuerdo
- Completamente de acuerdo

5 - Las funciones de esta aplicación están bien integradas *

- En completo desacuerdo
- En desacuerdo
- Neutral
- De acuerdo
- Completamente de acuerdo

6 - Creo que la aplicación es muy inconsistente *

- En completo desacuerdo
- En desacuerdo
- Neutral
- De acuerdo
- Completamente de acuerdo

7 - Imagino que la mayoría de la gente aprendería a usar esta aplicación en forma muy rápida *

- En completo desacuerdo
- En desacuerdo
- Neutral
- De acuerdo
- Completamente de acuerdo

8 - Encuentro que la aplicación es muy difícil de usar *

- En completo desacuerdo
- En desacuerdo
- Neutral
- De acuerdo
- Completamente de acuerdo

9 - Me siento con confianza al usar esta aplicación *

En completo desacuerdo

En desacuerdo

Neutral

De acuerdo

Completamente de acuerdo

10 - Necesité aprender muchas cosas antes de ser capaz de usar esta aplicación *

En completo desacuerdo

En desacuerdo

Neutral

De acuerdo

Completamente de acuerdo

Enviar

ANEXO F - RESULTADOS DE LAS PRUEBAS

En este anexo se detallarán los resultados obtenidos de las respuestas y los tiempos de resolución de la prueba, que fueron presentados de forma resumida en el Capítulo 6.

Se calcula el resultado del SUS de acuerdo al puntaje de cada pregunta (restando 1 a las preguntas positivas, restando a 5 las negativas y multiplicando el promedio por 2,5)

Participantes	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	Resultado SUS
p1	3	1	5	2	3	1	5	2	4	1	82,5
p2	1	3	3	5	5	1	3	3	2	1	52,5
p3	3	2	4	1	5	1	4	2	4	2	80,0
p4	5	2	4	2	5	4	5	1	4	1	82,5
p5	3	3	4	4	4	2	2	3	4	4	52,5

Tabla F.1

P1 y P3 (20%) afirmaron no ser experimentados en configuración de software.

A continuación se presenta la frecuencia de respuesta para cada pregunta, primero de forma global (Tabla F.2) y luego discriminando en participantes no experimentados (Tabla F.3) y experimentados (Tabla F.4).

1	2	3	4	5	
20%	0%	60%	0%	20%	Creo que usaría esta aplicación frecuentemente
20%	40%	40%	0%	0%	Encuentro esta aplicación innecesariamente compleja
0%	0%	20%	60%	20%	Creo que la aplicación fue fácil de usar
20%	40%	0%	20%	20%	Creo que necesitaría ayuda de una persona con conocimientos técnicos para usar esta aplicación
0%	0%	20%	20%	60%	Las funciones de esta aplicación están bien integradas
60%	20%	0%	20%	0%	Creo que la aplicación es muy inconsistente
0%	20%	20%	20%	40%	Imagino que la mayoría de la gente aprendería a usar esta aplicación en forma muy rápida
20%	40%	40%	0%	0%	Encuentro que la aplicación es muy difícil de usar
0%	20%	0%	80%	0%	Me siento con confianza al usar esta aplicación
60%	20%	0%	20%	0%	Necesité aprender muchas cosas antes de ser capaz de usar esta aplicación

Tabla F.2

1	2	3	4	5	
0%	0%	100%	0%	0%	Creo que usaría esta aplicación frecuentemente
50%	50%	0%	0%	0%	Encuentro esta aplicación innecesariamente compleja
0%	0%	0%	50%	50%	Creo que la aplicación fue fácil de usar
50%	50%	0%	0%	0%	Creo que necesitaría ayuda de una persona con conocimientos técnicos para usar esta aplicación
0%	0%	50%	0%	50%	Las funciones de esta aplicación están bien integradas
100%	0%	0%	0%	0%	Creo que la aplicación es muy inconsistente
0%	0%	0%	50%	50%	Imagino que la mayoría de la gente aprendería a usar esta aplicación en forma muy rápida
0%	100%	0%	0%	0%	Encuentro que la aplicación es muy difícil de usar
0%	0%	0%	100%	0%	Me siento con confianza al usar esta aplicación
50%	50%	0%	0%	0%	Necesité aprender muchas cosas antes de ser capaz de usar esta aplicación

Tabla F.3

1	2	3	4	5	
33%	0%	33%	0%	33%	Creo que usaría esta aplicación frecuentemente
0%	33%	67%	0%	0%	Encuentro esta aplicación innecesariamente compleja
0%	0%	33%	67%	0%	Creo que la aplicación fue fácil de usar
0%	33%	0%	33%	33%	Creo que necesitaría ayuda de una persona con conocimientos técnicos para usar esta aplicación
0%	0%	0%	33%	67%	Las funciones de esta aplicación están bien integradas
33%	33%	0%	33%	0%	Creo que la aplicación es muy inconsistente
0%	33%	33%	0%	33%	Imagino que la mayoría de la gente aprendería a usar esta aplicación en forma muy rápida
33%	0%	67%	0%	0%	Encuentro que la aplicación es muy difícil de usar
0%	33%	0%	67%	0%	Me siento con confianza al usar esta aplicación
67%	0%	0%	33%	0%	Necesité aprender muchas cosas antes de ser capaz de usar esta aplicación

Tabla F.4

En la Tabla F.5 se indican los tiempos de creación de tareas (t1) y de planificación (t2) junto con sus promedios discriminados según si tenían (SI) o no (NO) experiencia.

	p1	p2	p3	p4	p5	promedio	promedio no	promedio si
t1	19	10	11	12	10	12,4	15	10,66667
t2	5	3	4	3	3	3,6	4,5	3
total	24	13	15	15	13	16	19,5	13,66667

Tabla F.5

ANEXO G - ACERCA DE LOS ARCHIVOS FUENTE

Los archivos fuente están disponibles en un repositorio público de GitHub que puede encontrarse en <https://github.com/mokocchi/DEHIA>.

Cada directorio originalmente era un repositorio con su mismo nombre que pueden encontrarse en este mismo usuario de GitHub.

En `Artículos relacionados` pueden encontrarse los artículos publicados en torno al objetivo de la tesina.

El código está organizado de la siguiente manera:

- El cliente web está en `prototipo-app-actividades` y se inicia con `yarn install; yarn start` (previa instalación de yarn)
- El gateway está en `dehia_gateway` y se inicia con `docker-compose up` (previa instalación de docker y docker compose)
- Los microservicios están en `dehia_auth`, `dehia_define`, `dehia_collect` y `dehia_results`. Se inician con `docker-compose up`
- La aplicación móvil está en `prototipo-app-actividades` y se inicia con `react-native start; react-native run-android`
- En `dehia_mock`s hay mocks de reemplazo de los servicios `define` y `auth`, en caso de querer probar solamente el cliente.