UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

MEJORA DE LA USABILIDAD DE UNA HERRAMIENTA DE MODELAMIENTO DE INTERACCIONES EN SISTEMAS COLABORATIVOS MÓVILES

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN

ALEJANDRO ADRIÁN LUMÁN BAHAMONDES

PROFESOR GUÍA:
SERGIO OCHOA DELORENZI

PROFESOR CO-GUÍA:
DANIEL PEROVICH GEROSA

MIEMBROS DE LA COMISIÓN:
EDUARDO GODOY VEGA
ÉRIC PIERRE TANTER

SANTIAGO DE CHILE
2022

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL EN
COMPUTACIÓN
POR: Alejandro Adrián Lumán Bahamondes
FECHA: 2022
PROFESOR GUÍA: Sergio Ochoa Delorenzi

MEJORA DE LA USABILIDAD DE UNA HERRAMIENTA DE MODELAMIENTO DE INTERACCIONES EN SISTEMAS COLABORATIVOS MÓVILES

Actualmente se ha expandido en gran forma el uso de aplicaciones móviles que apoyan procesos colaborativos entre personas. Este tipo de aplicaciones permite que dos o más usuarios -que pueden cumplir diversas funciones o "roles"- puedan interactuar de forma no estructurada para cumplir un objetivo común. Por lo general, estas aplicaciones dividen su diseño en dos partes: un ambiente compartido (*shared workspace*) y un ambiente de soporte a las interacciones entre los usuarios.

Para poder diseñar estos ambientes de soporte a las interacciones, se han creado diversas notaciones y lenguajes de modelamiento que permiten representar escenarios de interacción. Una de estas notaciones es CIMoN (*Computer-Supported Interaction Modeling Notation)*, que es sobre la cual se basa este trabajo. En esta notación, cada usuario es representado por un nodo de un grafo, y las posibles interacciones entre usuarios son representadas por aristas. Se cuenta con una aplicación *web* (editor gráfico) que apoya el modelamiento de los escenarios de interacción. Sin embargo, esta herramienta carece de algunas capacidades importantes, y tiene problemas de usabilidad.

Por lo tanto, en este trabajo de título se realiza una reingeniería de esta herramienta *web,* mejorando la usabilidad y haciendo más rápido y cómodo el modelamiento de las interacciones. Además, se agregan a la aplicación características importantes que le faltaban, como, por ejemplo, un sistema de manejo de usuarios.

Durante el proceso, se rediseñan completamente las vistas de la aplicación legada, además de añadirse nuevas características. También, se tuvo especial cuidado en escribir código mantenible y de fácil comprensión, permitiendo posibles extensiones de forma fácil de cara al futuro.

Para evaluar el trabajo realizado se hace uso de dos técnicas. En primer lugar, se utiliza una evaluación de *keystrokes* (KLM)*,* en donde se cuenta el número de *clicks* o movimientos del *mouse* que realiza el usuario. La segunda evaluación involucra a usuarios reales a quienes se les aplica una encuesta SUS modificada. En ambos casos se compara la aplicación legada con la herramienta desarrollada en esta memoria.

Los resultados de las evaluaciones indican que la herramienta implementada alcanza un nivel de usabilidad muy bueno, además de mostrar reducciones en el tiempo que toma realizar acciones comunes. A pesar de esto, existen mejoras posibles de realizar a futuro, además de añadir algunas funcionalidades indicadas como deseables.

Tabla de contenido

1. Introducción

En la actualidad, existe un importante número de aplicaciones de software que pretenden apoyar procesos colaborativos entre personas. Ejemplos populares de este tipo de aplicaciones son *Uber*, que permite a pasajeros encontrar choferes para un viaje programado, o *Cornershop*, que permite a personas realizar una lista de compras basada en productos de ciertos supermercados y tiendas, que luego será utilizada por otra persona registrada para realizar las compras de manera presencial, y entregarla en un domicilio acordado.

Muchas de estas aplicaciones apoyan procesos desestructurados, es decir, donde no hay un flujo de interacciones fijo entre los participantes de dichos procesos. La interacción entre los actores se lleva a cabo de manera espontánea, y sólo si hay una necesidad de comunicación entre ellos.

A pesar de no tener un flujo de trabajo predefinido, estos procesos suelen tener etapas o fases, y requieren que las aplicaciones que las apoyan permitan una participación flexible de distintos actores con el fin de hacer al proceso más efectivo.

Las aplicaciones que apoyan estos procesos dividen su diseño en dos grandes ámbitos que son complementarios (Figura 1.a): 1) el ambiente compartido (*shared workspace*) y 2) el ambiente de soporte a las interacciones que puedan llevar a cabo diferentes grupos de usuarios con determinados *roles*. La Figura 1.b muestra la interfaz del espacio de trabajo compartido de Uber, la cual brinda acceso a la lógica de negocio propia de la aplicación (o servicios de negocio).

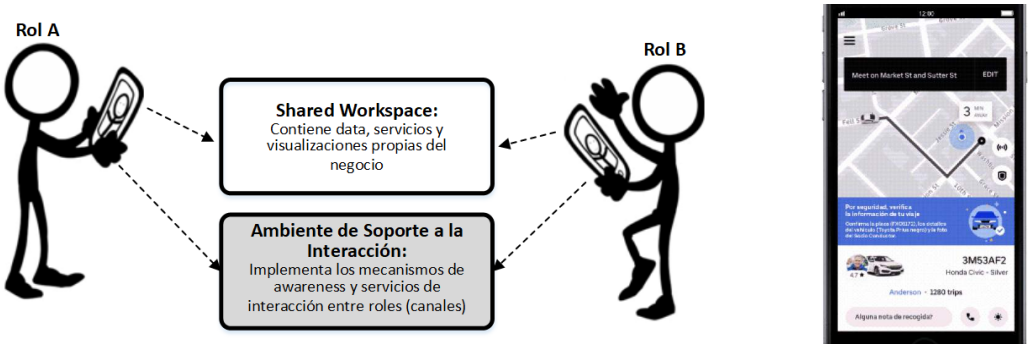

Figura 1. (a) Componentes de un sistema colaborativo móvil; (b) Interfaz del *workspace* de Uber.

El modelamiento del espacio de trabajo compartido puede ser abordado con herramientas tradicionales de ingeniería de software. Sin embargo, el modelamiento del ambiente de soporte a la interacción ha demostrado ser complejo, y difícil de abordar con las herramientas, lenguajes y notaciones disponibles.

1.1. Diseño de Interacciones en Aplicaciones Colaborativas Móviles

El diseñar un sistema colaborativo de este tipo involucra identificar cuatro componentes importantes: los actores, las actividades que estos realizan, las necesidades de interacción entre ellos, y la percepción del contexto (o *awareness*) que la aplicación debe brindar a los participantes. Este último componente es el que mantiene a los participantes informados sobre el proceso, para mediar de mejor manera la colaboración entre ellos.

Todos estos aspectos deben tenerse en cuenta desde la fase de concepción de la solución o preventa del proyecto, pues allí el desarrollador y el cliente deben ponerse de acuerdo respecto a las funcionalidades que debe poseer la aplicación (que engloba tanto las interacciones entre los actores, como otras funciones); es decir, determinar el alcance del proyecto. Idealmente este proceso de definición de la funcionalidad debería ser expedito y de bajo costo para los desarrolladores, puesto que estos últimos no cobran por los presupuestos que entregan. Además, el resultado de este proceso debe permitir la comprensión mutua (entre cliente y desarrollador) acerca de lo que el producto debe hacer en términos de funcionalidad.

Para poder determinar la funcionalidad de una aplicación colaborativa móvil de forma efectiva, se han creado varias técnicas que permiten representar los diferentes escenarios de interacción, validando los roles y necesidades. Algunas de estas técnicas permiten también derivar requisitos de software para el desarrollo de las aplicaciones que los apoyan.

Particularmente, este trabajo de memoria apoya a técnicas de modelamiento que utilizan la notación *Computer-Supported Interaction Modeling Notation* (CIMoN) [1] para representar escenarios de interacción a ser apoyados por una aplicación móvil. Cada escenario se especifica a través de un grafo, el cual vincula actores (nodos del grafo) e interacciones entre ellos (arcos entre nodos), utilizando la notación antes mencionada.

1.2. Situación Inicial

Para realizar la definición conjunta entre el cliente y el desarrollador acerca del alcance (funcionalidad) del producto, particularmente en lo que se refiere al ámbito del soporte de interacciones entre los participantes, actualmente se cuenta con una herramienta web[1]. Esta herramienta permite modelar los procesos colaborativos, los escenarios de interacción entre los roles participantes, y además genera mockups de una aplicación móvil para apoyar el proceso modelado. Los mockups incluyen solamente el aspecto de interacción y *awareness* en las interfaces de usuario.

Esta herramienta web fue desarrollada parcialmente en el curso CC5401: Ingeniería de Software II, de la carrera de Ingeniería Civil en Computación, y también como parte de la tesis de doctorado de Maximiliano Canché [2]. Si bien la herramienta cumple con las características básicas que permiten poner en práctica la técnica de modelado, actualmente posee varios problemas de usabilidad, y además carece de diversas capacidades que le permitirían mejorar su completitud y utilidad.

[1] https://cimonv3.dcc.uchile.cl/

En primer lugar, la aplicación no cuenta con un sistema de manejo de usuarios, ni ningún tipo de mecanismo de seguridad. Esto conlleva a que tanto los procesos creados, como todos los grafos dentro de estos, puedan ser modificados por cualquier persona que tenga el enlace a la plataforma.

Otra capacidad importante que falta es poder representar (de forma básica) la lógica de negocio de los procesos, particularmente el espacio compartido de las aplicaciones móviles, de forma de poder mostrar *mockups* más similares a una aplicación real. Actualmente los *mockups* permiten visualizar solamente el soporte a la interacción.

Por otra parte, la aplicación cuenta con varios problemas de usabilidad. Por ejemplo, la navegación general es deficiente, y en algunas vistas no existe un botón para volver a la interfaz anterior. Además, el uso del editor de interacciones es poco intuitivo, y la semántica de muchas opciones no es obvia. Otra limitante consiste en que la vista que contiene las fases y ámbitos del proceso modelado no deja en claro el propósito de esta, y las opciones que existen para modelar el proceso tampoco son muy evidentes.

Adicionalmente, la calidad del código de la herramienta presenta algunos problemas estructurales y de diseño, los cuales hacen difícil la mantenibilidad y extensibilidad de esta. Otro problema importante es que algunas librerías usadas en la implementación de esta versión del modelador tienen dependencias deprecadas.

Tal como se indicó antes, el objetivo de usar esta herramienta en la fase de preventa de un proyecto radica en poder definir, de forma temprana, las funcionalidades a ser implementadas. En esta definición participan los representantes del cliente y del desarrollador. Debido a que ésta ocurre durante la fase de preventa del proyecto, es de vital importancia que dicha definición se realice de la forma más rápida y eficiente posible, puesto que este proceso puede llevar costes importantes (en tiempo y dinero), tanto a los desarrolladores como a los clientes. Por esto mismo, la usabilidad y la experiencia de usuario que la herramienta entregue a los participantes en esta definición, son esenciales para promover un uso efectivo de la aplicación.

Para abordar los problemas antes planteados, se hace necesario realizar un proceso de reingeniería de la aplicación actual. Éste implica llevar a cabo un trabajo exploratorio sobre herramientas similares, elaboración de prototipos, la reimplementación del *frontend* y *backend* de la herramienta, y finalmente, evaluar tanto la usabilidad, como de cumplimiento de requisitos de software por parte de la herramienta.

1.3. Objetivos de la Memoria

El objetivo principal de este trabajo de memoria es realizar un proceso de reingeniería y extensión a la actual herramienta de modelado antes descrita, buscando mejorar la usabilidad y experiencia de usuario, así como en la mantenibilidad y extensibilidad a futuro de dicha aplicación. Además, se agregan características útiles de las cuales carece actualmente la herramienta.

Para alcanzar este objetivo general se definieron los siguientes objetivos específicos:

- Reevaluar las interfaces de usuario actuales y definir mejoras, principalmente en el ámbito de navegación dentro de la aplicación y la comprensión de las vistas actuales de procesos, fases y ámbitos.
- Definir un modelo genérico para las interfaces de aplicaciones que apoyan procesos colaborativos entre personas (roles), y que en dichas interfaces se represente el espacio para desplegar la funcionalidad propia del proceso de negocio que éstas apoyan.
- Reimplementar el *frontend* de la actual herramienta, teniendo especial cuidado en la calidad del código, lo que permitirá una mejor mantención a futuro. Además, durante este proceso, implementar las mejoras a interfaces y el modelo genérico de la lógica de negocio, lo que permitirá que los *mockups* generados se asemejen de mayor forma a una aplicación real.
- Reimplementar el *backend* de la aplicación, agregando características para el manejo de usuarios, y un sistema de seguridad y privacidad, pues la herramienta legada no lo posee.

1.4. Evaluación de las Nuevas Interfaces de Usuario

Para la evaluación de la mejora de las interfaces de usuario, se consideró el uso de las siguientes técnicas:

- Evaluación de *keystrokes* [3]: Esta técnica permite evaluar el número de clicks o acciones requeridas para realizar determinadas tareas, utilizando las interfaces del sistema. Usando esta técnica se puede realizar una comparación entre lo requerido por la herramienta legada, versus la nueva versión de esta.
- Evaluación *think-aloud* [4]: Esta técnica permite ver cómo usuarios finales se comportan frente a ambas aplicaciones, y recibir los comentarios que tengan sobre éstas mientras realizan acciones específicas.
- Evaluación *heurística* [5]: Luego de la evaluación usando *think-aloud*, los usuarios podrían evaluar características específicas asociadas a la experiencia de usuario, utilizando una escala de 1 a 5.
- Evaluación con la escala *SUS (System Usability Scale)* [6]: Esta escala permite, de manera simple y rápida, medir la usabilidad según una escala determinada. Para esto es necesario aplicar un cuestionario de 10 ítems o aseveraciones, que expresan un juicio de valor con respecto a la plataforma web que se quiere medir. Quien conteste este cuestionario debe indicar su nivel de acuerdo o desacuerdo con cada afirmación que se presenta, contestando con un valor entre 1 y 5; siendo 1 la respuesta de estar "completamente en desacuerdo", y 5 cuando se está "completamente de acuerdo" con lo expresado en el ítem.

Luego de analizar pros y contras de estas técnicas de evaluación, se decide utilizar la encuesta SUS [6] para determinar el valor de las nuevas interfaces del sistema, principalmente por razones de simplicidad y pertinencia. Además, para obtener métricas de las nuevas interfaces en comparación a las antiguas, también se hace uso de evaluación

mediante *keystrokes* [3]. Por otra parte, los nuevos servicios de la herramienta se evalúan según el cumplimiento de estos, con respecto a los requisitos definidos en el capítulo 3.

1.5. Estructura de la Memoria

A continuación, se presentan los puntos abordados en los próximos capítulos de este documento:

- *Capítulo 2: Marco teórico*. En este capítulo se introducen los conceptos requeridos para entender el funcionamiento de la herramienta. Además, se describe la nomenclatura visual utilizada por el editor de grafos.
- *Capítulo 3: Análisis del sistema de legado*. En este capítulo se analizará la aplicación ya existente al momento de iniciar el trabajo, tanto en su capa de *frontend* como *backend*. Además, se explicará en más detalle por qué esta implementación es poco satisfactoria y por qué se requiere un proceso de reingeniería.
- *Capítulo 4: Concepción de la solución*. En esta sección se detalla cómo fue llevada a cabo la implementación de la nueva aplicación *web,* considerando las tecnologías utilizadas, los requisitos de software considerados, además de su arquitectura y su modelo de datos.
- *Capítulo 5: Rediseño de las interfaces de usuario*. Aquí se detallan las nuevas interfaces de la aplicación, explicando su funcionamiento y uso utilizando imágenes como soporte.
- *Capítulo 6: Evaluación de la nueva herramienta*. En este apartado se detalla el proceso de evaluación y validación de la implementación de la herramienta, además de los resultados obtenidos en esta.
- *Capítulo 7: Conclusiones y trabajo a futuro*. En este capítulo se analizan los resultados obtenidos y el éxito el cual estos tuvieron en relación con los objetivos planteados. Además, se discute sobre mejoras o extensiones que podría tener la herramienta a futuro y posibles pasos para lograrlo.

2. Marco Teórico

Antes de iniciar el análisis de la aplicación legada, se detallarán los conceptos necesarios para entender el funcionamiento de esta. Además, se explica la notación que utiliza el editor de grafos para representar los escenarios de interacción.

2.1. Conceptos Generales

- *Escenario de interacción:* Modelo que describe de qué forma se comunican e interactúan entre sí los distintos roles mientras realizan una cierta actividad. En el contexto de la aplicación, los escenarios de interacción se representan mediante grafos, en donde los nodos simbolizan los roles, y las aristas representan las relaciones entre ellos.
- *Sistema colaborativo:* Aplicación utilizada por un grupo de individuos, los cuales realizan una tarea (o persiguen un objetivo) en común. La realización de dicha tarea genera interacciones entre los usuarios del sistema.
- *Proceso colaborativo:* Es el conjunto de tareas interrelacionadas que se ejecutan para lograr un propósito determinado. En el contexto de la herramienta, un proceso colaborativo involucra a todas las actividades que diversos roles realizan para lograr un objetivo en común.
- *Fase:* Son etapas secuenciales en que divide a un proceso. Un proceso puede contener una fase (si es que no existe secuencialidad), o más de una.
- *Ámbito:* Representa a un área de trabajo, y usualmente está asociada a una fase. Un proceso colaborativo puede tener varios ámbitos o áreas de trabajo ejecutándose en paralelo. Cabe destacar que existen dos tipos de ámbitos: el primero es individual para cada usuario y no se representa mediante un escenario de interacción (grafo), mientras que el segundo es colaborativo, y utiliza un grafo para modelar las interacciones que deban soportarse en la aplicación que está siendo modelada.

2.2. Notación Visual Utilizada en la Aplicación

- *Roles:* Los roles representan los perfiles de los usuarios (actores) que participan en un proceso colaborativo. Existen tres tipos de roles, los cuales se indican a continuación:

	*Usuario humano:* Persona que usa los servicios del sistema y cumple un rol definido durante el proceso colaborativo.
	*Servicio*: Componente de software autónomo que se comporta de acuerdo con una lista predeterminada de acciones.
	*Repositorio*: Componente de software pasivo que solo almacena datos y produce respuestas a preguntas que fueron solicitadas por humanos o servicios.

- *Enlaces*: Estos representan las necesidades y capacidad de interacción entre dos roles. Existen dos tipos de enlaces:

—	*Interacción:* Representa el hecho de que un participante (rol) requiere comunicación con otro. Una interacción que va desde un nodo (rol) a sí mismo, representa una interacción entre actores que cumplen el mismo rol.
--▷	*Herencia:* Representa el concepto de grupo o relación "es un/una", en donde el nodo del que sale la flecha puede cumplir el rol asociado al nodo al que apunta la flecha. Ambos roles deben ser del mismo tipo.

- *Número de participantes:* Cada nodo tiene una etiqueta que determina el número de participantes que pueden cumplir dicho rol:

0..*	*Cero o más*: El número de instancias de este rol que pueden estar presentes al momento de llevar a cabo el proceso puede ser cero o más. Los roles que permiten una cardinalidad de cero son roles opcionales.
0..1	*Cero o uno:* El número de instancias de este rol para llevar a cabo el proceso puede ser cero o uno (el rol es opcional).
1	*Exactamente uno:* El número de instancias requeridas de este rol, para llevar a cabo el proceso, debe ser exactamente uno (el rol es obligatorio).
1..*	*Uno o más:* El número de instancias de este rol para llevar a cabo el proceso debe ser uno o más (el rol es obligatorio).
?	*Indefinido:* El número de instancias de este rol es desconocido o aún no se ha definido.

- *Tipo de usuario:* Cada rol también está etiquetado con el tipo de usuario, el cual corresponde con el uso que cada rol hará de la aplicación que se está modelando. Aquí la información se presenta para un actor humano, pero aplica también para servicios y repositorios:

	*Usuario interno:* Los actores de este rol harán uso o pertenecen a la aplicación que se está modelando.

	*Usuario externo:* Los actores de este rol no usarán (o no pertenecen) a la aplicación que se está modelando. Actores humanos pueden usar una aplicación externa, o servicios y repositorios pueden ser externos a la aplicación.
	*Usuario interno/externo:* Los actores de este rol pueden usar (o pueden pertenecer a) la aplicación en desarrollo, una externa, o una mezcla de ambas para interactuar.

- *Roles abstractos:* En las relaciones del tipo “es un/una”, pueden existir roles que están visualmente etiquetados con “*abstract*” arriba de su nombre. Por defecto, un rol es considerado “concreto”. Cuando se especifica que un rol es abstracto en una relación de herencia, significa que el rol no es instanciable. Es decir, no tendrá actores que ejecuten sus acciones, y sólo podrán realizarlas los usuarios con los roles que heredan de él.

3. Análisis del Sistema Legado

La herramienta de diseño de escenarios de interacciones, particularmente la versión que se tomó como base, cuenta con un *frontend* y un *backend*. En las siguientes secciones se describen ambos componentes.

3.1. Análisis del Front-End del Sistema

El *frontend* del sistema legado se encuentra implementado en el lenguaje JavaScript, junto a la librería Vue.js. Éste cuenta con tres vistas principales: procesos, escenarios de interacción y las interfaces de los prototipos. Una de las principales deficiencias generales que se pueden encontrar aquí, es que no existe un sistema de navegación para cambiar entre las vistas, por lo que el usuario tiene que utilizar los botones “atrás” y “adelante” del navegador, los cuales pueden no estar disponibles (porque se ingresó directamente a una vista) o pueden no funcionar como se espera. A continuación, se describe cada una de las vistas.

3.1.1. Vista de Procesos

Esta vista permite crear, modificar y eliminar procesos (Figura 2). Cada proceso cuenta con una o más fases (o etapas), y ámbitos que son actividades que se realizan en paralelo dentro de una fase. Para cada fase y ámbito se debe especificar el escenario de interacción que ocurre como parte del proceso colaborativo. Una vez seleccionado un proceso, se puede ver, crear, modificar y eliminar sus ámbitos y fases dentro de una subvista muy similar.

Un problema importante tanto de la vista de procesos, como de la subvista de fases es que presentan muy poca información por sí solas, además de ser redundantes. Adicionalmente, los modales que se despliegan para crear o modificar procesos, ámbitos o fases pueden resultar confusos para el usuario.

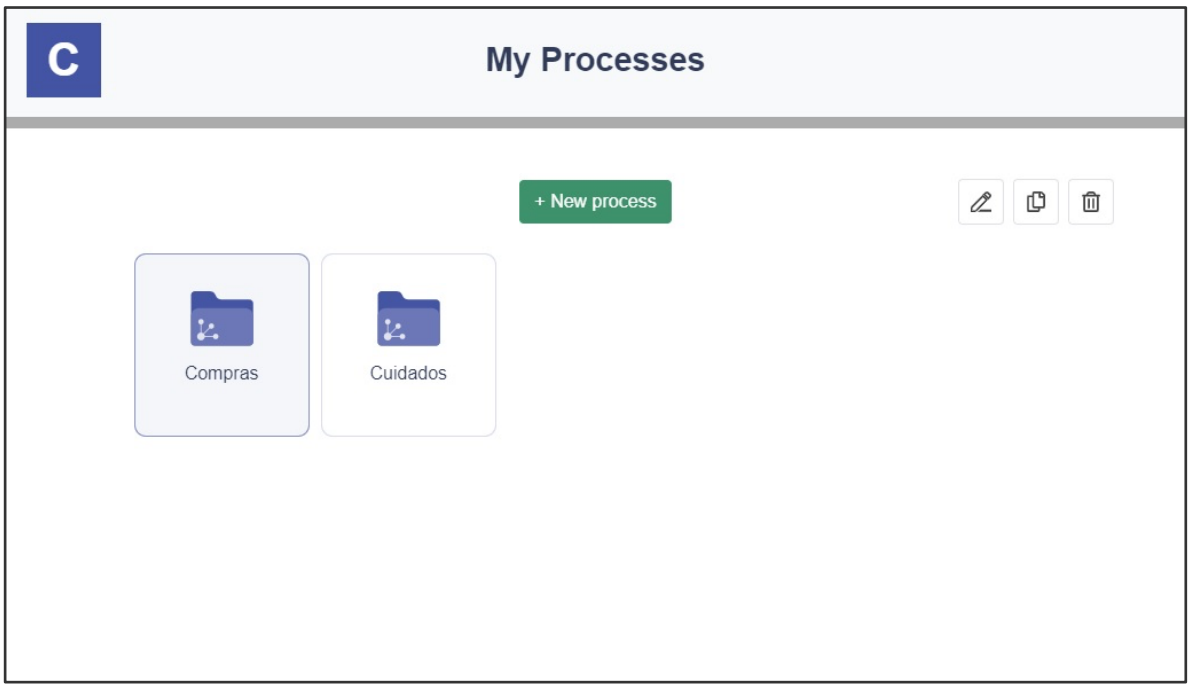

Figura 2. Vista de procesos modelados.

3.1.2. Vista de Escenarios de Interacción

Los escenarios de interacción son representados a través de grafos que vinculan a los roles (nodos) participantes en el proceso (Figura 3). Por lo tanto, esta vista presenta un editor de grafos, con opciones para añadir roles e interacciones entre estos. También permite especificar el tipo de interacciones entre nodos. Existe simbología en cada rol, lo cual permite modelar la cantidad de roles de cada tipo, además de si un rol interactúa con otros dentro de la aplicación que se está modelando, o por vía externa.

En la parte superior de la Figura 3 se pueden observar las diversas herramientas que permiten modelar el grafo. El primer grupo permite agregar roles e interacciones mediante *clicks* del *mouse,* mientras que en el segundo se cuenta con opciones para hacer zoom o deshacer y rehacer cambios. Además, se cuenta con opciones de exportación a imágenes o una lista de requisitos para cada rol.

Cabe destacar que algunas de las interacciones presentes en esta vista pueden resultar poco intuitivas. Por ejemplo, para agregar un rol, se hace necesario hacer *click* sobre el ícono correspondiente (rectángulos blancos en la barra superior), sin embargo, al hacer esto, no ocurre nada, ya que además es necesario volver a hacer *click* sobre un lugar del editor donde se desea colocar el rol.

La vista también cuenta con varios errores. Por ejemplo, el grafo desaparece al realizar algunas acciones y es necesario recargar la página para arreglar el problema. También permite ciclos de herencia que provocan que la aplicación quede estancada.

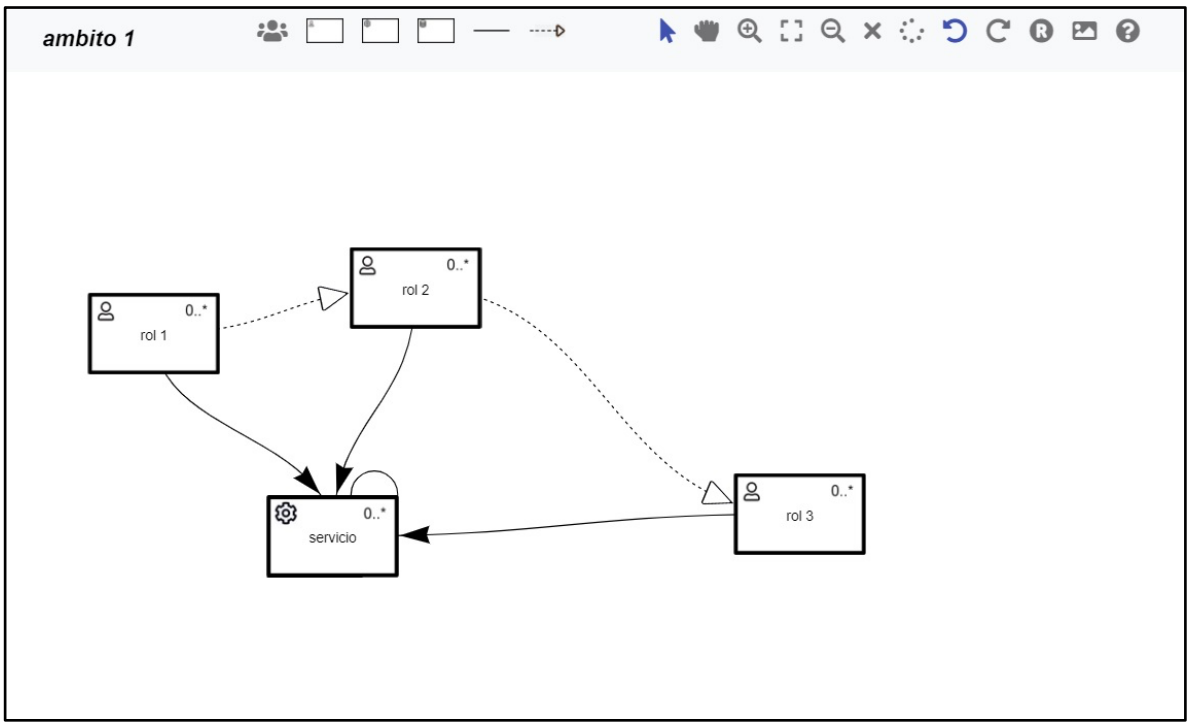

Figura 3. Diagramador de escenarios de interacción.

A partir de las definiciones previas (es decir, procesos y escenarios), la herramienta genera diversas interfaces de usuario de la aplicación a desarrollar (Figura 4).

3.1.3. Interfaces de Usuario de la Aplicación

Tanto los grafos de interacciones entre roles, así como las interfaces generadas a partir de ellos, entregan una representación visual que facilita la validación de la funcionalidad a incluir en la aplicación móvil. En consecuencia, esto ayuda a lograr acuerdos entre el cliente y el desarrollador respecto a la funcionalidad a embeber en el software. Cabe destacar que esta vista también permite editar las interacciones entre los roles mediante una lista de opciones por cada rol. Estos cambios se ven inmediatamente reflejados en el *mockup* del teléfono móvil.

Los *mockups* son distintos para cada rol dependiendo de los servicios seleccionados en sus interacciones, pero estos roles deben ser del tipo humano. Los servicios se presentan en forma de matriz para las interacciones que ya están agregadas en la vista del editor de interacciones. Esto hace que sea engorroso el querer añadir una nueva interacción y luego poder revisar el *mockup* inmediatamente.

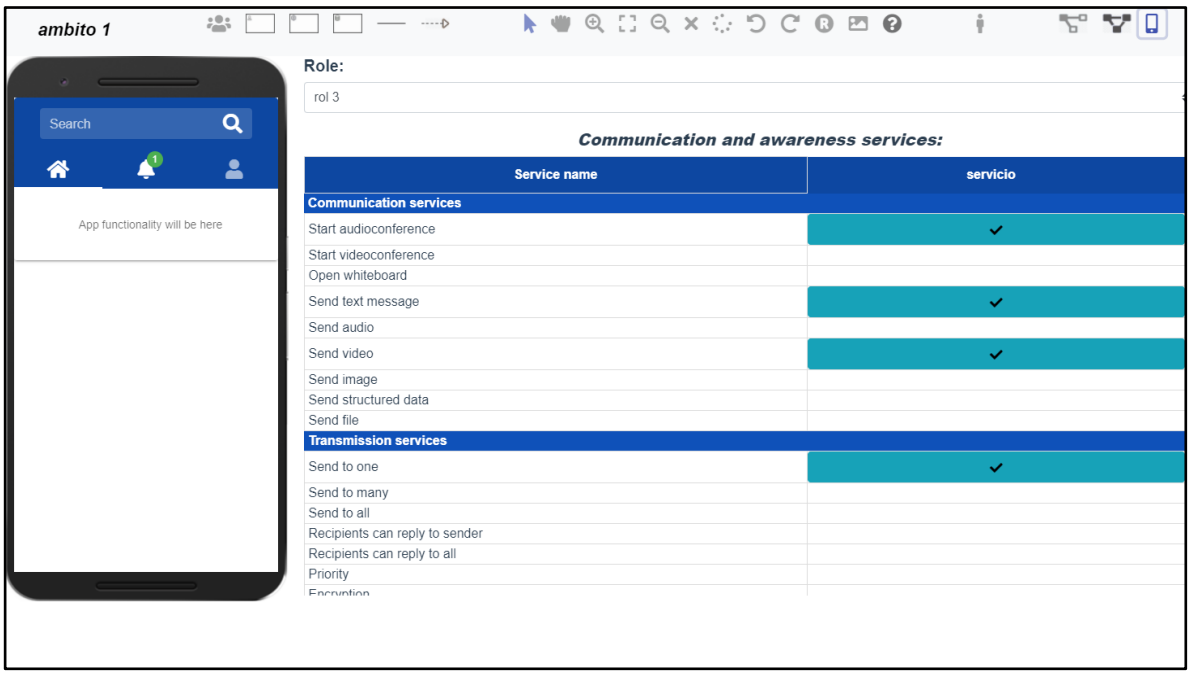

Figura 4. Interfaces de usuario de la aplicación móvil.

3.2. Backend del Sistema

A continuación, se presentan los principales componentes del *backend* del sistema legado, los cuales están también reportados en [7].

3.2.1. Arquitectura Física

La arquitectura física del sistema cuenta con 3 capas, tal como lo muestra la Figura 5. La capa 1 corresponde al navegador (*internet browser*) que se utiliza para acceder a la aplicación web. Luego el protocolo HTTP hace de intermediario entre la capa 1 y la capa 2.

Por su parte, la capa 2 consta del servidor web, encargado de hospedar la aplicación, al componente "CIMoN Manager" y a los servicios que éste último entrega. Todo lo anterior es deployado (desplegado) sobre el servidor Orión del Departamento de Ciencias de la Computación de la Universidad de Chile. Finalmente, la capa 3 consta del modelo de datos que se encuentra del lado del servidor, y que utiliza una base de datos PostgreSQL.

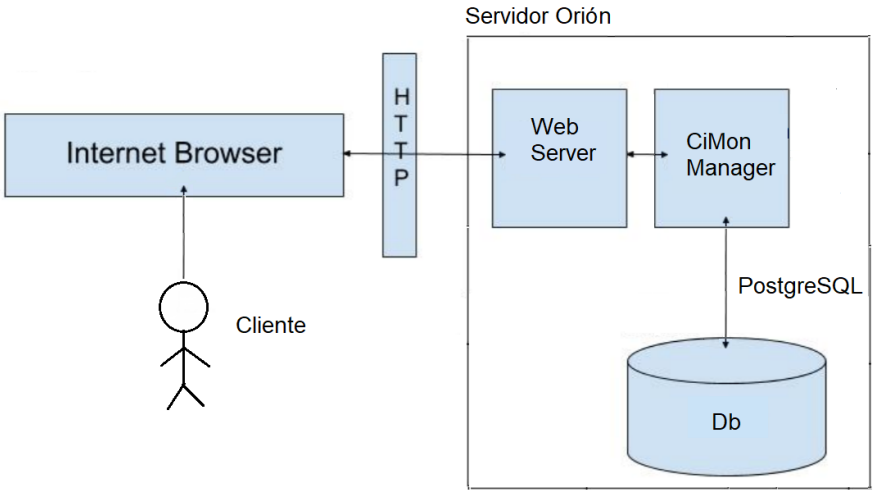

Figura 5. Arquitectura física del sistema legado (obtenido de [7])

3.2.2 Arquitectura Lógica

La arquitectura lógica del sistema consta de siete módulos, separados en 3 capas, tal como se muestra en la Figura 6. En la *capa de presentación* se ubica el Gestor de Procesos y el Modelador de Escenarios; este último es la herramienta gráfica destinada a la edición y construcción de los grafos de interacción entre roles (perfiles de usuario).

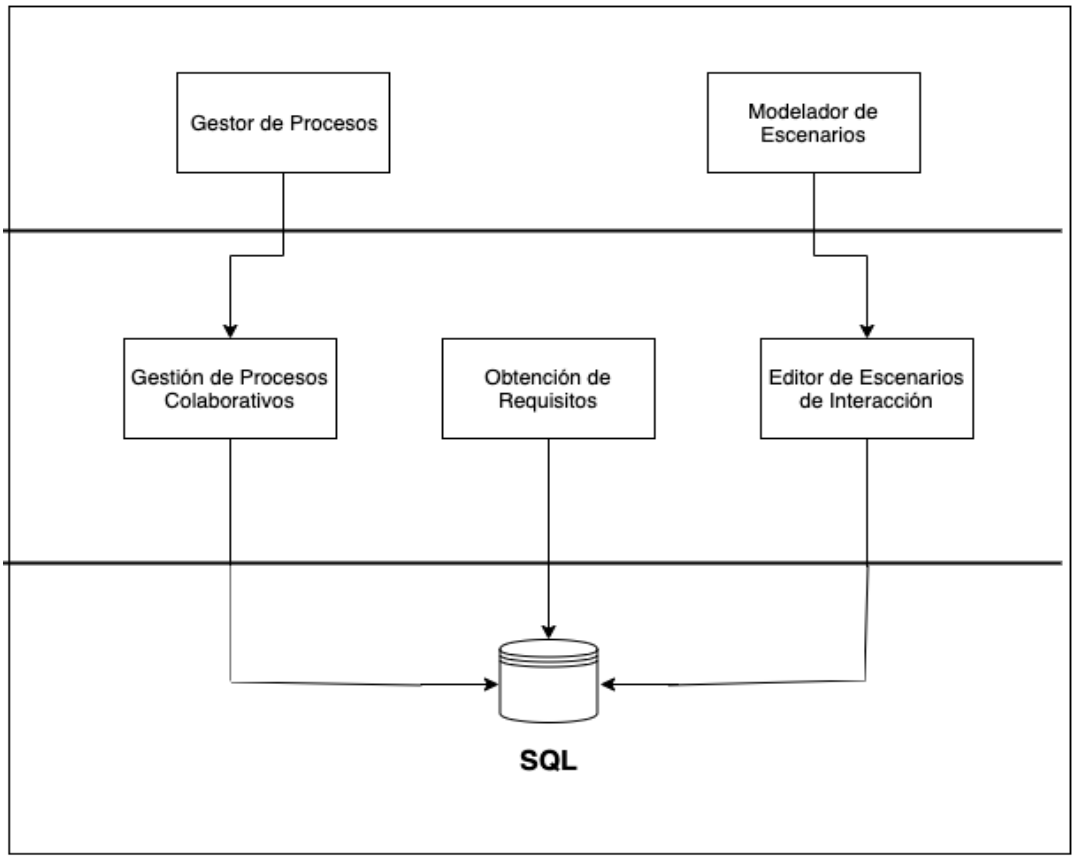

Figura 6. Arquitectura lógica del sistema legado (obtenido de [7])

En la *capa de lógica de negocio* se encuentran los servicios asociados a la Gestión de Procesos Colaborativos. Esta capa permite crear y editar procesos junto a sus componentes, y obtener requisitos funcionales a partir del grafo construido. El Editor de Escenarios de Interacción es quien se conecta con la aplicación permitiendo editar los grafos.

Finalmente, la *capa de datos* contiene la base de datos (en un modelo de datos relacional) encargada de almacenar dos cosas: 1) la información de los procesos, y 2) los archivos JSON que representan ámbitos dentro de un proceso, pero como un modelo de datos no relacional, permitiendo almacenar las relaciones dentro de cada grafo.

3.2.3 Modelo de Datos del Sistema Legado

Como se mencionó antes, la aplicación legada utiliza un modelo de datos relacional, y seis tablas, las cuales se describen a continuación:

- *Procesos:* Esta tabla almacena la información de los procesos de negocio a modelar. Cada proceso cuenta con un identificador único y un nombre.

- *Fases:* Cada proceso puede contar con múltiples fases (o etapas), las cuales poseen un identificador único, un nombre y el identificador del proceso al que están vinculadas.

- *Ámbitos (fields*): Los ámbitos corresponden a una actividad en una línea de trabajo específica. Esta actividad ocurre de forma paralela a otros ámbitos de la misma fase. Los ámbitos poseen un identificador único, un nombre, el tipo de ámbito (individual o colaborativo), el identificador del grafo asociado, un campo *booleano* para determinar si el grafo asociado está validado, y un campo *booleano* para determinar si los requisitos se encuentran generados.

- *Grafo:* Los grafos están asociados a ámbitos, y poseen un identificador y un campo JSON que almacena los nodos y las relaciones entre estos, además de diversas características y propiedades asociadas.

- *Requerimientos*: Estos corresponden a los requisitos funcionales de interacción (o servicios de interacción), potencialmente derivables de un grafo. Estos requisitos cuentan con un identificador único, un nombre y un aspecto a considerar. Como es una relación *Many-To-Many* respecto a los ámbitos (un ámbito puede tener muchos requisitos, y un requisito puede estar asociado a muchos ámbitos), el modelo considera una tabla intermedia *Field Reqs* para cubrir esta necesidad.

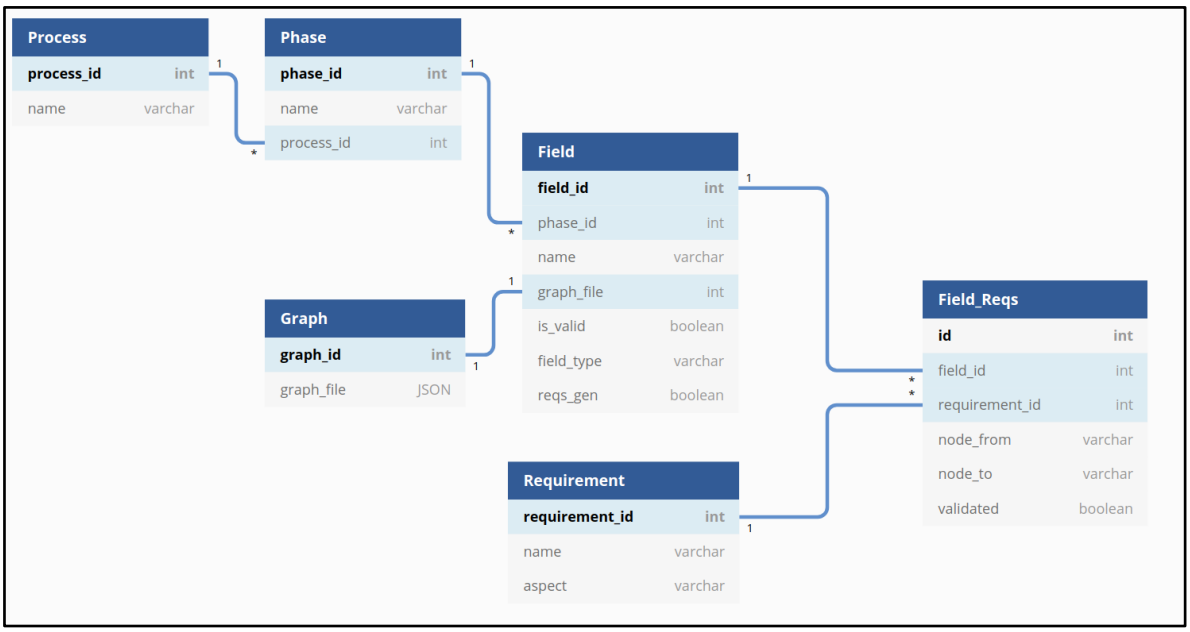

Figura 7. Modelo de datos del sistema legado (obtenido de [7])

3.3. Problemas generales de la herramienta

Si bien la herramienta posee la funcionalidad suficiente para ser útil, posee diversos problemas que dificultan su uso, y, también, su extensibilidad y mantenimiento a futuro:

1. Se pierde contexto al cambiar de una vista a otra, debido a que cada vista es muy especializada y solo presenta información específica.
2. El modelo de datos guarda mucha información redundante o innecesaria, sobre todo en el campo no relacional *graph_file*.
3. El código de los componentes de la herramienta presenta archivos con lógica muy acoplada y difícil de reutilizar, además de tener archivos muy largos (más de 5000 líneas para algunos componentes).
4. El diseño gráfico general y las interacciones de usuario no siguen patrones modernos y pueden resultar confusas o poco familiares para el usuario.
5. Hacen falta características como un sistema de manejo de usuarios (cualquier usuario con acceso a la herramienta puede editar los archivos).

Las soluciones ofrecidas para estos problemas serán explicadas en los siguientes capítulos.

4. Concepción de la Solución

A continuación, se presentan las principales decisiones de diseño y características relevantes de la nueva versión de la herramienta, así como las tecnologías usadas en su implementación.

4.1. Tecnologías Utilizadas

Las herramientas utilizadas en este trabajo se pueden dividir en dos grandes grupos: las utilizadas en el *frontend* y las utilizadas en el *backend*. Cabe destacar que la gran mayoría del trabajo se enfoca en el *frontend,* que es el que contiene la mayor parte de la lógica, siendo el *backend* usado en su mayoría solo para almacenar datos y entregar *endpoints* para comunicarse con los clientes.

En el *frontend* se pueden destacar las siguientes herramientas:

- *TypeScript*: Lenguaje de programación y extensión del lenguaje JavaScript, al que agrega principalmente tipado estático y otras características menores. Se escoge este lenguaje por los beneficios que entrega en prevención de errores e integración con editores de texto superior a JavaScript.
- *React*: Librería para crear interfaces gráficas mediante componentes. Se escogió debido a su buena integración con TypeScript y su gran ecosistema, que cuenta con herramientas para muchos casos específicos.
- *React-query*: Librería para el manejo de la sincronización del estado del cliente y el servidor. Además, permite de forma fácil el manejo de errores y de estados de carga.
- *Zustand*: Librería para el manejo de estado global en el cliente.

Mientras que en el *backend* se utilizan:

- *Python*: Lenguaje de programación dinámico y multipropósito.
- *Django*: Framework para desarrollar aplicaciones web de forma rápida y robusta. Se escoge principalmente porque ya se tenían conocimientos previos en la herramienta.
- *Django REST Framework*: Extensión de Django que permite crear APIs REST que hace que cualquier cliente pueda comunicarse con el servidor.
- *PostgreSQL*: Base de datos relacional, que además tiene soporte robusto para manejar campos no relacionales a través de archivos tipo JSON.

Además de las herramientas anteriores, se utilizan muchas otras que cumplen funciones menores, tanto en el *frontend* como en el *backend*.

4.2. Requisitos para la Reimplementación de la Herramienta

A continuación, se presenta una lista resumida de los requisitos de software que se abordan durante el desarrollo de esta memoria. Particularmente, se separan las funcionalidades a crear en el frontend y backend:

1. Implementación del *backend*:
    a. Creación de modelos de datos.
    b. Incorporar sistema de manejo de usuarios, incluyendo autenticación y autorización.
    c. Habilitar *endpoints* que permitan operaciones sobre los grafos.
2. Implementación del *frontend*:
    a. Implementar los componentes del sistema legado, y sus funcionalidades.
    b. Incorporar el sistema de manejo de usuarios.
    c. Crear la vista de *login* y registro.
    d. Agregar las interacciones con la interfaz de usuario a través de operaciones *drag-and-drop*.
    e. Implementar la generación de mockups de la aplicación móvil que se busca prototipar.
    f. Implementar el tipo de interacción para los mockups.
    g. Implementar las características de exportación (a pdf, imágenes, etc.) de datos.
    h. Implementar validación automática de los grafos.

4.3. Arquitectura Lógica de la Solución

La aplicación sigue una arquitectura clásica con un modelo de tres capas. A continuación, se detalla cada una de éstas:

- *Capa de Presentación:* Contiene todo el diseño de la interfaz además de la lógica necesaria para representar a los elementos visualmente. Gran parte del código del *frontend* se encuentra en esta capa. Dentro de esta se encuentran tres vistas principales:
    - Vista de *login* y registro: lugar en donde un usuario se identifica, o puede crear una cuenta nueva en caso de no poseerla.
    - Vista de gestión de procesos: Al estar autenticado, un usuario puede acceder a la lista de procesos que ha creado, o crear nuevos procesos. Además de esto, en esta vista se pueden manipular las fases y los ámbitos de cada proceso.
    - Vista de editor de escenarios de interacción: Presenta el editor de grafos y diversas opciones relacionadas a éste.
- *Capa de lógica de negocio:* Aquí se encuentra la lógica necesaria para poder hacer operaciones en las vistas anteriormente mencionadas. Esta capa también es mayormente responsabilidad del *frontend,* aunque contiene componentes

importantes en el *backend* de la aplicación. Los módulos principales dentro de ésta son la gestión de usuario (relacionada al proceso de autenticación y autorización), la gestión de los procesos, y el manejo y lógica detrás del editor de escenarios de interacción.

- *Capa de datos:* Esta capa almacena la información necesaria de cada proceso, además de los datos de usuario. Es de responsabilidad única del *backend* de la aplicación. Es importante destacar que el manejo de datos se encuentra abstraído por Django, y no se accede a PostgreSQL directamente.

4.4. Modelo de Datos

El modelo de datos se ha simplificado con respecto a la aplicación de legado, el cual tenía información redundante o innecesaria. En la Figura 8 se puede apreciar este nuevo modelo.

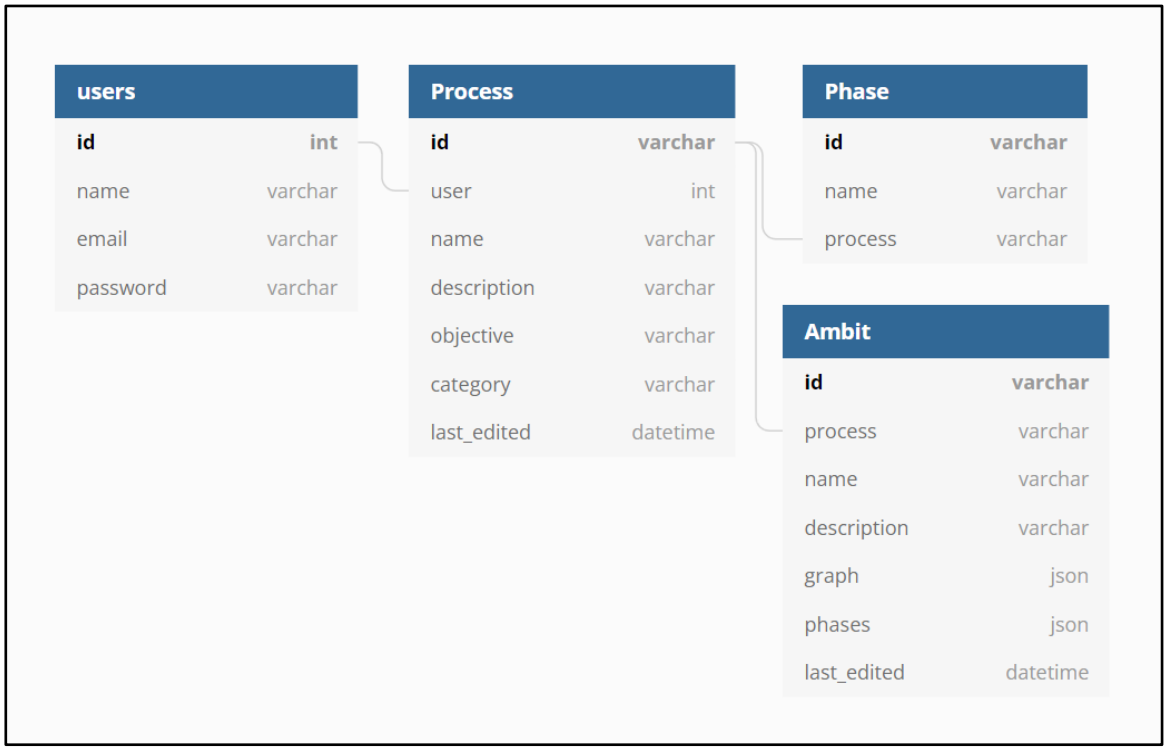

Figura 8. Modelo de datos de la aplicación.

El modelo cuenta con cuatro entidades que se explican a continuación:

- *Usuario:* Modelo de datos para identificar a cada usuario. Extiende el modelo de usuarios de Django. En la imagen solo se incluyen los campos que son relevantes para la aplicación.
- *Proceso:* Incluye toda la información para modelar un proceso. Se agregan campos como descripción, objetivo y categoría, permitiendo tener información relevante durante el proceso de modelado.
- *Fase:* Incluye a las fases dentro de un proceso. Además, cuenta con la restricción

de que el nombre de cada fase debe ser único dentro de un proceso.

- *Ámbito:* Contiene toda la información relevante a un ámbito. Se agrega una descripción en comparación al modelo anterior. Además, el modelo se relaciona directamente al proceso y no a una fase, debido a que un ámbito puede pertenecer a varias fases distintas. Esto se modelaba de forma muy redundante con la estructura anterior. Por su parte, un ámbito cuenta con dos campos de tipo JSON, uno para almacenar la lista de IDs de las fases a la que pertenece, y otro para almacenar el grafo correspondiente.

El modelo no relacional del grafo se muestra en la Figura 9.

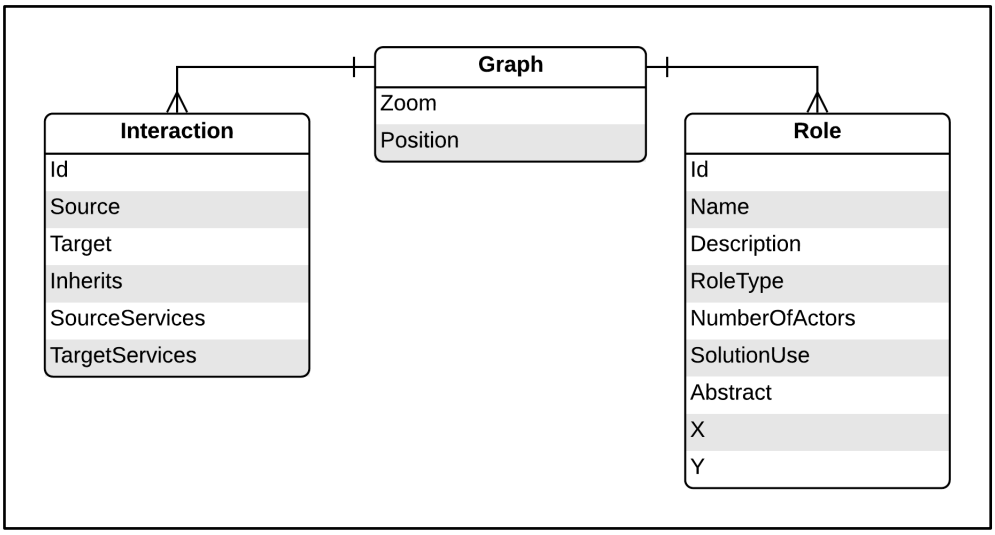

Figura 9. Modelo de datos de los grafos.

Cada grafo cuenta con atributos posicionales que permiten guardar el estado en el que se encuentra en el editor. Además de esto, cuenta con una lista de roles y una lista de interacciones que se describen a continuación:

- *Roles*: Estas entidades de datos cuentan con un ID, un nombre y una descripción. Los siguientes cuatro atributos forman parte de la notación establecida en la técnica de modelado: tipo de rol, número de actores, uso de solución, y abstracto. Finalmente, los roles cuentan con coordenadas para establecer su posición relativa en el editor.

- *Interacciones*: Las interacciones sólo se identifican mediante un ID. Los atributos de fuente y objetivo indican los IDs de los nodos relacionados (participantes en la interacción). El siguiente atributo indica si es una relación de herencia o no. Finalmente, los últimos dos atributos representan listas con las posibles acciones que puede realizar un rol en esta interacción, como, por ejemplo, llamar por teléfono, o enviar archivos adjuntos.

4.5. Diseño de API REST

Debido a que el *backend* y el *frontend* de la aplicación son componentes separados, se hace necesario establecer una vía de comunicación entre ambos. Para esto se utiliza una API REST [8] implementada con la ayuda de Django REST Framework.

Un aspecto importante introducido en la nueva versión de la aplicación es un sistema de manejo de usuarios. Para controlar el sistema de autenticación y autorización ligados a este mediante la API REST, se utilizó un sistema de *JSON web tokens* (JWT) [9], en donde un usuario obtiene dos *tokens* (strings de texto "firmados" por el servidor). Uno de ellos cumple la función de que el usuario se identifique y se envía en cada petición desde el cliente al servidor, y el otro, llamado de "*refresco*" permite obtener nuevos *tokens* en caso de que éstos expiren.

Los puntos que dispone el servidor para la comunicación (*endpoints*) son los siguientes:

- *register/:* Sólo acepta peticiones del tipo HTTP POST con datos de usuarios nuevos.
- *token/*: Entrega los *tokens* necesarios a un usuario que acaba de iniciar sesión con su correo electrónico y contraseña. Sólo acepta peticiones HTTP POST.
- *token/refresh/*: Entrega nuevos *tokens* si se envía un *token* de refresco válido. Sólo acepta peticiones HTTP POST.
- *processses/*: Son los *endpoints* para obtener la lista de procesos de un usuario, y también crear un nuevo proceso. Acepta peticiones HTTP GET y POST. Se puede añadir la ID de un proceso ya creado (de la forma *processes/:id/*) para obtener los datos de un proceso en específico, además de modificarlo o eliminarlo. En este caso también acepta peticiones tipo HTTP DELETE y PATCH.
- *phases/:* Funciona de forma análoga a endpoint de procesos, pero con datos de las fases.
- *ambits/:* Funciona de forma análoga a endpoint de procesos, pero con datos de los ámbitos. Además, cuenta con *endpoints* adicionales:
    - *ambits/roles/:* Permite crear, editar y eliminar roles.
    - *ambits/interactions/:* Permite crear, editar y eliminar interacciones.

5. Rediseño de las Interfaces de Usuario

El proceso de rediseño de las interfaces se realiza de forma iterativa, y se incluyen en este las opiniones y sugerencias de usuarios en diversos aspectos, tanto en el proceso de diseño como de desarrollo del nuevo sistema. Para comenzar, se estudia el funcionamiento de diversas aplicaciones de diagramación, como, por ejemplo, *Miró*[2], *Lucidcharts*[3], *Diagrams.net*[4] y *Canva*[5]. A partir de estas aplicaciones se obtienen algunas ideas en común, que sirven de base para representar las interacciones en el editor de grafos; es decir, en el editor de escenarios de interacción entre roles participantes en un proceso de colaboración.

El diseño de las demás vistas de la aplicación se basa en la revisión de interacciones comunes entre los usuarios y las interfaces gráficas de aplicaciones modernas. Para llevar a cabo este diseño también se utiliza el conocimiento que el memorista ya tenía sobre el diseño de interfaces gráficas. Además, se consideraron principios de diseño ampliamente aceptados como psicología de *Gestalt* [10], y también se priorizan los tipos de interacciones a las que los usuarios ya podrían estar acostumbrados, como el gesto de *drag and drop*. También se tiene especial cuidado en que las interfaces gráficas sean consistentes, y que cada elemento tenga un significado claro para el usuario.

Para lograr lo anterior, se hace necesario establecer un lenguaje de diseño desde el comienzo, y para ello se utilizan principalmente prototipos de alta fidelidad, que se validan en forma iterativa con los usuarios. Se introducen, además, cambios a estos durante el desarrollo, de acuerdo con el *feedback* entregado por los usuarios, aunque se mantuvo en gran parte el aspecto estético y funcional planteado inicialmente pues resulta cómodo e intuitivo para estas personas.

En cuanto a la experiencia de usuario, se reduce el número de vistas disponibles en la aplicación, debido principalmente a que algunas vistas, como por ejemplo las de "procesos" y "fases", contienen muy poca información por si solas en la aplicación de legado. Esto da lugar al rediseño de las vistas, que ahora muestran más información y esta es más relevante.

Otro aspecto importante por destacar es que se introduce el concepto de "actualizaciones optimistas", en donde el usuario no siente el tiempo de latencia entre su dispositivo y el servidor. Un ejemplo de esto es que, en la aplicación de legado, al agregar un rol al grafo de interacciones (en la vista del editor de interacciones), el usuario debe esperar un tiempo indeterminado antes de observar los cambios en su pantalla, el cual puede ser alto en caso de tener una conexión inestable o de baja calidad entre el dispositivo cliente y el servidor. Con la nueva aplicación, los cambios se realizan instantáneamente. Luego el cliente realiza la petición al servidor, existiendo un efecto de *rollback* que deshace los cambios en caso de que se produzca un error, o que el diseñador decida deshacer la operación realizada.

También se realizan varios cambios menores, como, por ejemplo, que al abrir un modal se enfoque directamente la primera caja de texto o botón relevante, lo que le permite al

[2] https://miro.com/
[3] https://www.lucidchart.com/
[4] https://www.diagrams.net/
[5] https://www.canva.com/

usuario ahorrar movimientos del *mouse* y *clicks*. Además, se enfatiza mucho el aspecto estético de la herramienta, incluso entregando la opción de escoger una interfaz con colores oscuros. Esto permite que los usuarios se sientan más cómodos con la aplicación, y que les sea más placentero utilizarla.

5.1. Mapa de Navegación

A continuación, se presenta el mapa de navegación de la nueva herramienta, así como sus principales interfaces de usuario.

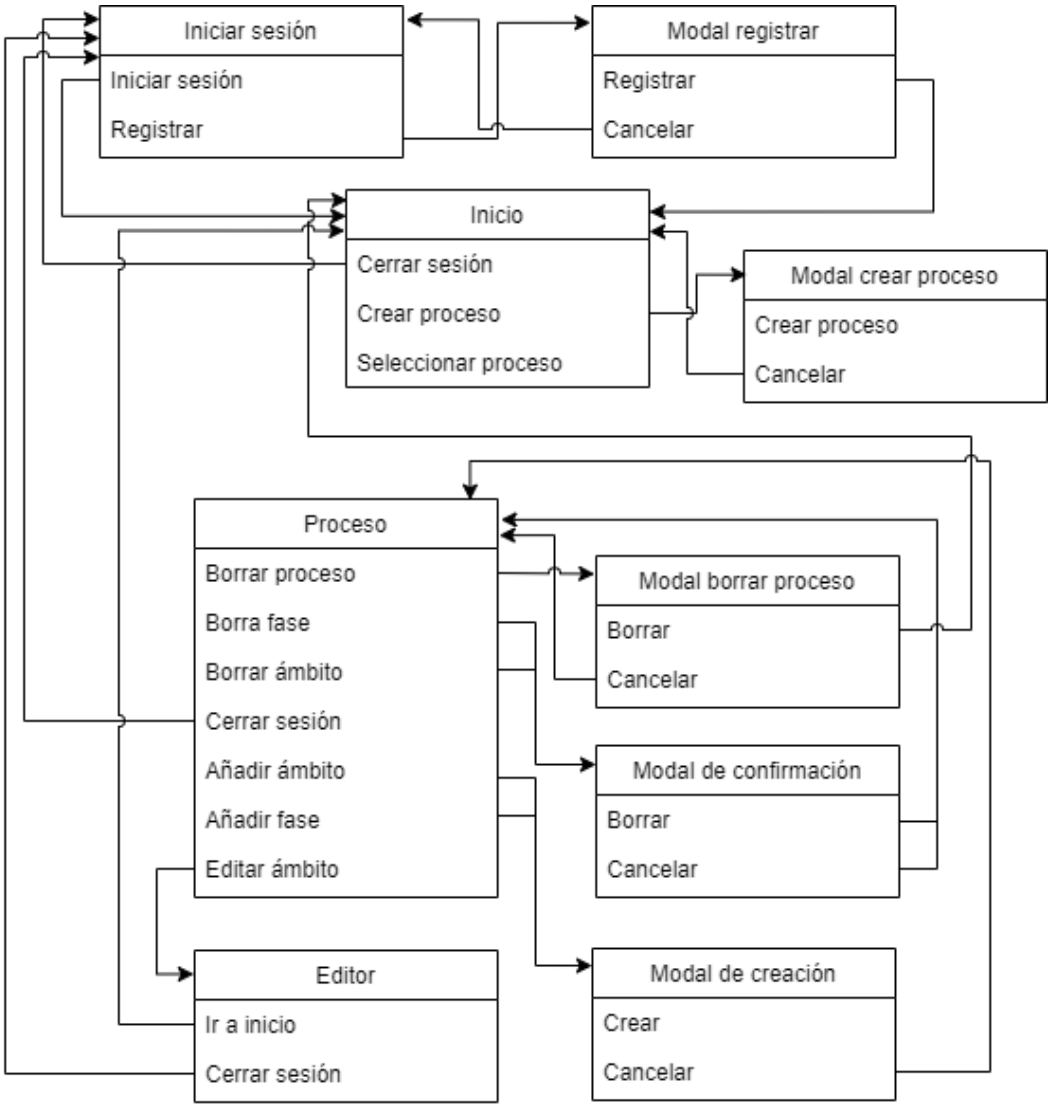

Figura 10. Mapa de navegación.

La aplicación se divide en tres vistas principales. En primer lugar, al ingresar a la página, un usuario (diseñador del sistema) observará una pantalla para iniciar sesión, y un botón para registrarse. Este último despliega un modal con un formulario para el registro en el

sistema. Una vez que el usuario se haya registrado o iniciado sesión, se redirigirá a la vista de inicio de la aplicación.

En la vista de inicio, al usuario de le presenta una barra lateral con la lista de procesos ha creado, además de una barra de búsqueda y un botón para crear un proceso nuevo. Este botón despliega un modal que permite ingresar los datos del nuevo proceso. Al seleccionar un proceso, el usuario pasa a la vista de proceso, que es una versión expandida de la vista de inicio. Acá se pueden ver y editar los detalles de cada proceso, además de desplegar modales para eliminar el proceso, agregar ámbitos y fases, o confirmar la eliminación de ámbitos y fases. Si un usuario hace *click* sobre un ámbito ya creado, pasará a la vista del editor de interacciones (o editor del grafo de interacciones).

La vista del editor de interacciones no presenta elementos de navegación además de la barra de navegación que también está presente en la vista de inicio y de proceso. Las acciones de creación son más simples, y la presencia de botones para deshacer y rehacer acciones permite que ya no se requieran modales para interactuar con la aplicación.

5.2. Vista de Inicio de Sesión

Esta vista es la más simple, y solo se compone de un formulario para iniciar sesión (Figura 11) y un botón para registrar una cuenta que abre un modal de registro (Figura 12). Ambos formularios cuentan con validación de errores para ayudar al usuario.

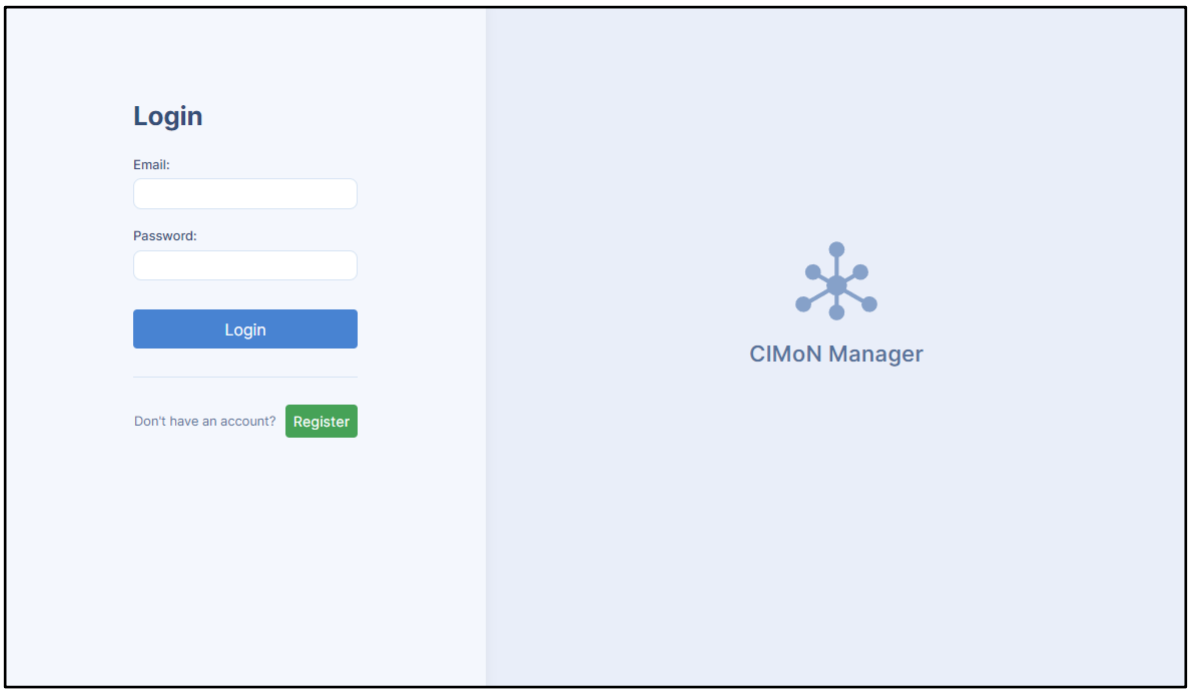

Figura 11. Vista de inicio de sesión.

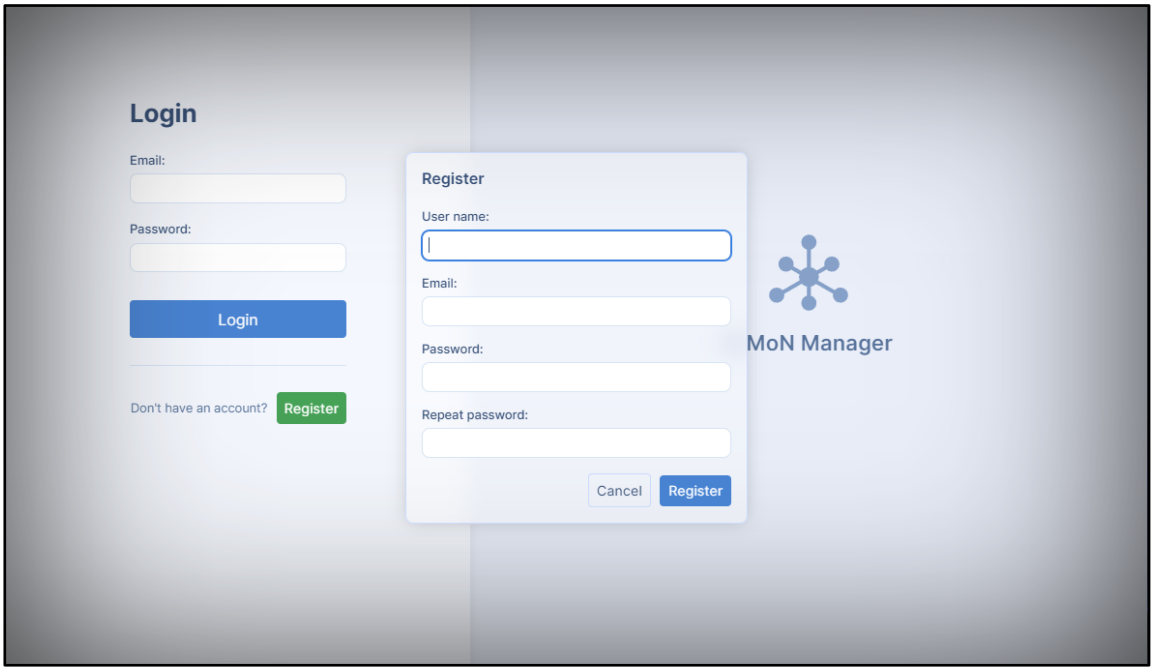

Figura 12. Modal de registro.

5.3. Nueva Vista de Procesos

La vista de procesos es la que ha tenido más cambios con respecto a la versión anterior de la aplicación. Ahora los procesos que ha creado un usuario se encuentran en una barra lateral, que además presenta una barra de búsqueda para permitir encontrar un proceso determinado. Al deslizar el *mouse* por encima de cada proceso se muestra un resumen de éste, sin necesidad de que el usuario tenga que hacer *click* en él.

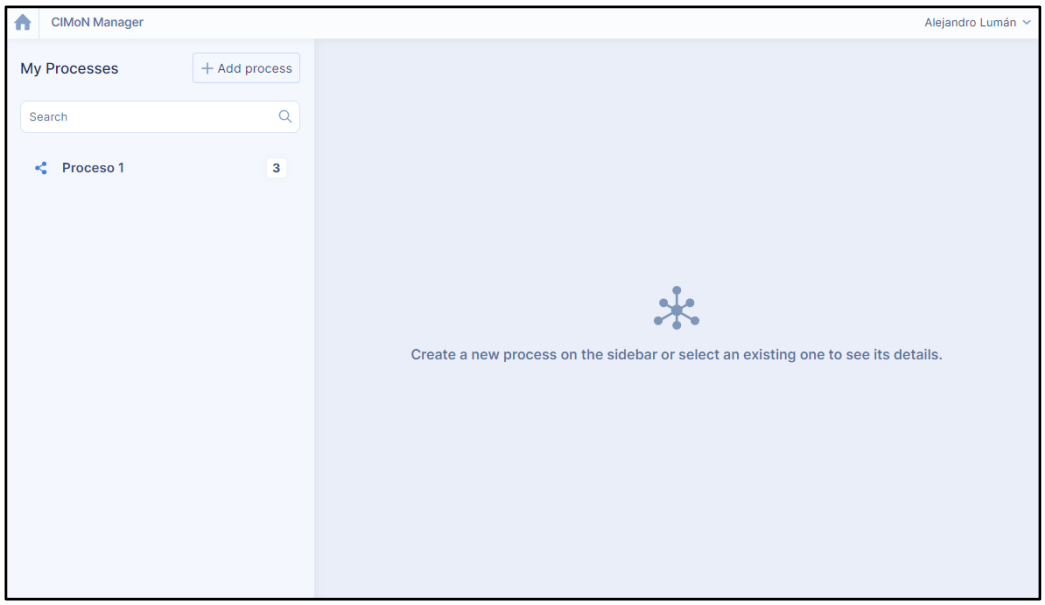

Figura 13. Nueva vista de procesos.

Esta vista cuenta con dos modos, uno sin un proceso seleccionado (Figura 13), y otro cuando un usuario ya ha hecho *click* en un proceso (Figura 14), es decir, ya ha seleccionado el proceso sobre el cual va a trabajar. Además de esto, se agrega una barra de navegación que también está presente en la vista de editor de interacciones. Esta barra permite ver el nombre de usuario que ha iniciado sesión. Si se hace *click* sobre éste, se muestra un menú que permite cambiar el tema (a modo oscuro o claro), cerrar sesión y mostrar información de la aplicación. El menú también cuenta con un ícono que permite volver a la vista de inicio.

Cuando el usuario ha seleccionado un proceso, se pueden ver los detalles de éste, donde la aplicación muestra un ícono que representa el tipo de aplicación a modelar, el nombre del proceso, su objetivo y una descripción de este, así como también un resumen de los roles que participan en éste.

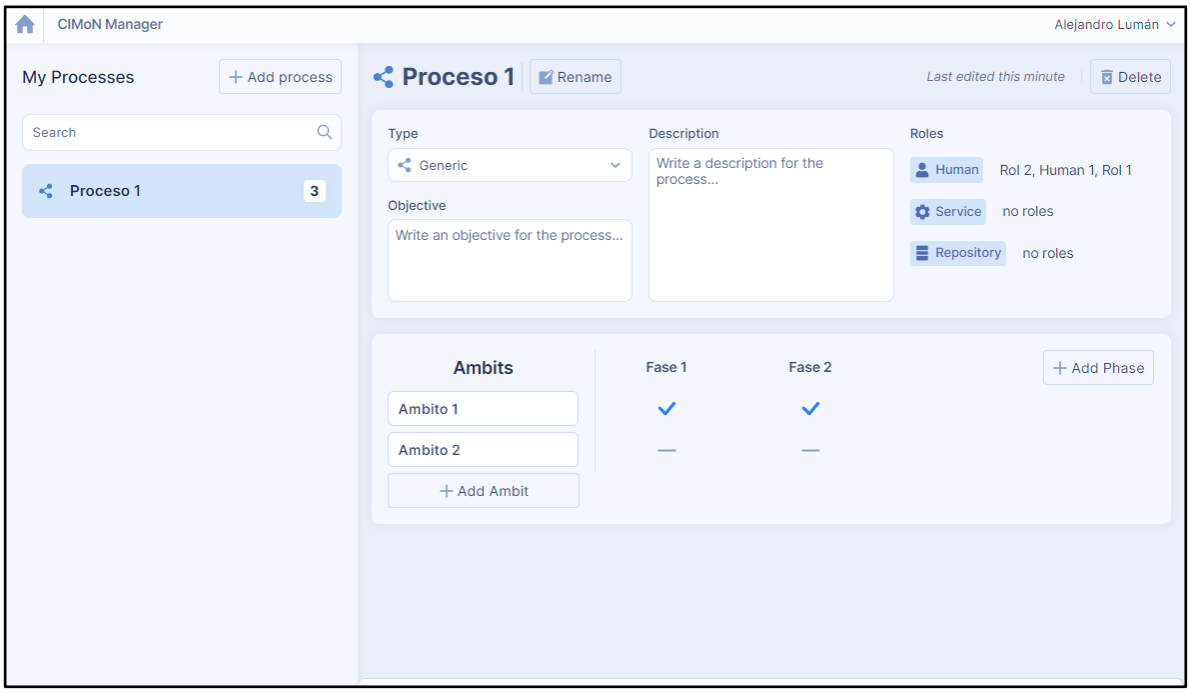

Figura 14. Vista del proceso seleccionado.

Finalmente, los ámbitos y fases dentro de un proceso se representan mediante una matriz, en donde la primera columna representa todos los ámbitos del proceso, y las siguientes representan cada fase que se ha agregado. El usuario puede hacer *click* dentro de cada celda para agregar o quitar un ámbito de una fase determinada.

Cuando un usuario desliza el *mouse* sobre el nombre de una fase o un ámbito, se muestra un ícono que permite editar su nombre o eliminarlo. Además, la manipulación de los ámbitos cuenta con capacidad para hacer *drag and drop,* lo que permite reordenarlos dentro de la matriz.

Una vez que el usuario hace *click* en un ámbito, se ingresa a la vista del editor de interacciones.

5.4. Nueva Vista de Editor de Escenarios de Interacción

El editor de interacciones (o editor de escenarios de interacción) cambia principalmente en la forma en la que el diseñador interactúa con él, más que en el orden de los elementos gráficos que se le presentan. En la Figura 15 se puede apreciar que se mantiene la barra lateral del diseño anterior. Sin embargo, en vez de contener menús de selección que requieren varios *clicks* para seleccionar una opción, estas opciones ahora se presentan de manera tal que permiten su manipulación directa.

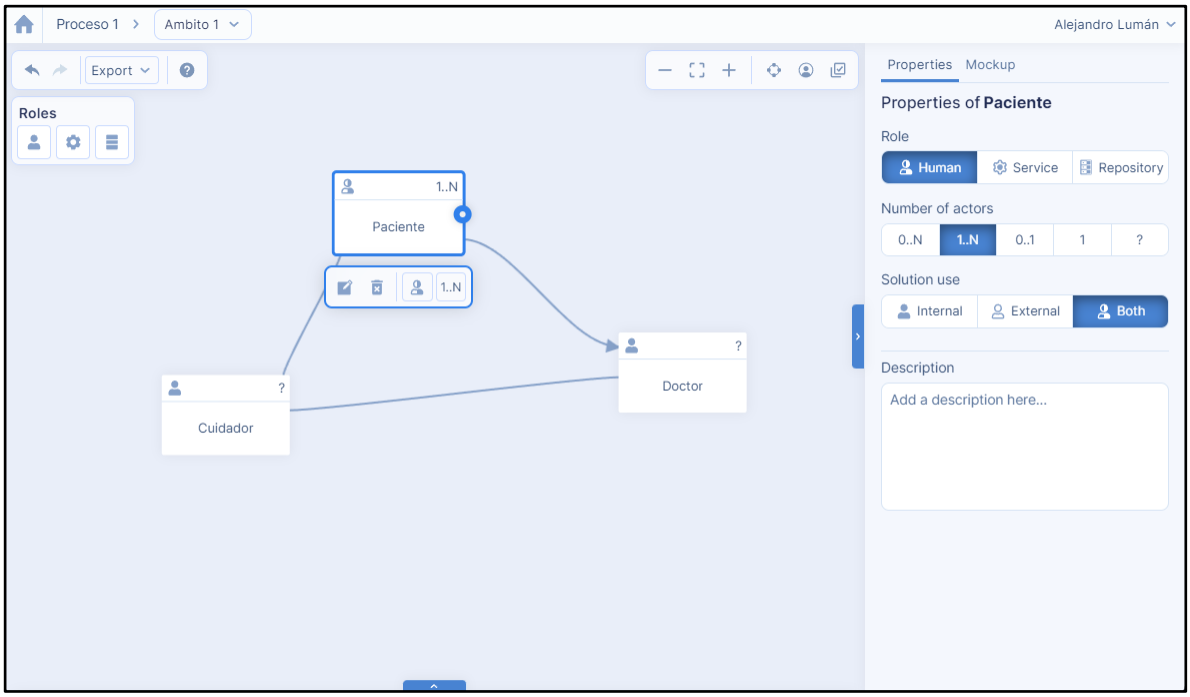

Figura 15. Nuevo editor de escenarios de interacción

La barra con íconos se divide en tres barras distintas, una para acciones relacionadas a la forma en que se ve el grafo (hacer *zoom*, realizar el ordenamiento automático de los nodos, etc.). Hay otra barra que contiene las acciones de deshacer/rehacer, exportación de los datos a diversos formatos y ayuda, y una tercera dedicada a la representación de roles. Esta última también muestra roles adicionales que se han creado en otros ámbitos, ofreciendo una forma rápida de reutilizarlos.

Para agregar un rol, ahora es necesario solo realizar un gesto de *drag and drop* desde la barra de roles. El rol se crea automáticamente con un nombre predefinido, por si el usuario desea postergar el ingreso de un nombre para dicho rol. Luego de creado el rol, y siempre que éste esté seleccionado, se muestra un menú flotante. En su parte inferior éste muestra opciones para editar el nombre del rol, eliminarlo, cambiar el tipo de este, o sus características. Las interacciones entre roles se agregan de forma similar a los roles, es decir, arrastrando un círculo de un rol a otro y generando de esa manera un vínculo entre ellos.

Por otra parte, se eliminan las opciones de "*selección*" y "*panning*" de la versión anterior de la aplicación, debido a que hacen muy engorrosa la manipulación del grafo. Ahora el usuario puede mover el grafo y los nodos libremente en todo momento.

Finalmente, en la barra de navegación, esta vista cuenta con opciones adicionales que se muestran al lado del ícono para volver al inicio. Una de ellas muestra el nombre del proceso actual, y lleva al usuario a la vista de este proceso, en caso de hacer *click* sobre ésta. La otra vista muestra un menú con todos los ámbitos dentro del proceso, para así poder cambiar de forma rápida entre un ámbito y otro.

5.5. Selección de Servicios de Interacción y Vista de la Herramienta

En la aplicación anterior, esta vista es independiente del editor de grafos, lo que hace poco evidente la relación entre éste y el mockup final de la aplicación que se está modelando. Como se puede observar en la Figura 16, tanto la matriz de selección de servicios disponibles para un rol, como el mockup de la aplicación se incorporan directamente a la vista del editor. De esta manera, se pueden observar al mismo tiempo los cambios, tanto en el grafo como en el *mockup*. Una desventaja de esto es que los servicios ahora están ordenados en pestañas y no se pueden mostrar todos de forma simultánea.

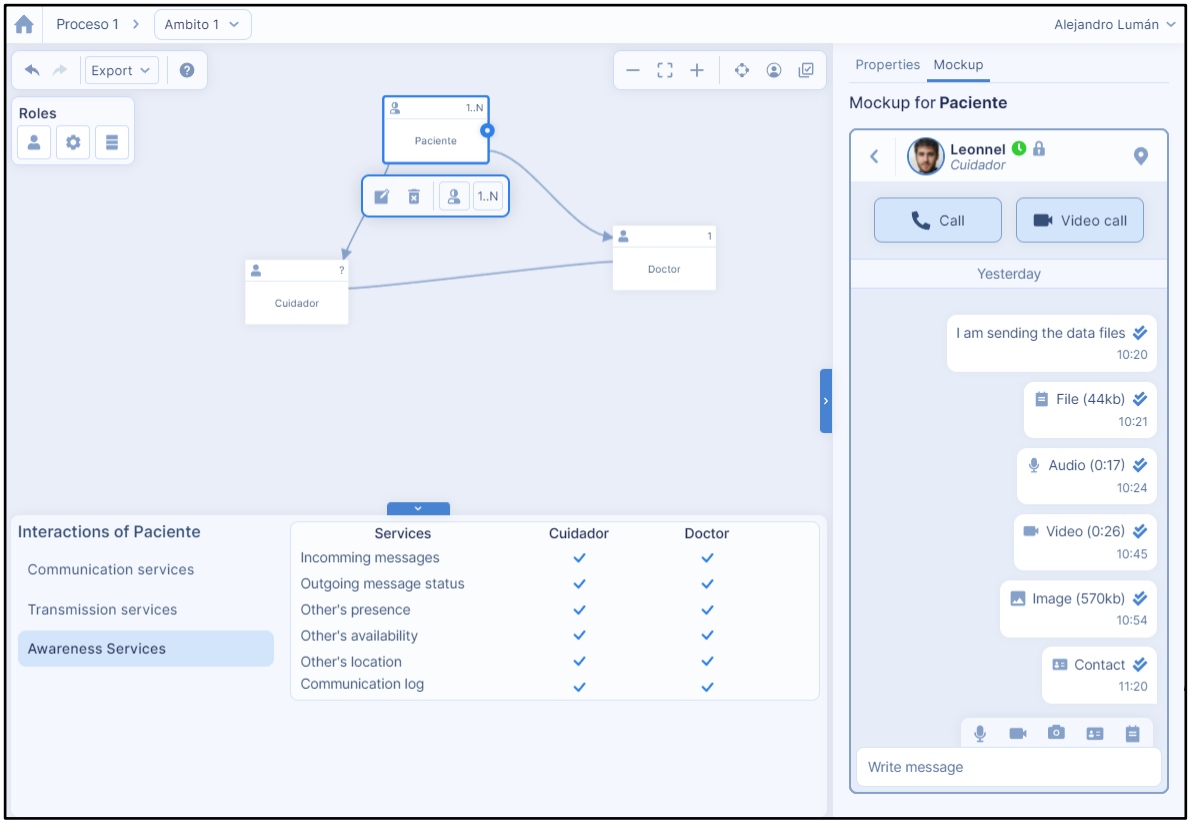

Figura 16. Vista de selección de servicios y prototipo de la aplicación móvil resultante

5.6. Nuevos Elementos de la Herramienta

Las vistas del mockup de la aplicación colaborativa móvil (Figura 17) son muy similares a la herramienta anterior, solo cambiando los estilos. La aplicación móvil representa como

un usuario (correspondiente al rol seleccionado en el grafo) observaría la aplicación a la cual se está modelando. Las vistas correspondientes a la aplicación cambian según los servicios que han sido seleccionados (es decir, las interacciones posibles entre un rol y otro).

La figura 17 representa una aplicación en donde la mayoría de los servicios se encuentran habilitados para las interacciones. A la izquierda se puede observar una lista de actores con los que el rol seleccionado puede interactuar. Al centro se observa una interacción con servicios de mensajería para un actor del tipo "Cuidador", y finalmente a la derecha se observa el servicio de videollamadas para esta interacción.

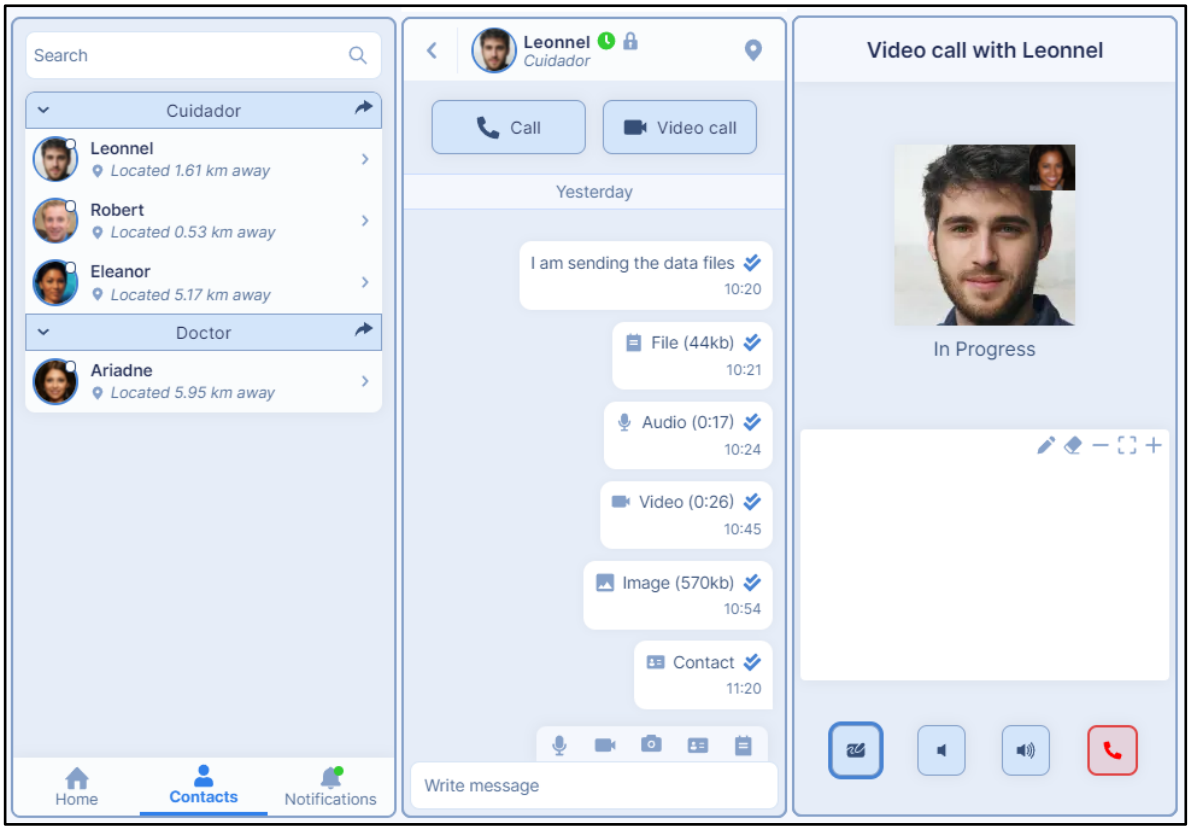

Figura 17. Vistas del prototipo de la aplicación móvil.

Además, se agregan *mockups* de prototipos para diferentes tipos de aplicaciones colaborativas (Figura 18). Estos mockups dependen del tipo de proceso en el que se encuentre el ámbito, y además se puede personalizar con una imagen que ingrese el usuario. En la figura, la imagen de más a la izquierda corresponde a un proceso de tipo "restaurant" en donde se puede vender y entregar comida. Al centro se observa un prototipo de aplicación para compartir vehículos (tipo "*ride sharing*"), y a la derecha, una aplicación para manejar tareas de tipo colaborativo (tipo "*todo / kanban*").

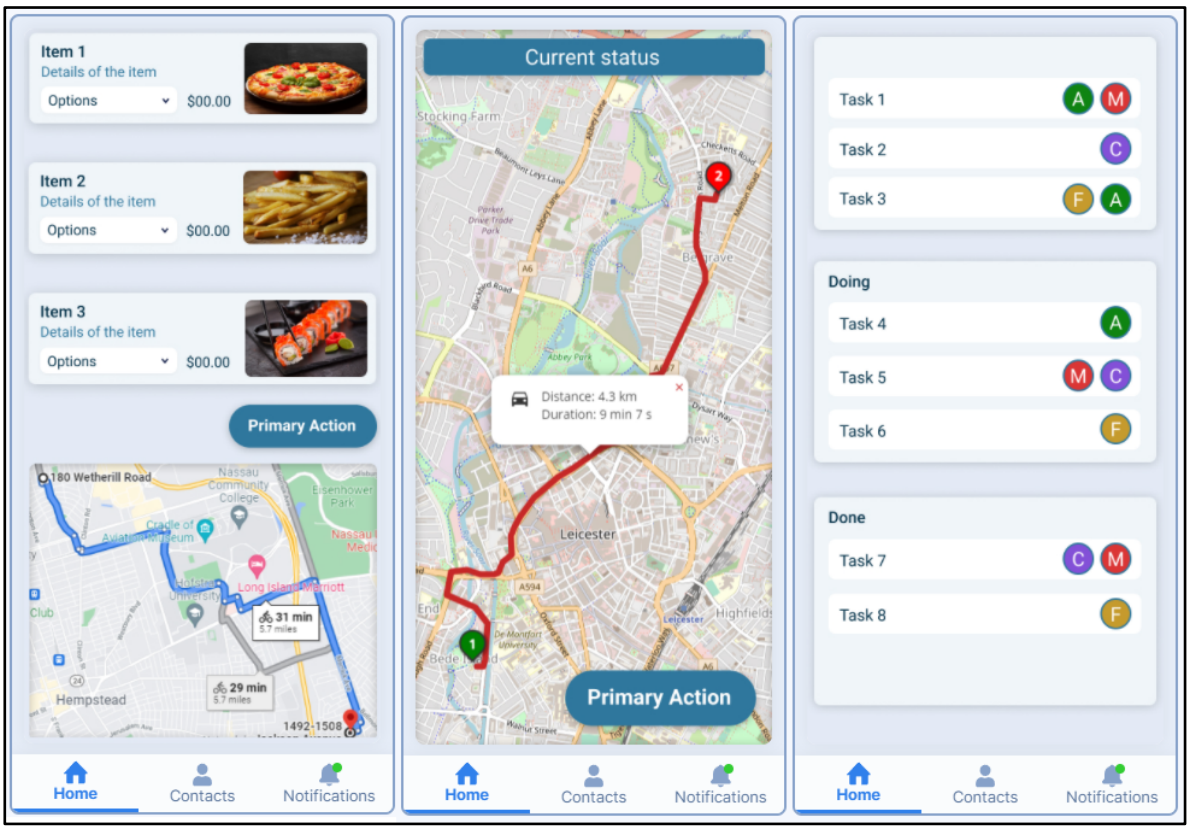

Figura 18. Ejemplo de *mockups* de una aplicación móvil generada con la herramienta.

6. Evaluación de la Nueva Herramienta

En esta sección se describe el proceso de evaluación de la herramienta desarrollada[6]. Para llevar a cabo esta labor, se realiza una evaluación con potenciales usuarios de la aplicación, además de una evaluación comparativa con respecto al sistema legado. Antes de esto, cabe destacar que durante el desarrollo su pudieron cumplir todos los requisitos definidos en la sección 4.2 de este documento.

Además, durante el desarrollo se utilizan herramientas como *linters* y formateadores de código que permiten que el código escrito se adecue a convenciones ampliamente aceptadas. Los componentes se encuentran en archivos pequeños y se procura que sean reutilizables. Debido a esto, se considera que la calidad del código, y, por lo tanto, la capacidad de extenderlo a futuro ha mejorado en comparación a la aplicación de legado.

6.1. Evaluación Comparativa Usando KLM

A continuación, se describe el método conocido como *keystroke-level model* (KLM), la forma en que éste se usa para evaluar las prestaciones de la nueva herramienta, y los resultados obtenidos.

6.1.1. Descripción del método KLM

El método comparativo KLM intenta predecir el tiempo en que un usuario experto demora en cumplir una tarea rutinaria en un sistema, sin cometer errores. Éste se basa en dividir las acciones que realiza un usuario en seis operadores distintos, los cuales tienen una determinada ponderación en segundos. De esta forma, el tiempo que toma una secuencia de acciones que el usuario siga, se puede aproximar como la suma de todos estos tiempos.

Si bien este método es muy útil para comparar versiones distintas de un diseño, tiene como desventaja que solo se puede aplicar a acciones unitarias, más que a un proceso completo. Además, sólo evalúa el tiempo de ejecución, y no de aprendizaje. Por esto mismo, resulta un buen complemento a las evaluaciones realizadas por medio de la escala SUS.

Los seis operadores que tiene este método son los siguientes:

- *K: presionar una tecla o botón*. Este es el operador más frecuente. Si bien el tiempo que un usuario tarda en escribir puede variar bastante según las habilidades de este, se puede tomar como un promedio alrededor de 0.2 segundos para una tecla. En el caso de un botón del *mouse*, se puede tomar un tiempo promedio de 0.1 segundos (se designará como B en la comparativa para poder diferenciar).

- *P: apuntar con un mouse a un objetivo en la pantalla*. Si bien por la ley de Fitts [11], el tiempo que toma un usuario en realizar esta acción es variable, e incrementa según la distancia que recorra el cursor, para este modelo de evaluación se establece 1.1 segundos como un valor de referencia.

[6] https://cimon.vercel.app/

- *H: mover las manos del teclado al mouse o viceversa*. Incluye el tiempo al moverse entre los dispositivos y los ajustes de posición de las manos. Se establece un valor de 0.4 segundos como referencia.
- *D: dibujar de forma manual.* El operador depende de líneas que el usuario dibuja manualmente. La evaluación de esta variable es muy específica, y no aplica al sistema a evaluar en esta memoria.
- *M: preparación mental para ejecutar una acción*. Éste es el tiempo que el usuario demora en pensar o tomar una decisión para realizar una acción. El tiempo para este operador se establece en 1.35 segundos. Existe una heurística especial para determinar en dónde se debe agregar este operador, que se muestra a continuación:
    - Primero: Insertar M antes de todos los operadores de tipo K que no sean una continuación de texto (*string* de letras o números). Insertar M antes de todos los operadores P que seleccionan algún comando.
    - Segundo: Si el operador que sigue a M puede ser anticipado por el usuario en el operador previo, entonces borrar M.
    - Tercero: Si una repetición de operadores MK pertenece a una unidad cognitiva (por ejemplo, tipear el nombre de un comando), eliminar todas las M que no sean la primera.
    - Cuarto: Si K corresponde a el ingreso de un comando justo luego de ingresar un argumento (o *string* de texto), eliminar M enfrente de K.
    - Quinto: Si K representa el fin de un *string* constante, eliminar M enfrente de K. En caso de que sea un *string* variable (que el usuario debe procesar), mantener la M.
- *R: tiempo de respuesta del sistema*. Es muy dependiente del contexto y del sistema que se esté utilizando, por lo que no se puede considerar como constante. Para sistemas web, es posible considerarlo como la latencia desde el dispositivo del cliente hacia el servidor. El operador sólo se utiliza si el usuario debe esperar para seguir usando el sistema.

6.1.2. Resultados obtenidos

Para realizar la evaluación, se han seleccionado diez interacciones comunes dentro de la aplicación, para las cuales se obtienen todos los operadores, tanto en la versión de legado, como la implementada durante este trabajo. Con los operadores de cada una, se procede a calcular el tiempo que demora un usuario promedio en realizar estas acciones y se presentan los resultados en tablas comparativas.

- Operación 1: Crear un rol en la vista del editor de interacciones:

Aplicación legada	Aplicación nueva
1) Iniciar la acción (M) 2) Encontrar el ícono de rol (M) 3) Mover el cursor al ícono (P) 4) Presionar sobre ícono (B) 5) Mover el cursor a donde se desea agregar el rol (P) 6) Presionar botón del *mouse* (B) 7) Mover manos al teclado (H) 8) Escribir nombre para el rol (K * 6) 9) Presionar tecla *enter* (K)	1) Iniciar la acción (M) 2) Encontrar el ícono de rol (M) 3) Mover el cursor al ícono (P) 4) Presionar y mantener presionado el cursor (B) 5) Arrastrar el rol hacia el escenario (P) 6) Soltar botón del *mouse* (B) 7) Mover manos al teclado (H) 8) Escribir nombre para el rol (K * 6) 9) Presionar tecla *enter* (K)
Total = 2M + 2B + 2P + H + 7K	Total = 2M + 2B + 2P + H + 7K
Tiempo = 6.9 segundos	Tiempo = 6.9 segundos

Tabla 1. Comparativa al crear un rol.

- Operación 2: Crear una interacción en la vista del editor de interacciones:

Aplicación legada	Aplicación nueva
1) Iniciar la acción (M) 2) Encontrar el ícono de interacción (M) 3) Mover el cursor al ícono (P) 4) Presionar sobre ícono (B) 5) Mover el cursor al rol origen de la interacción (P) 6) Presionar y mantener presionado el cursor (B) 7) Arrastrar hasta el rol destino de la interacción (P) 8) Soltar botón del *mouse* (B)	1) Iniciar la acción (M) 2) Encontrar el ícono del conector en el rol de origen (M) 3) Mover el cursor al ícono (P) 4) Presionar y mantener presionado el cursor (B) 5) Arrastrar hasta el rol destino de la interacción (P) 6) Soltar botón del *mouse* (B)
Total = 2M + 3B + 3P	Total = 2M + 2B + 2P
Tiempo = 6.3 segundos	Tiempo = 5.1 segundos

Tabla 2. Comparativa al crear una interacción.

- Operación 3: Cambiar propiedad de un rol (tipo, número de actores o uso de aplicación) ya seleccionado:

Aplicación legada	Aplicación nueva
1) Iniciar la acción (M) 2) Encontrar botón con la propiedad que se desea cambiar (M) 3) Mover cursor al botón de la propiedad que se va a cambiar (P) 4) Hacer *click* (se desplegará un menú) (B) 5) Mover cursor al nuevo valor a seleccionar (P) 6) Hacer *click* (B)	1) Iniciar la acción (M) 2) Encontrar botón con la propiedad que se desea cambiar (M) 3) Mover cursor al ícono con el valor a cambiar (P) 4) Hacer *click* (B)
Total = 2M + 2B + 2P	Total = 2M + B + P
Tiempo = 5.1 segundos	Tiempo = 3.9 segundos

Tabla 3. Comparativa al cambiar propiedad de rol.

- Operación 4: Añadir un enlace de grupo (herencia) en la vista del editor de interacciones:

Aplicación legada	Aplicación nueva
1) Iniciar la acción (M) 2) Encontrar el ícono de herencia (M) 3) Mover el cursor al ícono (P) 4) Presionar sobre ícono (B) 5) Mover el cursor al rol origen de la interacción (P) 6) Presionar y mantener presionado el cursor (B) 7) Arrastrar hasta el rol destino de la interacción (P) 8) Soltar botón del *mouse* (B)	1) Iniciar la acción (M) 2) Encontrar el ícono del conector en el rol de origen (M) 3) Mover el cursor al ícono (P) 4) Presionar y mantener presionado el cursor (B) 5) Arrastrar hasta el rol destino de la interacción (P) 6) Soltar botón del *mouse* (B) 7) Mover cursor a ícono de herencia (P) 8) Hacer *click* en ícono de herencia (B)
Total = 2M + 3B + 3P	Total = 2M + 3B + 3P
Tiempo = 6.3 segundos	Tiempo = 6.3 segundos

Tabla 4. Comparativa al añadir un enlace de grupo.

- Operación 5: Cambiar de ámbito desde la vista del editor de interacciones:

Aplicación legada	Aplicación nueva
1) Iniciar acción (M) 2) Encontrar ícono "atrás" del navegador (M) 3) Mover puntero al ícono (P) 4) Hacer *click* en el ícono (B) 5) Esperar carga de la página (R) 6) Encontrar botón del ámbito deseado (M) 7) Mover puntero al botón (P) 8) Hacer *click* en el botón (B)	1) Iniciar acción (M) 2) Encontrar botón para cambiar ámbito (M) 3) Mover puntero al botón (P) 4) Hacer *click* en el botón (B) 5) Encontrar ámbito deseado (M) 6) Mover puntero al ámbito (P) 7) Hacer *click* en el ámbito (B)
Total = 3M + 2P + 2B + R	Total = 3M + 2P + 2B
Tiempo = 6.45 segundos y tiempo de respuesta del servidor (~1 segundo)	Tiempo = 6.45 segundos

Tabla 5. Comparativa al cambiar de ámbito.

- Operación 6: Agregar un ámbito ya creado a una fase en la vista de proceso:

Aplicación legada	Aplicación nueva
1) Iniciar acción (M) 2) Mover el cursor al ámbito deseado (P) 3) Hacer *click* derecho sobre el ámbito (B) 4) Mover cursor a la opción para configurar ámbito (P) 5) Hacer *click* en la opción configurar ámbito (B) 6) Encontrar nombre de la fase a la que se desea agregar el ámbito (M) 7) Mover cursor al *checkbox* al lado del nombre (P) 8) Hacer *click* en el *checkbox* (B) 9) Mover cursor al botón para terminar la configuración (P) 10) Hacer *click* sobre el botón (B)	1) Iniciar acción (M) 2) Encontrar nombre de la fase a la que se desea agregar el ámbito (M) 3) Mover cursor al *checkbox* en la columna de la fase (P) 4) Hacer *click* en el *checkbox* (B)
Total = 2M + 4P + 4B	Total = 2M + P + B
Tiempo = 7.5 segundos	Tiempo = 3.9 segundos

Tabla 6. Comparativa al agregar un ámbito a una fase.

- Operación 7: Crear un proceso:

Aplicación legada	Aplicación nueva
1) Iniciar acción (M) 2) Encontrar botón para añadir proceso (M) 3) Mover cursor al botón (P) 4) Hacer *click* en el botón (B) 5) Mover cursor a la caja de texto (P) 6) Hacer *click* en la caja de texto (B) 7) Mover manos al teclado (H) 8) Escribir nombre para el proceso (K*6) 9) Presionar tecla *enter* (K)	1) Iniciar acción (M) 2) Encontrar botón para añadir proceso (M) 3) Mover cursor al botón (P) 4) Hacer *click* en el botón (B) 5) Mover manos al teclado (H) 6) Escribir nombre para el proceso (K*6) 7) Presionar tecla *enter* (K)
Total = 2M + 2P + 2B + 7K + H	Total = 2M + P + B + 7K + H
Tiempo = 6.9 segundos	Tiempo = 5.7 segundos

Tabla 7. Comparativa al crear un proceso.

- Operación 8: Volver a la vista de procesos desde la vista de editor de interacciones:

Aplicación legada	Aplicación nueva
1) Iniciar acción (M) 2) Encontrar ícono “atrás” del navegador (M) 3) Mover puntero al ícono (P) 4) Hacer dos *clicks* en el ícono (B)	1) Iniciar acción (M) 2) Encontrar ícono de inicio (M) 3) Mover puntero al ícono (P) 4) Hacer *click* en el ícono (B)
Total = 2M + P + 2B	Total = 2M + P + B
Tiempo = 4 segundos	Tiempo = 3.9 segundos

Tabla 8. Comparativa al volver a la lista de procesos.

- Operación 9: Cambiar el rol del *mockup*:

Aplicación legada	Aplicación nueva
1) Iniciar acción (M) 2) Encontrar botón para cambiar rol (M) 3) Mover cursor al botón (P) 4) Hacer *click* en el botón (B) 5) Encontrar nuevo rol (M) 6) Mover cursor a nuevo rol (P) 7) Hacer *click* en nuevo rol (B)	1) Iniciar acción (M) 2) Encontrar nuevo rol (nodo del grafo) (M) 3) Mover cursor al rol (P) 4) Hacer *click* en el rol (B)
Total = 3M + 2P + 2B	Total = 2M + P + B
Tiempo = 6.35 segundos	Tiempo = 3.9 segundos

Tabla 9. Comparativa al cambiar al modo *mockup*.

- Operación 10: Exportar datos a PDF desde la vista de editor de interacciones:

Aplicación legada	Aplicación nueva
1) Iniciar acción (M) 2) Encontrar botón para exportar a PDF (M) 3) Mover cursor al botón (P) 4) Hacer *click* en el botón (B)	1) Iniciar acción (M) 2) Encontrar menú de exportación (M) 3) Mover cursor al menú (P) 4) Hacer *click* en el menú (B) 5) Mover cursor al botón para exportar a PDF (P) 6) Hacer *click* en el botón (B)
Total = 2M + P + B	Total = 2M + 2P + 2B
Tiempo = 3.9 segundos	Tiempo = 5.1 segundos

Tabla 10. Comparativa al exportar a PDF.

De estos resultados se puede observar que se logra mantener o reducir el tiempo en el que se realizan la gran mayoría de las interacciones comunes en la aplicación. Esto conlleva a que los usuarios puedan realizar el modelamiento de forma más rápida y eficiente, puntos importantes dentro de la usabilidad de la aplicación.

Se puede observar también que en la operación 10, la aplicación nueva obtiene un tiempo peor que la aplicación legada. Esto es una decisión de diseño, ya que se opta por agrupar las opciones de exportación en un menú, el cual entrega más claridad respecto a la

funcionalidad en comparación con los íconos de la aplicación legada. Además, no debiese afectar en el modelamiento, pues la exportación se realiza una vez terminado el trabajo y no resulta ser una operación demasiado común durante el uso de la aplicación.

Por último, y como consideración importante, es que durante el diseño de la nueva aplicación también se toman en cuenta las distancias que debe recorrer el usuario al realizar acciones, tanto de forma visual como al mover el puntero por la pantalla. Las reducciones de tiempo y la mejor agrupación de la información que se obtienen gracias a esto no pueden ser evaluadas mediante este método, debido a que el tiempo que el usuario se demora en mover el cursor es considerado constante. Sin embargo, debido a la ley de Fitts, se sabe que este tiempo es menor si las distancias son más cortas, por lo que los tiempos calculados para la nueva aplicación podrían ser aún menores.

6.2. Evaluación Heurística Usando la Encuesta *SUS*

Tal como se mencionó antes, la escala SUS (System Usability Scale) [6] se aplica a través de una encuesta a usuarios de una aplicación. Su función es determinar cuán usable es un sistema según la percepción de los usuarios. Para ello se reclutan 7 participantes; todos utilizaron la herramienta para modelar las interacciones que se dan entre los roles involucrados en un proceso colaborativo. En base a eso la herramienta genera un prototipo (interfaces *clickeables*) de la aplicación móvil que apoya a dicho proceso.

Después de realizar esa labor, los participantes completan un cuestionario que incluye los ítems de la escala SUS, y además otros ítems que buscan determinar la utilidad de la herramienta, según la percepción de los usuarios. A continuación, se describen brevemente los distintos aspectos del proceso de evaluación, así como los resultados obtenidos.

6.2.1. Participantes del proceso

Tres de los participantes son académicos de la Universidad Politécnica de Cataluña (UPC), España, quienes colaboran actualmente con académicos del DCC en este ámbito. Los investigadores de la UPC acceden a participar en una sesión de trabajo para evaluar la herramienta desarrollada en esta memoria. Al finalizar el proceso, discuten brevemente los ítems de evaluación y completan en forma conjunta el instrumento que se describe en la sección 6.2.3.

Además, participan cuatro estudiantes y egresados del DCC, los cuales evalúan la herramienta de forma individual. Dos de estas personas ya estaban familiarizadas con la versión antigua de la herramienta, mientras que las otras dos eran usuarios nuevos. A estos últimos se les explican los conceptos necesarios para poder modelar un proceso muy similar al que se presenta en la sección 6.2.2. Además, se les deja utilizar libremente la aplicación para finalmente completar el mismo instrumento de evaluación.

6.2.2. Ejemplo de proceso modelado

A continuación, se muestra uno de los procesos colaborativos modelados, el cual corresponde a la coordinación entre repartidores de comida (tipo Globo o Uber Eats), y los restaurantes que proveen dichos productos. En el grafo de interacciones (Figura 19) se

pueden ver los roles participantes: el repartidor (*picker*), un agente inteligente que recibe y valida las solicitudes de comida (*collector*), la cola de pedidos (*orders queue*), el administrador del local (*manager*), el jefe de cocina del establecimiento (*kitchen chief*) y el personal de cocina (*kitchen personnel*).

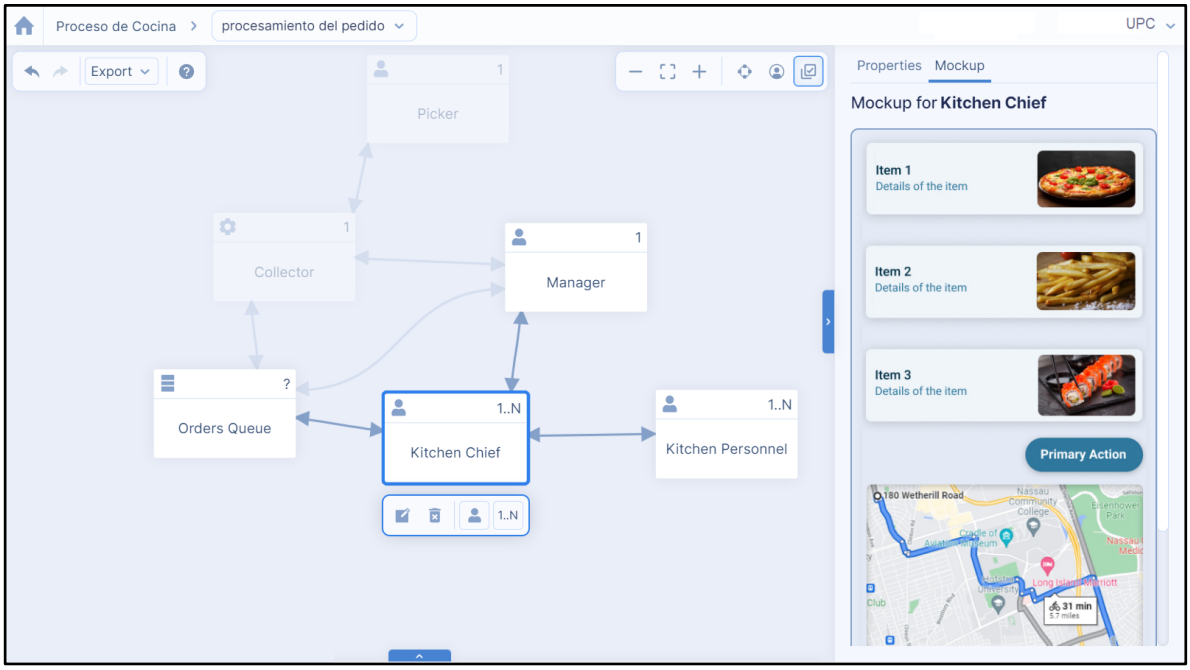

Figura 19. Escenario de interacción modelado.

En la Figura 19 se ha hecho foco en el rol "*kitchen chief*", y por lo tanto se muestran las características y capacidades de ese rol. Particularmente, en la parte derecha de la interfaz se muestra parte del prototipo al que tiene acceso dicho rol.

6.2.3. Instrumento de evaluación utilizado

La escala SUS [6] permite medir de forma simple y rápida la usabilidad de un sistema, basándose en las respuestas que entregue un usuario a un cuestionario de diez aseveraciones. El evaluador debe emitir un juicio de valor con respecto a cada una de éstas, y señalar qué tan de acuerdo (o desacuerdo) está con cada ítem. Luego de responder, cada respuesta se convierte en un puntaje entre 1 y 5, en donde 1 corresponde a la aseveración "*estoy completamente en desacuerdo*" y el 5 corresponde a "*estoy completamente de acuerdo*".

El cuestionario SUS, aplicado durante la evaluación de la herramienta es el siguiente:

1) Creo que me gustaría usar este sistema frecuentemente.
2) Encuentro al sistema innecesariamente complejo.
3) Creo que el sistema es fácil de usar.
4) Creo que necesitaría ayuda de una persona con conocimientos técnicos para usar este sistema.
5) Las funciones de este sistema están bien integradas.
6) Creo que el sistema es muy inconsistente.

7) Imagino que la mayoría de la gente aprendería a usar este sistema en forma muy rápida.
8) Encuentro que el sistema es muy engorroso de usar.
9) Me siento seguro/a al usar este sistema.
10) Necesité aprender muchas cosas antes de aprender a usar este sistema.

Una vez obtenido el puntaje correspondiente a cada ítem, se procede a transformarlo en un puntaje total (Nota de Usabilidad: NUs), para lo cual se utiliza la siguiente fórmula, en donde cada Ri corresponde a la respuesta en el ítem i:

$$NUs = 2{,}5 * [(R1 - 1) + (R3 - 1) + (R5 - 1) + (R7 - 1) + (R9 - 1) + (5 - R2) + (5 - R4) + (5 - R6) + (5 - R8) + (5 - R10)]$$

Este resultado determina el rango de usabilidad en el que se encuentra la aplicación para un determinado usuario. Los rangos son los siguientes:

- Menos de 25 puntos, el escenario es "el peor imaginable".
- Entre 25 y 38, entonces la usabilidad es considerada "pobre".
- Entre 39 y 52, la usabilidad es considerada "ok" (puede ser considerado un mínimo aceptable).
- Entre 53 y 73, la usabilidad es considerada "buena".
- Entre 74 y 85, la usabilidad es considerada "excelente".
- Entre 86 y 100, el escenario es "lo mejor posible".

Adicionalmente al cuestionario anterior, se agregaron cinco aseveraciones que permiten evaluar la utilidad de la herramienta, de modo que se pueda evaluar qué tanto se cumplieron las expectativas iniciales de un usuario al utilizar la aplicación. Estas aseveraciones son las siguientes:

1) Los servicios ofrecidos por la herramienta son suficientes como para determinar cómodamente la solicitud del cliente y su alcance.
2) Hay servicios que deben ser agregados a la herramienta para realizar esta labor de manera más apropiada.
3) Hay servicios que, aunque no son mandatorios, podrían ser agregados a la herramienta para hacerla más efectiva.
4) *Comentario abierto*: Indicar los servicios que a su juicio deben ser agregados a la herramienta, y también aquellos que usted considera como deseable de agregar.

Los primeros tres ítems se responden de forma similar que el apartado anterior, aunque estos se refieren a la utilidad del sistema. El último ítem es un comentario abierto, que no se incluye en la evaluación final. La utilidad de la herramienta puede ser calculada por la siguiente fórmula (Nota de Utilidad: NUt):

$$NUt = 0{,}5 * (R11 - 1) + 0{,}35 * (5 - R12) + 0{,}15 * (5 - R13)$$

De esta forma se puede observar que cada ítem tiene una ponderación diferente dependiendo de cuánto influye en la utilidad percibida de la aplicación. El puntaje máximo e ideal es 4, y la aplicación es considerada más útil cuanto más se acerque a este puntaje.

6.2.4. Resultados obtenidos

A continuación, se presenta una tabla con los resultados obtenidos luego de la aplicación del instrumento de evaluación. Es importante aclarar que, para calcular los promedios, el grupo evaluador de tres personas se ponderó adecuadamente.

Usuario / Ítem	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Total**
Usuario 1	5	2	5	2	5	1	4	1	5	1	92.5
Usuario 2	5	1	5	1	4	1	4	1	4	1	92.5
Usuario 3	5	1	5	4	5	1	5	1	5	1	92.5
Usuario 4	4	1	4	3	5	1	4	1	4	2	82.5
Grupo (3 personas)	4	2	4	1	4	1	4	1	4	2	82.5
Promedio	4.43	1.57	4.43	1.85	4.43	1	4.14	1	4.29	1.57	**86.8**

Tabla 11. Resultados de evaluación de usabilidad.

Se puede observar que existe un gran consenso en las respuestas entregadas por los usuarios evaluadores, especialmente en los ítems 6 y 8. Adicionalmente, para todos los usuarios se obtuvo el rango de usabilidad “excelente” o “lo mejor posible”, quedando el promedio dentro de la categoría “lo mejor posible”, lo que muestra que la usabilidad de la nueva herramienta es muy alta y los usuarios no presentaron mayores dificultades para modelar el proceso solicitado, o explorar las diversas opciones de la aplicación por su cuenta.

Un punto interesante que discutir, son los resultados del ítem 4 (la aseveración *“Creo que necesitaría ayuda de una persona con conocimientos técnicos para usar este sistema.”*). Se puede observar que los primeros dos usuarios, que ya estaban familiarizados con la aplicación legada, no estuvieron de acuerdo con ésta; sin embargo, los usuarios nuevos tuvieron respuestas dispares. En primer lugar, los usuarios 3 y 4, que evaluaron la aplicación de forma solitaria, respondieron que sí necesitan ayuda, lo que se puede explicar por la falta de conocimientos específicos del dominio de la aplicación. Los usuarios que evaluaron en grupo creen que no necesitan ayuda, lo que puede deberse a que tienen más conocimientos del dominio de la aplicación, o simplemente, al evaluar en grupo, se les hizo más fácil entender estos conceptos mediante la discusión entre pares.

En cuanto a la utilidad de la herramienta, los resultados son los siguientes:

Usuario / Ítem	**1**	**2**	**3**	**Total**
Usuario 1	5	1	1	4
Usuario 2	4	3	3	2.5
Usuario 3	5	1	2	3.85
Usuario 4	5	1	4	3.55
Grupo (3 personas)	4	2	4	2.7
Promedio	4.43	1.71	3.14	**3.14**

Tabla 12. Resultados de la evaluación de la utilidad del sistema.

Al igual que en la tabla anterior, los promedios se calculan con la respectiva ponderación para el grupo de tres personas. Los resultados están dentro de un rango medio-alto, lo que indica que, si bien la aplicación resulta útil para el modelamiento de aplicaciones que soportan procesos colaborativos, existen características deseables que se podrían agregar.

Luego de responder las preguntas de selección, los usuarios escribieron comentarios sobre servicios deseables que faltan en la aplicación, además de apreciaciones generales. A continuación, se presentan puntos importantes a rescatar de estos:

- *Vista de procesos:* El comentario que más se repite entre los usuarios es que la matriz con fases y ámbitos en la vista de procesos resulta un poco difícil de entender. Los usuarios tuvieron dificultad para saber que corresponde a un ámbito o una fase. Si bien se optó por este diseño para dejar en claro que un ámbito puede pertenecer a varias fases, existe cierto nivel de indirección y queda poco claro *para qué* sirve la matriz. Esto se podría solucionar agregando una especie de tutorial explicando cada paso, o bien, cambiando el diseño y volver a uno más tradicional como la aplicación de legado.

- *Estética y cohesión de los elementos de la interfaz:* Casi todos los usuarios destacan el aspecto estético de la herramienta, además de la consistencia y cohesión de los elementos gráficos, lo que finalmente ayuda a dejar en claro el propósito de cada opción sin que se necesiten muchas explicaciones. Los usuarios también comentan que la aplicación les pareció bastante *intuitiva*.

- *Funciones de exportación e importación:* Los usuarios valoran la capacidad de exportar los resultados en diversos formatos, pero mencionan algunas posibles mejoras: al exportar un PDF solo se incluyen los servicios por rol, sin embargo, también se podría incluir una imagen del grafo obtenido o del *mockup* de la aplicación por cada rol.

- *Mejoras menores en el editor:* Un usuario menciona que hace falta la capacidad de invertir una relación (es decir, cambiar el rol de origen por el de destino, y viceversa). Esto podría resultar útil, ya que actualmente se tendría que borrar la interacción y agregar una nueva. También se menciona que sería útil añadir una forma de marcar todos los servicios de cierto tipo a la vez, en vez de ir marcándolos uno por uno, lo que agilizaría el proceso de modelamiento.

- *Características deseables:* Se menciona que falta la capacidad de añadir servicios de interacción dinámicos (creados por el usuario), debido a que muchos procesos cuentan con servicios propios a su dominio y que difícilmente son representados por los que ya están incluidos. También hace falta la capacidad de importar un grafo, para complementar la capacidad de exportar.

7. Conclusiones y Trabajo a Futuro

Si bien la versión de la aplicación legada cumplía con los requisitos mínimos para poder modelar un proceso colaborativo, ésta contaba con diversos problemas de usabilidad que hacían poco eficiente el modelado. Por ejemplo, tenía problemas de navegación entre las vistas de los modelos, requería un alto número de acciones para realizar acciones simples, y había diversos errores que dificultaban la operación. Además, el código fuente de la herramienta estaba escrito de una forma que hacía muy difícil su comprensión, y, en consecuencia, la factibilidad de solucionar errores y extenderla a futuro.

El trabajo realizado recae, por lo tanto, en un rediseño completo de la aplicación, partiendo por establecer las mejoras necesarias para crear propuestas de prototipos de alta fidelidad. Estos prototipos fueron evaluados iterativamente por usuarios a medida que se implementaba la nueva aplicación, por lo que se pudo incluir mejoras que no se habían tomado en cuenta cuando comenzó el proceso de diseño.

También, como parte del trabajo, se incluyó un sistema de manejo de usuarios, algo fundamental para que el sistema cuente con un uso real. Además, se agregaron diversas características menores, como la capacidad de buscar procesos, o añadir una descripción a cada ámbito. Esto permite que el modelado sea más expresivo, y que se pueda incluir información adicional útil para durante el proceso.

De la evaluación mediante el sistema KLM se obtuvieron resultados satisfactorios, mostrando una reducción en el tiempo necesario para realizar acciones comunes sobre la herramienta. A pesar de esto, en varios casos no se logró reducir mucho este tiempo, pero si se logró reducir las distancias que el usuario necesita recorrer tanto con su visión, como con el puntero del *mouse*. Estas distancias se redujeron principalmente con la correcta agrupación de acciones similares en la interfaz.

Los resultados obtenidos mediante la evaluación con usuarios, utilizando la escala SUS, fueron muy positivos. La herramienta quedó calificada como con “la mejor usabilidad posible” según dicha escala. En los comentarios los usuarios destacaron la estética, consistencia e intuitividad de ésta. Además de esto, la utilidad de la herramienta fue evaluada en un rango medio-alto, y los usuarios hicieron alusión a algunos servicios que podrían faltar, o mejoras que se podrían implementar a futuro. Como resultado final, se puede establecer que se cumplió con el objetivo principal del trabajo, que era mejorar la usabilidad de la herramienta. Además, se logró cumplir con los objetivos de reimplementación del *backend* y *frontend* de la aplicación, así como de un sistema simple para representar la lógica de negocio de la aplicación en los *mockups* que genera la herramienta.

A futuro, la aplicación puede seguir siendo extendida para incrementar la utilidad que presenta, y ayudar tanto a desarrolladores como *stakeholders* a llevar el proceso de modelado de una forma exitosa. En primer lugar, se pueden incluir las recomendaciones mencionadas por los usuarios durante la evaluación de la herramienta, tales como mejorar la comprensibilidad de la matriz en la vista de procesos, o agregar servicios de forma dinámica. Otro aspecto deseable, y que podría requerir de un trabajo considerable, es agregar un *sistema colaborativo* a la plataforma. Esto permitiría que varios usuarios,

tanto desarrolladores como *stakeholders,* participen en el modelamiento del proceso en *tiempo real,* desde computadores diferentes.

Durante el desarrollo de este trabajo se necesitó profundizar conocimientos en diversas tecnologías de desarrollo web, además de aprender algunas nuevas. También, fue siempre necesario organizar el tiempo adecuadamente. Originalmente, se tenía planeado terminar el trabajo durante las 14 semanas correspondientes del semestre, sin embargo, se excedió este plazo, principalmente debido a que los últimos detalles a implementar requirieron considerablemente más tiempo del estipulado, por lo que en planificaciones futuras se tendrá en cuenta que el tiempo necesario durante el desarrollo final puede ser mayor al esperado.

Finalmente, se valora la experiencia obtenida al trabajar con usuarios de forma iterativa durante el desarrollo de este trabajo. En muchos casos, los usuarios son dejados de lado en el transcurso de un proyecto, aun cuando estos pueden aportar información y puntos de vista muy valiosos para el éxito de éste. Sin duda, los buenos resultados en la evaluación de esta herramienta fueron en gran medida gracias a eso.

8. Bibliografía

[1] Canché, M., Ochoa, S.F., Perovich, D., Gutiérrez F.J. Analysis of notations for modeling user interaction scenarios in ubiquitous collaborative systems. J. of Ambient Intelligence and Humanized Computing (2019). https://doi.org/10.1007/s12652-019-01578-7.

[2] Canché, M. Modeling Computer-Mediated Interactions that Supports People-Driven Collaborative Processes. Propuesta de Tesis de Doctorado de Computación. DCC, FCFM, Universidad de Chile. Tesis en proceso (2021).

[3] Card, S.K., Newell, A., Moran, T.P. The Psychology of Human-Computer Interaction. L. Erlbaum Associates Inc., USA (1983).

[4] Ericsson, K.A., Simon, H.A. Verbal reports as data. Psychological Review, 87(3), 215–251 (1980) https://doi.org/10.1037/0033-295X.87.3.215.

[5] Nielsen, J., Molich R. Heuristic evaluation of user interfaces. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'90). Pages 249–256 (1990). https://doi.org/10.1145/97243.97281.

[6] Brooke, J. SUS: A “quick and dirty” usability scale. In Jordan, P.W., Thomas, B., McClelland, I.L., & Weerdmeester, B. (Eds.). Usability Evaluation in Industry (1st ed.). (pp.189-194). CRC Press. (1996). https://doi.org/10.1201/9781498710411.

[7] Documento Histórico del Proyecto. Proyecto Titulado: Editor de Escenarios de Interacción. Curso CC5401: Ingeniería de Software II. Semestre Otoño 2020. DCC, FCFM, Universidad de Chile, (2020).

[8] Fielding, R. Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, Irvine, 2000. https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

[9] Jones, Michael B.; Bradley, Bradley; Sakimura, Sakimura (2015). JSON Web Token (JWT). IETF. https://doi.org/10.17487%2FRFC7519

[10] Koffka, Kurt. Principles of Gestalt Psychology. London: Routledge & K. Paul, 1962.

[11] Fitts, Paul M. (June 1954). The information capacity of the human motor system in controlling the amplitude of movement. Journal of Experimental Psychology. https://doi.org/10.1037%2Fh0055392