Generación Automática de Mashups en Dispositivos Móviles

Monografía presentada para optar al título de Ingeniero en Electrónica y Telecomunicaciones

Diego Fabian Gómez Pardo
Franklin Esteban Navia Urbano

Director: PhD. Juan Carlos Corrales Muñoz
Asesor: Ing. Luis Javier Suarez Meza

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones

Departamento de Telemática

Popayán, Noviembre de 2014

Agradecimientos

Agradezco en primer lugar a Dios por permitirme a través de mis padres, familiares, profesores y amigos adquirir valores como la honestidad, humildad, responsabilidad, sacrificio y respeto para conmigo mismo y quienes me rodean.
Agradezco de manera especial a mi director y codirector de tesis, PhD. Ing. Juan Carlos Corrales y MSc. Luis Javier Suarez Meza, respectivamente, por su colaboración durante el tiempo de realización de la presente tesis, para guiar y conducir pensamientos y técnicas que permitieron desarrollar adecuadamente el presente trabajo.
Y finalmente a todas aquellas personas que durante mi paso por la universidad. dejaron en mí enseñanzas de entrega, sacrificio y ambición por conseguir siempre lo mejor tanto a nivel académico y profesiona.

Tabla de Contenido

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

1.1 Motivación

En los últimos años la evolución de las tecnologías, tanto web como móvil, han propiciado el crecimiento de la información y nuevas formas de crear aplicaciones, donde el usuario tiene la capacidad tanto de consumir como de generar contenido [3, 4]. Sin embargo, este crecimiento de información ha provocado que los procesos de búsqueda de recursos lleguen a ser tediosos para el usuario, debido a la cantidad de tiempo que debe invertir examinando su contenido y seleccionado los recursos apropiados para su tarea de búsqueda.

En respuesta a lo anterior, surgen los *Sistemas de Recomendación* (SR). Estos corresponden a mecanismos de filtrado de información, que analizan tanto el perfil como el comportamiento del usuario, entre otros, extrayendo información útil que permita encontrar recursos que se ajusten a las necesidades e intereses de una persona en particular [5].

Por otra parte, la combinación de los SR con herramientas de integración de recursos (por ejemplo Widgets, Mashups, etc.) pretenden que un usuario tenga la posibilidad de generar sus propios contenidos e incluso sus aplicaciones de forma más sencilla, a partir de recursos distribuidos en la web.

No obstante, si bien este conjunto de herramientas logran simplificar en cierta medida el proceso de búsqueda e integración de recursos, aún presentan inconvenientes; entre los más relevantes se destaca el conocimiento que las personas deben tener en temas de composición de recursos, a fin de seleccionar e integrar apropiadamente los recursos. Tareas que no son triviales y que demandan un tiempo considerable, el cual los usuarios no están dispuestos a invertir [6, 7].

Por otra parte, debido a la incorporación de los dispositivos móviles en el día a día de las personas, su comportamiento durante la búsqueda en la web ha cambiado notablemente, trasladando las búsquedas tradicionales sobre computadores personales a búsquedas ubicuas; es decir, que se puede efectuar en cualquier momento, lugar, e independientemente del dispositivo de acceso. Esta es una de las razones por la cual, el uso de dispositivos móviles ha tenido un aumento en los últimos años, a tal punto que cerca del 80 % de la población mundial tiene a su disposición algún dispositivo móvil, bien sean teléfonos inteligentes o tabletas que permitan llevar a cabo diversas actividades [8].

1.2 Problema de Investigación

Como consecuencia de la masificación en el uso de los dispositivos móviles, surge la *Búsqueda Móvil* (BM), que propicia el acceso ubicuo a Internet, permitiendo que cada vez más personas lleven a cabo diversas actividades vía Web sobre múltiples escenarios (como en el trabajo, vehículos, centros comerciales, avenidas, entre otros.). No obstante, existen actividades que no pueden ser satisfechas mediante el consumo de un único recurso, demandando la creación de nuevos recursos con base en los existentes (ej. Un usuario que desea aprender charango, conocer restaurantes cercanos a un sitio en especifico u obtener información respecto al estado del clima) [9].

Por lo tanto, la BM no sólo representa el cambio de búsqueda en la Web sobre un PC a buscar en un dispositivo móvil, sino que define un entorno que propicia nuevas formas de interacción y comunicación entre los individuos y su contexto. Lo anterior evidencia desafíos importantes, entre los que se destaca: la agregación de recursos heterogéneos (media tradicional y funcionalidades de los dispositivos móviles) sensibles al contexto y comportamiento del usuario. Escenario que propició la adopción del concepto de *Mashups* en la Web.

Así, en la actualidad el término Mashups se entiende como una aplicación híbrida, compuesta por diversos recursos (ej., datos, RSS, imágenes, etc.) procedentes de varias fuentes, con el fin de producir resultados de valor agregado, que satisfagan en mayor proporción las necesidades de información de las personas en la Web, propiciando una mejor experiencia del usuario [10, 3].

Sin embargo, la creación de mashups se está tornando insostenible, ya que la cantidad y diversidad de recursos disponibles en la Web, excede la capacidad humana para procesar su contenido de forma manual; fenómeno que afecta entre otros aspectos, la toma de decisiones de una persona durante su búsqueda en la Web.

Frente a este panorama, diversas herramientas han sido desarrolladas, entre las más populares se encuentran: Yahoo! Pipes[1], Presto Cloud[2] e IBM Mashup Center[3]. No obstante, son soluciones que si bien buscan soportar a usuarios finales durante la generación de mashups, estos sistemas requieren que sus usuarios posean cierto nivel de conocimiento sobre la tecnología subyacente para conseguirlo [11].

Hoy en día, la generación de mashups convencionales presenta una evidente desventaja, motivada por su pobre escalabilidad y difícil integración de recursos, resultado de la naturaleza estática con la cual se concibió este concepto. En consecuencia, investigaciones recientes han incorporado *ontologías*[4], con el fin de relacionar diversas fuentes de información de manera transparente, avanzando así al concepto de *Mashups Semánticos* (MS) [12].

Otros trabajos, como MARIO [13], están enfocados en la generación automática de mashups, que exponen propuestas interesantes sobre integración automática de recursos a nivel de capa de datos y de lógica de negocio (control), aspecto que favorece a los usuarios finales sin conocimientos técnicos sobre Internet y sus tecnologías base. Sin embargo, el inconveniente radica en el tipo de recurso sobre el cual esta propuesta trabaja: datos; siendo lo anterior, similar a la composición tradicional de servicios web, como BPEL (*Business Process Execution Language*). A pesar de ello, esta aproximación ofrece una base interesante hacia la concepción de una "nueva forma de generación de mashups", puesto que muy poca atención se ha prestado a la *integración*[5] con base

[1] **Disponible en:** http://pipes.yahoo.com/pipes/.
[2] **Disponible en:** http://mdc.jackbe.com/enterprise-mashup/.
[3] **Disponible en:** http://www.ibm.com/developerworks/lotus/products/mashups/.
[4] **Ontologías:** Conjunto de términos que relacionan definiciones amplias.
[5] **Integración:** Combinación coherente de recursos que se encuentran en la web, con el fin de mostrar contenido

en la interacción del usuario con los recursos heterogéneos a nivel de capa de presentación (*Interfaz de Usuario*, UI por sus siglas en ingles) [10, 7]; es decir, permitir a un usuario —sin conocimientos técnicos— generar sus propios mashups en tiempo de ejecución —es decir, de manera totalmente transparente para él—, mediante su refinamiento iterativo con los mashups resultantes, hasta obtener el resultado deseado.

Con base en el panorama descrito, este trabajo de grado busca responder la siguiente pregunta de investigación:

¿Cómo permitir que una persona sin conocimientos técnicos (usuario final), genere mashups sobre dispositivos móviles?

1.3 Hipótesis y Objetivos

1.3.1 Hipótesis De Investigación

A continuación se introducen la hipótesis que se pretende probar por medio del presente trabajo, y la cuale son formuladas con base en la pregunta de investigación planteada en la sección 1.2.

- **H1:** *La interacción del usuario con mashups parciales (comportamiento del usuario con el sistema) posibilita la generación paulatina del mashup final.*

1.3.2 Objetivos De Investigación

A continuación se presentan los objetivos de investigación, mediante los cuales se persigue explorar la hipótesis planteada.

1.3.2.1 Objetivo General

Automatizar la generación de mashups en un dispositivo móvil a partir de la interacción de un usuario final con el sistema en tiempo de ejecución.

1.3.2.2 Objetivos Específicos

1. Proponer un modelo que capture los elementos necesarios para crear un mashup sobre un dispositivo móvil.

2. Proponer un mecanismo que genere mashups en tiempo de ejecución con base en el modelo definido.

3. Evaluar la relevancia[6] del mashup generado por el mecanismo propuesto en un entorno de prueba.

pertinente como resultado.

[6] **Relevancia:** Entiéndase como el grado de aceptación por parte del usuario, es decir, un mashup generado es relevante en tanto se ajusta a los requerimientos iníciales del usuario.

1.4 Alcance

Esta tesis de pregrado buscar satisfacer los requerimientos de las consultas de búsqueda a través de la generación automática de mashups en dispositivos móviles. De esta manera, el presente proyecto pretende reducir el nivel de conocimiento necesario para generar dichos mashups, de manera que los usuarios finales puedan generarlos y satisfacer aquellas tareas diarias que así lo requieran. Cabe destacar que en el presente trabajo de investigación se abordaran los siguientes aspectos:

- Las fuentes de información utilizadas para el proyecto son: Flicker, Youtube, Wikipedia y Foursquare, las cuales son de naturaleza informacional.
- En el presente proyecto solo se desarrolla mashups de clientes y no de tipo empresarial, Sección 2.2.1.1.

1.5 Contribuciones

Las principales contribuciones de éste proyecto de grado son:

- Definición e implementación de un modelo de generación automática de mashups. Este es uno de los más importantes aportes del presente proyecto de grado, debido a que permite integrar diversos recursos Web de forma amigable a partir de la interacción (comportamiento) del usuario con el sistema en tiempo de ejecución.
- Definición e implementación de mecanismos de recomendación de componentes basados en los objetivos que los recursos pueden cumplir y el conocimiento de composición de mashups, que permitan satisfacer los objetivos de búsqueda de los usuarios.
- Definición de una arquitectura para sistemas de generación de mashups sobre dispositivos móviles. Este aporte es el resultado del análisis a nivel de ingeniería de varios trabajos que abordan la creación de mashups, teniendo como objetivos principal la integración automática de recursos heterogéneos.
- Construcción de una herramienta nativa sobre un dispositivo móvil, en el cual se implementan los lineamientos propuestos en este proyecto. Diseñado con el propósito de facilitar la creación de mashups.
- Evaluación de relevancia, en el cual se determina la pertinencia de los mashups generados a través de la herramienta propuesta en el presente proyecto; es decir que tengan la capacidad de satisfacer los requerimientos de los usuarios finales.
- Evaluación de producto, en el cual se determina el grado de aceptación de la herramienta propuesta por parte de usuarios finales, con el objetivo de conocer si dicha herramienta puede ser utilizada para satisfacer los requerimientos de búsqueda.
- Se realizo una contribución a la comunidad de desarrolladores de RubyOnRails, sobre las librerías (Gem) Apriori y Neography, llevado a cabo en el presente proyecto de investigación, con el fin de ser usados en las versiones de Ruby 2.x.

- Motivar el desarrollo y la investigación en la comunidad académica de la Universidad del Cauca sobre la generación automática de mashups y el desarrollo de mecanismos inteligentes para la integración de recursos, implementando un prototipo funcional que cree un precedente alrededor de este tema, y despertar así el interés por estas tecnologías en la comunidad investigativa del país, debido al positivo panorama socio-económico, factible de obtener con su difusión.

- Fortalecimiento de las áreas de investigación del GIT y del grupo de interés en desarrollo de aplicaciones móviles e inalámbricas W@PCOLOMBIA a través de la incursión en el desarrollo de mashups para dispositivos móviles, el cual es un tema emergente de investigación en la actualidad.

- Como contribución final se destaca el aporte realizado al trabajo de maestría, titulada: "*Estimación del intento de búsqueda del usuario para la recuperación de recursos en la web*", desarrollado por el Ingeniero Luis Javier Suarez Meza, llevado a cabo en la Universidad del Cauca. De esta manera se contribuye al proceso de investigación en torno a la estimación del intento de búsqueda a través de la interacción del usuario con el sistema propuesto, con el fin de satisfacer la tareas de búsqueda.

1.6 Publicaciones

En construcción.

1.7 Organización del Documento

El contenido de la presente monografía se encuentra organizada en 6 capítulos, tal como se describe a continuación:

- **Capítulo 2. Contexto de Investigación**

Aborda las definiciones formales de conceptos claves para la comprensión de esta investigación, se construye una base inicial de conocimiento sobre los temas directamente relacionados con el presente proyecto de grado y se presentan los trabajos relacionados correspondientes.

- **Capítulo 3. Modelo de Mashup**

Este capítulo tiene como objetivo describir el enfoque propuesto en el cual se fundamenta el presente trabajo. En primera instancia, se expone un modelo que soporta la solución propuesta para generar mashups de manera automática, justificando su selección. Posteriormente, se enfatizará en los métodos para la recuperación y selección de componentes a partir del modelo propuesto.

- **Capítulo 4. Anatomía de la Solución**

En este capítulo se conceptualiza la solución propuesta, a partir de la selección de los mecanismos necesarios para el desarrollo del presente proyecto. Por último, se expone un estudio de caso con el fin de ejemplificar el enfoque propuesto.

- **Capítulo 5. Implementación de la Solución**

Corresponde a una descripción detallada del trabajo realizado alrededor del desarrollo de un sistema software, en el cual se implementa la propuesta del presente proyecto.

- **Capítulo 6. Evaluación y Resultados**

Contiene una descripción de la metodología de evaluación empleada, así como la selección y justificación del método de evaluación sugerido por la metodología. Además, se puntualiza el desarrollo de las pruebas y la interpretación de los resultados.

- **Capítulo 7. Conclusiones y Trabajos Futuros**

Por último, se exponen las conclusiones y algunas perspectivas de la investigación, analizando los resultados del trabajo realizado y brindando un conjunto de recomendaciones importantes para el desarrollo de trabajos futuros.

Capítulo 2

Contexto de Investigación

2.1 Introducción

Este capítulo está dedicado a describir los conceptos principales y trabajos de investigación relacionados con el presente proyecto. Inicialmente, se exponen los conceptos clave sobre los que se fundamenta este proyecto (Sección 2.2.1). Seguidamente, se presentan las bases teóricas que posibilitan categorizar las aproximaciones orientadas a la generación de mashups (Sección 2.2.2). Posteriormente, se describe el estado actual de la generación de mashups (Sección 2.3). Finalmente, como producto del análisis de las iniciativas detalladas, se resaltan las brechas existentes (Sección 2.4).

2.2 Marco Conceptual

Esta sección contiene conceptos relevantes para la comprensión y posterior análisis del problema central del presente trabajo de grado. Entre los conceptos más importantes se destacan los relacionados con mashups, tipos y la forma de generarlos.

2.2.1 Mashups

Un mashup es una composición definida por el usuario a partir de recursos distribuidos en la web, como servicios, fuentes de datos, APIs, entre otros, con el objetivo de proveer aplicaciones orientadas a resolver una necesidad específica, generando recursos enriquecidos que mejoran la experiencia de los usuarios en la web [14, 15].

Por ejemplo, un mashup puede ser la combinación de datos proporcionados por servicios de clima y un mapa de Google [16]; otro ejemplo puede centrarse en aplicaciones empresariales, que combinen información de los empleados de una compañía y sus actividades de trabajo, con el fin de llevar a cabo una evaluación su rendimiento, esto mediante la integración de servicios SOAP con graficas estadísticas.

De esta manera, el concepto de Mashups ha venido cobrando relevancia en Internet, puesto que brinda una perspectiva versátil al problema de combinar recursos heterogéneos de la Web, permitiendo a usuarios crear nuevas aplicaciones y/o contenido a la medida. Así, debido a las diversas investigaciones que ofrecen un compendio de categorías, la siguiente sección proporciona una clasificación a partir de los criterios empleados en las investigaciones sobre los que se

fundamenta el presente trabajo.

2.2.1.1 Tipos de Mashups

En primer lugar, cabe aclarar que hasta la fecha en la literatura no existe un consenso para clasificar los mashups, así como las herramientas para su generación debido a que no existe una madurez que definan dichas clasificaciones [17]. En su lugar, durante el crecimiento de la comunidad y la creación de herramientas para su generación, se han propuesto términos, criterios y definiciones —que aunque no son axiomas— corresponden a una base importante para su categorización [18].

A continuación, se exponen algunos de los criterios tomados en investigaciones para proponer una clasificación apropiada de mashups. Florian et. al. [19], define 8 categorías: mashups simples, de multi-paginas, guiados, de flujo de páginas, de páginas-compartidas, de espacio-compartido, de cooperación y de procesos, a partir de la definición de 3 criterios que son: el soporte de múltiples usuarios, navegación entre vistas y por último, los flujos de trabajo. No obstante, Zhao et. al. [20] propone 8 tipos: mashups de clientes, empresariales, front-end, back-end, de personalización de páginas Web, de procesos, horizontales y verticales, asociados a diversos criterios: el flujo de datos empleado, el tipo de resultado anhelado en función de las características de los recursos que se están combinando, entre otros.

Por otra parte, Hoyer et. al. [17] coincide con Zhao, en que los mashups pueden estar clasificados en dos categorías principales: mashups de clientes y empresariales. De esta manera, el criterio utilizado para clasificarlos se centran en el grupo de usuarios que crearán y harán uso de los mashups. Por lo tanto, esta categorización es relevante para el presente proyecto, debido a que los usuarios finales generaran sus propios mashups.

De acuerdo a esto, los mashups pueden centrarse en dos grupos: usuarios finales o desarrolladores, conocidos como: *mashups de clientes* y *empresariales*, descritos a continuación.

- **Mashups de Clientes:**

También conocido como "*Mashups de Consumo*", este tipo de mashup es creado para resolver problemas de la vida cotidiana y bajo una situación en particular. Inicialmente se crean para uso propio pero pueden satisfacer los requerimientos de otras personas que tengan necesidad es similares [17]. Por ejemplo, la visualización de una página web personalizable, donde se pueda combinar recursos de noticias, correos electrónicos y el estado del clima en una misma pantalla.

- **Mashups Empresariales:**

Este tipo de mashup está orientado a resolver problemas enfocados a entornos empresariales y lógica de negocios, que integren sistemas de "*back-end*" existentes y por lo tanto, requieren de una mayor colaboración para llevar a cabo los procesos de negocio de manera coordinada. En contraste a los mashups de clientes, estos requieren características adicionales como seguridad, calidad y disponibilidad [17, 20]. Por ejemplo, la generación de reportes sobre la cuota de mercado de inmobiliarias dado por dos servicios, el primero extrae una lista de todas las casas vendidas por cada inmobiliaria y el segundo, calcula el porcentaje de venta de cada inmobiliaria teniendo como entrada la salida del primer servicio.

En la siguiente sección se examinarán los tipos de generación de los mashups, aspectos en los que se fundamenta la presente propuesta.

2.2.2 Generación de Mashups

Tradicionalmente, la generación de mashups era realizada por desarrolladores expertos quienes identificaban necesidades comunes para grupos de personas que evidenciaban demandas similares. Actualmente, existen diversas plataformas de desarrollo centradas en el usuario final, como Mario [13] y NaturalMash [21], entre otros. Sin embargo, a pesar de la cantidad de trabajos que promulgan la "generación de mashups centrada en el usuario final", estas aproximaciones requieren que los usuarios posean cierto conocimiento de la tecnología subyacente para efectuar una composición de contenido exitosa.

De esta manera, debido a la aproximación de las herramientas para la generación de mashups aparece el concepto de programación ligera, ya que consiguen abstraer aspectos importantes de programación, haciendo este proceso más intuitivo para los usuarios finales. Según Pietschmann [1], el nivel de conocimiento sobre la tecnología subyacente para los procesos de composición centrados en el usuario final, debe ser cada vez menor y transparente.

Así, para alcanzar un proceso transparente para el usuario, Pietschmann establece dos aspectos importantes a la hora de componer recursos, como se observa en la Figura 2.1: *i*) la simplificación de la tecnología y *ii*) el soporte para la reutilización de componentes y conocimiento. El primero, busca medios por los cuales el usuario genere sus aplicaciones sin tener conocimiento de desarrollo software. El segundo, busca que cada vez que el usuario desee generar una aplicación nueva, él pueda reusar tanto de componentes de alto nivel de implementación (ej., lector de RSS o filtros de datos, etc.), como de conocimiento de dominio[1] y de composición[2] en otras palabras, reutilizar las recomendaciones de patrones de composición desempeñadas exitosamente por aplicaciones previas. En consecuencia, estos aspectos marcan una pauta importante a la hora de implementar herramientas que permitan a usuarios sin conocimientos técnicos ejecutar actividades de composición.

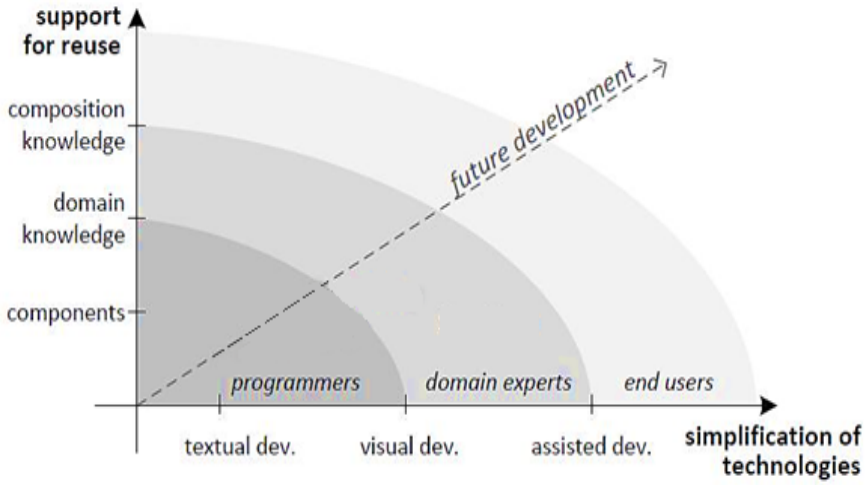


Figura 2.1: Relación de enfoques de composición [1].

[1] **Conocimiento de dominio específico:** Es el conocimiento acerca del entorno en el cual opera un determinado sistema [22].

[2] **Conocimiento de composición:** Es el conocimiento suministrado por una comunidad de desarrolladores, para resolver problemas específicos de modelado/composición a través de diferentes tipos de patrones [23].

Teniendo en cuenta los criterios anteriores, el proceso de generación de mashups puede considerarse como uno de los principales criterios a la hora de clasificar el tipo de herramienta, dado que varias de ellas consiguen generar un mismo tipo de mashup. No obstante, el esfuerzo de un usuario puede cambiar dependiendo del nivel de utilidades que provea la herramienta para facilitar este proceso, así como, la flexibilidad para que el resultado pueda ser modificado eventualmente [24].

La clasificación de herramientas al igual que los tipos de mashups, no se encuentra estandarizada. Por ejemplo, mientras que Hoyer et. al. [17] define un modelo basado en 3 criterios de evaluación: tipo de información del proveedor de servicios junto con el tipo de usuario a quien se dirige el proceso de generación, la funcionalidad y la usabilidad de dichas herramientas. Pientschmann [1] abstrae la clasificación a través de otros aspectos que permiten comparar sistemáticamente los diversos enfoques, tales como: el modelo de componente, el modelo de composición, y los entornos de desarrollo y ejecución.

Por otra parte, existen investigaciones orientadas a tratar criterios de evaluación específicos. Por ejemplo, Fischer [25] clasifica las herramientas de generación de acuerdo a la técnica de programación optada. Mientras Florian et. al. [19] expone una categorización que se encuentra enfocada en el dominio de aplicaciones empresariales. Por último, Koschmider et. al. [26] propone una clasificación orientada a resolver los problemas de implementación en tiempo de ejecución.

Añadiendo a lo anterior, la mayoría de las herramientas que generan mashups llevan a cabo su proceso de generación en dos etapas [27]. La primera, es un entorno de diseño donde es modelado el mashup, es decir, el usuario realiza la integración de los componentes, que representan los recursos y/o servicios que constituyen el mashup por medio de diferentes paradigmas, tales como arrastrar y soltar (*Drag&Drop*) dichos componentes, entre otros. La segunda etapa, es el entorno de ejecución que proporciona los servicios para ejecutar los procesos generados en la etapa previa —compilación y ejecución del modelo del mashup generado—, dando como resultado el despliegue del mashup final.

A continuación se describen los procesos más utilizados por las herramientas en la etapa de diseño, denominados *Paradigmas*. Posteriormente, se estudian los tipos de generación de mashups teniendo en cuenta estos paradigmas.

2.2.2.1 Paradigmas de Diseño

En la etapa de diseño se destaca la diversidad de paradigmas de generación de mashups, basados en lenguajes de programación textual y visual[3]; métodos por los cuales una persona puede romper las barreras de programación que le impiden desarrollar mashups [28]. Entre los paradigmas más relevantes para este trabajo de grado se encuentran:

- **Lenguaje de Dominio Especifico conocido como DSL (por sus siglas en inglés, *DomainSpecificLanguage*):**

Lenguaje de programación creado a partir de un lenguaje genérico (por ejemplo, JavaScript o Ruby), que permite introducir sintaxis textual con el objetivo de utilizar términos propios de la composición y creación de componentes para Mashups[10]. Este paradigma se centra exclusivamen-

[3]**Programación Visual:** Es un lenguaje o estilo de composición que intenta a través de símbolos generar un flujo de datos, sin la necesidad de crear líneas de código. Estos símbolos abstraen a un alto nivel todos los procesos y funcionalidades que definen la conexión, orden y filtro de los datos dentro del mashup.

te en programadores. Además, las herramientas que hacen uso de este paradigma, proporcionan al programador habilidades y capacidades necesarias para componer diversos tipos de recursos [29].

Uno de los principales beneficios que exhibe este paradigma, es proporcionar flexibilidad y poder de expresión en el proceso de generación de mashups, similar a un lenguaje genérico [30]. De esta manera, los usuarios pueden crear y componer componentes altamente desacoplados y con capacidad de personalización, tanto como lo permita el lenguaje y las capacidades del desarrollador.

Entre las herramientas que emplean este paradigma, se encuentra: MU [29], es un lenguaje de programación de alto nivel, lógico y funcional destinado a realizar mashups Web. Swashup [31], es un DSL textual para construir mashup sobre el framework de RubyOnRails, facilitando la composición de servicios en las aplicaciones Web.

- **Piping&Wiring:**

Proporciona un estilo de composición visual, que abstrae los recursos como componentes de alto nivel, también llamados Widgets o Bloques de Construcción (Building Blocks), los cuales son arrastrados a un lienzo donde sus entradas y salidas son conectadas, propiciando un flujo de datos [32, 19].

Cada componente contiene uno o varios recursos, los cuales necesitan ser conectados entre sí, de tal manera que un flujo de datos sea definido dentro del componente, esto es realizado por medio de los parámetros del componente, proceso que se denomina *Piping*. Por otro lado, el flujo de los datos entre los componentes, corresponde a *Wiring*.

- **Paradigma conocido como WYSIWYG (por sus siglas en inglés, *WhatYouSeeisWhatYouGet*):**

Lenguaje de programación visual, que permite diseñar y componer al mismo tiempo el mashup, de manera que lo que el usuario está visualizando es prácticamente su resultado final [33, 1]. Por consiguiente, dado que los usuarios siempre ven el mashup resultante, todo el proceso de desarrollo es más fácil de entender.

Entre las herramientas que emplean este paradigma, se encuentran: ServFaceBuilder [34], una herramienta que permita la composición de servicios en la capa de presentación mediante la combinación de interfaz de servicios y de anotaciones.

- **Programación por Demostración conocido como PbD (por sus siglas en inglés, *ProgrammingbyDemostration*):**

Permite mediante ejemplos instruirle al sistema cómo debe llevar a cabo una tarea específica [30], eliminando las barreras de programación, de manera que los usuarios no tienen que desarrollar, ni entender conceptos de programación, enfocándose únicamente en resolver los problemas de composición durante la construcción del mashup, proporcionando tan solo los ejemplos al sistema.

No obstante, este paradigma se orienta a usuarios con cierto nivel de experiencia, ya que éstos deben conocer la utilidad y funcionamiento de los recursos y las operaciones que conforman la herramienta.

Por ejemplo, en [35] proponen a Karma, una herramienta que permite a sus usuarios crear mashups, proporcionando ejemplos de cómo debe lucir el resultado final de cada operación, estos ejemplos representan la integración de los datos provenientes de diversos recursos Web.

- **Hojas de Cálculo:**

Por medio de este tipo de interacción se le permite al usuario almacenar, manipular y mostrar información tal como en las hojas de cálculo convencional, así como, proporcionar acceso a software para aplicaciones de negocios, bases de datos y archivos planos [27, 36]. Sin embargo, este paradigma no puede ser utilizado para diseñar la interfaz de usuario de un mashup. Por ejemplo, Husky [30].

- **Composición en Vivo (Live Composition o LivenessLevel 4):**

Lenguaje de programación visual que tiene como objetivo proporcionar un entorno de desarrollo para la creación de mashups en tiempo de ejecución; es decir, permite durante ejecución realizar y observar cambios inmediatamente en los componentes del mashup, sin la necesidad de que un usuario tenga que ejecutar un nuevo proceso con dichos cambios [30, 37]. Por lo tanto, el ciclo de desarrollo es rápido al combinar los entornos de diseño y de ejecución.

- **Generación mediante Lenguaje Natural:**

Combina paradigmas de lenguaje de programación visual con el lenguaje natural. Tiene como objetivo determinar el deseo de búsqueda de un usuario a través de una petición, en lenguaje natural y en tiempo de ejecución [7]. Por lo tanto, este paradigma permite simplificar la curva de aprendizaje centrado en el usuario final, al emplear el lenguaje natural como una de sus principales ventajas [21].

Entre las herramientas que emplean este paradigma, se encuentra: NaturalMash [21], una herramienta que hace uso de un lenguaje controlado, de manera que la estructura de las frases debe proporcionar un orden lógico de composición, para que el mashup sea ejecutable y expresar la búsqueda del usuario.

Generalmente, las herramientas de mashup hacen uso de uno o de la combinación de varios de estos paradigmas mencionados para ofrecer una mejor experiencia de desarrollo en la generación de mashups. Entre las combinaciones más populares se encuentran: WYSIWYG y Piping&Wiring.

Sin embargo, una de las limitaciones que ostentan los paradigmas de programación textual o visual, además de las herramientas que utilizan programación por demostración, es la falta de adaptabilidad; es decir, si las fuentes de información cambian su estructura o comportamiento, el mashup tiene que ser rediseñado, lo cual implica que se requiere conocimientos en programación [38].

Es así, como en la etapa de diseño, concretamente en los procesos de composición en la generación de mashups, se plantean retos en la creación de herramientas de desarrollo encaminadas a reducir las barreras de concepción de un mashup; es decir, la creación o adaptación de paradigmas que permitan automatizar este proceso, de tal manera que sea transparente para el usuario [28].

2.2.2.2 Tipos de Generación

Como contexto de esta investigación, a continuación se exponen los cuatro tipos de generación de mashups, clasificación que se define con base en los paradigmas de diseño estudiados:

- **Generación por código (C):**

Integra diversos recursos mediante DSL o un tipo de lenguaje genérico. Este proceso se lleva a cabo por usuarios expertos (por lo general, desarrolladores), [10, 18] quienes se encargan de implementar los componentes, así como de acoplarlos y de ajustar sus parámetros.

- **Generación Semiautomática (Sa):**

Este proceso de creación se orienta en automatizar ciertas tareas, como: la recuperación, selección y adaptación de recursos heterogéneos, representados por componentes conocidos como Bloques de Construcción (*Building Blocks*). Sin embargo, es necesaria la intervención de un usuario experto, capaz de establecer los flujos de control del mashup para su adecuado funcionamiento [28]. Entre los paradigmas usados, se encuentra los lenguajes visuales, tales como Piping&Wiring, PbD y WYSIWYG.

- **Generación Semiautomática-Asistida (SA):**

Este proceso automatiza parcialmente actividades propias de la composición; es decir, selecciona, adapta y ensambla componentes; también proporciona asistencia y orientación en el proceso de desarrollo, a través de actividades de recomendación de componentes y patrones de composición empleados en otros mashups. Sin embargo, las personas intervienen en el establecimiento de los flujos de control y datos [23, 30]. Al igual que el Semiautomático, usa los paradigmas de generación basados en lenguajes visuales y de lenguaje natural.

- **Generación Automática (A):**

Este enfoque está centrado en el usuario final, cuyo propósito se orienta a automatizar el proceso de desarrollo de mashups, el cual consiste en buscar, agregar, integrar y conectar —de manera automática y totalmente transparente para el usuario final— diferentes componentes, los cuales son desplegados y reorganizados según la interacción o comportamiento del usuario con el sistema [25]. Así, los entornos de diseño y de ejecución del mashup, se encuentran unificados.

Este proceso de generación emplea paradigmas de composición en vivo, donde el usuario no tiene que preocuparse por intervenir en el flujo de control, en su lugar, el usuario se ocupa del resultado deseado.

A su vez, la categorización expuesta se evalúa en función de un conjunto de criterios, los cuales se detallan a continuación:

- **Flujo de control:**

Criterio contenido dentro de la lógica de composición, que hace referencia al orden en que son llamados los conjuntos de instrucciones o tareas computacionales que posibilitan la ejecución apropiada del mashup. Este criterio de evaluación contempla dos estados: *Implícito* —las tareas son realizada automáticamente por el sistema— y *Explicito* —un usuario debe ejecutar manualmente las tareas— [30].

- **Tipo de contenido:**

Corresponde a lo que se despliega en la interfaz de usuario como resultado final del mashup. Contempla las siguientes clasificaciones: *Dinámico* —respuesta de los componentes a eventos en tiempo de ejecución, estos eventos modifican el mashup sin la necesidad de ser desplegado nuevamente— y *Estático* —no hay una respuesta a eventos que modifiquen el mashup en tiempo de ejecución—.

- **Tipo de creador:**

Tipo de usuario que realiza el proceso de generación del mashup. El cual puede ser catalogado como: *Desarrollador*, *Prosumer* —productor y consumidor— y *Usuario Final*.

Con base en lo anterior, el Cuadro 2.1 presenta un resumen de la clasificación mencionada:

Cuadro 2.1: Tipo de generación de mashups

Tipo de generación	Referencias	Flujo de Control	Tipo de Contenido	Tipo de Creador
C	DSL	Explícito	Estático Dinámico	Desarrollador
Sa	Lenguaje visual	Explícito	Estático	Prosumer
SA	Lenguaje visual	Explícito	Estático	Prosumer
A	Lenguaje visual	Implícito	Dinámico	Usuario final

Del Cuadro 2.1 se destaca que los procesos de generación semiautomático y semiautomático-asistido, aunque tratan de hacer amigable la integración de componentes en los mashups, no consideran que los usuarios finales no están interesados en realizar actividades de composición, como: seleccionar recursos de la web y definir los flujos de datos entre los componentes [7], por consiguiente, es poco probable que usuarios finales hagan uso de ellas.

De igual forma, las herramientas que proporcionan su proceso de generación por código, no son usados por usuarios finales. No obstante, este tipo de generación pueden ser utilizado como base para desarrollar nuevas herramientas, que brinde un nivel de abstracción más alto, de tal manera que su proceso de generación pueda convertirse en procesos Semiautomático o Automático.

Con base en el panorama descrito, el presente trabajo de grado se centrará en definir un mecanismo que permita generar mashups de cliente de forma automática. Este mecanismo debe considerar tanto la automatización de los flujos de datos y de control, la generación de interfaces graficas, además de la administración de los estados, la sincronización y deposición de los componentes, así como la selección e integración automática de nuevos componentes en tiempo de ejecución.

Por otro lado, la necesidad de satisfacer una tarea de búsqueda de la vida cotidiana por medio de un mashup, puede presentarse en cualquier momento y espacio donde el usuario no se encuentre frente a un computador portátil o de escritorio, siendo su dispositivo móvil su única opción, razón por la cual este dispositivo se convierte en la plataforma por excelencia para generar mashups. Teniendo en cuenta lo anterior, la siguiente sección describe el estado actual de los dispositivos móviles frente a los mashups.

2.2.3 Mashups Móviles

Con el desarrollo de nuevas tecnologías móviles y frente a la masificación de estos dispositivos, los usuarios se ven motivados a realizar sus actividades diarias en ellos. Como resultado de este crecimiento, estudios ostentan el valor promedio de la utilización de los dispositivos en diversos lugares, tales como: en el hogar (96 %), el trabajo (69 %), al movilizarse (83 %), en tiendas (76 %) y restaurantes (70 %), entre otros.

No obstante, aunque existe un panorama prominente en las herramientas de generación de mashups a nivel móvil, solo son utilizados en la etapa de despliegue del mashup final, pero no

se realiza el proceso de generación. Esto se debe a que las herramientas no están diseñadas para adaptarse a las características limitantes de los dispositivos, como el tamaño de pantalla y la fragmentación de la plataforma [39].

Hasta el momento, existen investigaciones que proponen arquitecturas basadas en los principios de diseño de SOA . Sin embargo, estas arquitecturas aun se encuentran a un alto nivel, debido a la falta de un diseño de arquitectura solida, que los respalde. Un ejemplo, se muestra en [39], donde proponen Mashup Web Móviles a través de la utilización de servicios y descomposición de procesos de negocio más pequeños, que permitan aligerar el despliegue de ellos. Cabe anotar, que una de las desventajas presentadas por la propuesta, es a nivel de los servicios que componen los mashups, debido a que cuando uno falla se entrega un resultado incompleto

2.3 Estado del Arte

A continuación se describen aquellas soluciones que hasta la fecha contribuyen significativamente al presente trabajo. Para tal fin se han organizado de acuerdo a la clasificación descrita en la Sección 2.1:

2.3.1 Generación semiautomática-asistida de Mashups

- **MashupAdvisor: A Recommendation Tool for Mashup Development [40]:**

MashupAdvisor es una herramienta que proporciona asistencia en tiempo de diseño al usuario final, ejecutando recomendaciones con base en el estado actual del mashup y un repositorio de composición desarrollado por una comunidad de usuarios. Este proceso se lleva a cabo en dos pasos: *i*) se extraen las recomendaciones que se pueden agregar al mashup, basados en un modelo probabilístico que clasifica las salidas en función de su popularidad y con base en los recursos utilizados en el mashup actual. *ii*) Cuando una de las salidas es aceptada por el usuario, MashupAdvisor automáticamente selecciona y configura los recursos y componentes necesarios para producir el resultado deseado en el mashup.

- **Hosted Universal Composition Models, Languages and Infrastructure in mashArt [32]:**

MashArt propone un paradigma llamado Integración Universal, cuyo propósito es combinar colectivamente a nivel de capa de datos, de control y de interfaz de usuario, los diferentes elementos que se desean agregar en el mashup. MashArt abstrae la composición de recursos heterogéneos usando un Modelo de Composición Universal, bajo el paradigma Drag&Drop y Pipering&Wiring. Además, proporciona vistas predefinidas para pre-visualizar y probar la aplicación, antes de ser desplegada; permitiendo ahorrar tiempo en la generación manual.

- **Overview of an end-user enabled model-driven development approach for interactive applications based on annotated services [33]:**

Este proyecto propone a ServFace, una herramienta de composición de aplicaciones basadas en servicios por medio de anotaciones, que contienen información para generar una interfaz de usuario para cada servicio (denominados ServiceFron-ends), similar a los formularios tradicionales de la Web, los cuales son empleados por ServFaceBuilder, un módulo que despliega los Service Front-ends y los interconecta por medio de acciones, tales como Drag&Drop y Pipering&Wiring. Este

mashup parcial es validado por CAM (*CompositeApplicationModel*). Este enfoque se basa en el paradigma WYSIWYG, puesto que le permite al usuario visualizar la aplicación final durante su creación.

- **Baya: assisted mashup development as a service [41]:**

Este proyecto permite mejorar las capacidades de una herramienta, al proporcionar asistencia a un usuario en especial a desarrolladores novatos, recomendando patrones de composición, que son extraídos de un repositorio de conocimientos sobre la herramienta integrada.

Baya es instalado como un "*plug-in*" de Firefox, que orienta el desarrollo de forma interactiva recomendando patrones de composición. Esta propuesta se fundamenta en la similitud sintáctica entre salidas y entradas de los servicios, con el fin de interconectar automáticamente los componentes agregados en el lienzo de composición.

2.3.2 Generación automática de Mashups

- **Wishfull Search: Interactive Composition of Data Mashups [13]:**

Este trabajo propone a MARIO (*MashupAutomationwithRuntimeOrchestration andInvocation*), una herramienta orientada a usuarios finales, que compone en tiempo de ejecución mashups de datos. Mario recibe consultas basadas en etiquetas, que describen un flujo de salida relevante, frente a un número de flujos de composición alternativos que los usuarios pueden visualizar y desplegar con el fin de refinar su tarea de búsqueda de manera iterativa.

- **Natural End-User Development of Web Mashups [21]:**

Este trabajo propone a NaturalMash, un sistema que permite a los usuarios finales mediante lógica computacional, desarrollar mashups en tiempo de ejecución, al automatizar las etapas de edición, compilación y despliegue de la aplicación por medio de cuatro elementos principales: El primero, un campo visual, donde se visualiza el resultado; el segundo, un campo textual, donde se describe textualmente la consulta en lenguaje natural; y tercero, un lienzo, donde se lista los componentes utilizados en el mashup. NaturalMash combina varios de los paradigmas mencionados, tales como lenguaje natural, WYSIWYG, programación en vivo y por Demostración.

2.4 Brechas Existentes

A continuación se detallan los aportes y las brechas de los trabajos relacionados estudiados en la sección anterior.

Cuadro 2.2: Aportes y brehas de trabajos relacionados

Articulo	Aportes	Brechas
MashupAdvisor: A Recommendation Tool for Mashup Development	• Un modelo de servicios semántico basado en tags para describir tanto los servicios como las propiedades de sus entradas y salidas. • Un mecanismo de planificación para recuperar composiciones relevantes acordes a los objetivos seleccionados por el usuario.	• Soporte limitado de recursos, debido a que trabaja únicamente con servicios web. • No está orientado a usuarios finales, puesto que se basa en el paradigma Drag&Drop. • No permite modificar la interfaz final del mashup una vez generado; para lograrlo, es necesario comenzar un nuevo flujo de composición.
Wishful Search: Interactive Composition of Data Mashups	• Modelo de metadatos para describir la semántica de servicios, de feeds y de flujos que permiten una composición automática, mediante el uso de etiquetas, folcsonomías y taxonomías. • Una interfaz adaptable a los cambios generados en la composición de un flujo de servicios y feeds, proporcionando una respuesta inmediata al usuario en cuanto a la información presentada y los flujos alternativos.	• Soporte limitado de recursos heterogéneos, debido a que se orienta únicamente a servicios web y datos. • Las composiciones generadas corresponden a mashups estáticos de datos, los cuales no varían con la interacción del usuario

Hosted Universal Composition Models, Languages and Infrastructure in mashArt	• Modelo de componentes Unificado, para adaptar y abstraer componentes tanto de UI como componentes de la aplicación (SOAP o RESTfullservices) y componentes de datos (feeds o XML). • Lógica de composición basada en eventos. • Modelo de Composición Universal, que permite que el usuario pueda componer en capa de presentación, sin la necesidad de sincronizar la UI y la orquestación del servicio, ya que lo hace automáticamente.	• No se enfoca en el usuario final. • Debido al paradigma de composición usado en esta aproximación (Drag&Drop), a medida que se incrementa la complejidad del mashup en desarrollo, se presentan problemas de legibilidad, causado por el gran número de conexiones entre componentes que constituyen el mashup.
Overview of an end-user enabled model-driven development approach for interactive applications based on annotated services	• Generación automática de interfaces para servicios SOAP por medio de anotaciones. • Paradigma WYSIWYG, que permite a usuarios sin conocimientos en programación, componer gráficamente, y a su vez, visualizar una aproximación final de las aplicaciones resultantes.	• Orientado únicamente a servicios descritos por WSDL. • Limitado número de recursos visuales en el mashup, dado que no soporta modificaciones en tiempo de ejecución.
Baya: assisted Mashup development as a service	• Un enfoque basado en minería de datos para reutilizar el conocimiento encontrado en previas composiciones y la definición de patrones que permiten organizar dicho conocimiento.	• Depende de las capacidades de las herramientas con las que se integrar para definir los tipos de recursos que utilizará y el tipo de mashup que realizará.

Natural End-User Development of Web Mashups	• Unificación del entono de desarrollo y entorno de ejecución a través del paradigma Live Programming. • Simplifica la curva de aprendizaje en la composición, al utilizar el paradigma de lenguaje natural. • Permite que usuarios expertos y/o programadores, construyan librerías para integrar nuevos recursos.	• Es necesario que los usuarios tengan conocimiento en composición, esto para la orquestación y correcta integración de los recursos en la web.

El Cuadro 2.2 evidencia la carencia o cumplimiento de los criterios descritos en la Sección 2.2. De esta forma, se ha definido un esquema de evaluación como el observado en el Cuadro 2.3, el cual se puede visualizar y analizar con mayor facilidad las brechas existentes y por ende, las ventajas del enfoque propuesto. Para esto, se consideran cinco aspectos importantes, que son: Simplificación de la Herramienta (SH), es decir, si posee los mecanismos que permitan ofrecer facilidades al usuario al momento de crear mashups; Intervención del Usuario con el Sistema (IUS); Integración de Diversos Recursos de la Web (IDR); Sincronización de Componentes (SC) y por último; Contenido Iterativo (CI), el cual indica la capacidad del sistema para agregar y eliminar contenido sin afectar la composición durante la ejecución del mashup final.

Cada uno de estos aspectos, se evaluaran haciendo uso de 4 niveles: Alto (A), Medio (M), Bajo (B) y Ausente (Au); que sintetizan el estado de los proyectos expuestos en esta sección, tal como se muestra en el Cuadro 2.3.

Cuadro 2.3: Comparación trabajos relacionados

Enfoque	**SH**	**IUS**	**IDR**	**SC**	CI
MashupAdvisor	M	A	B	B	B
Mario	A	B	B	M	M
MashArt	M	M	A	A	B
ServFace	M	M	B	A	M
iMashup	A	B	M	A	M
Baya	M	M	Au	Au	B
NaturalMash	M	B	A*	A	A

A partir de las aproximaciones estudiadas en los Cuadro 2.2 y 2.3, es evidente que el nivel de conocimiento necesario para que una persona genere sus propios mashups es aún un inconveniente, puesto que requieren ejecutar actividades como: selección manual de las fuentes de información, conocer la estructura y los procesos importantes sobre la herramienta utilizada, además de establecer los flujos de control y datos en un entorno de diseño.

Por otra parte, las herramientas compuestas por dos entornos de desarrollo presentan una limitación notable, puesto que cada vez que se desee realizar un cambio en los resultados es

necesario volver a diseñar y ejecutar el mashup, actividades no apropiadas para usuarios finales sin conocimientos técnicos [25]. Razón por la cual, se evidencia una necesidad latente por herramientas que simplifiquen este proceso.

Aún con los esfuerzos realizados por parte de los trabajos en mención, la generación de mashups centrada en el usuario final, presenta diversos inconvenientes, entre los que se encuentran: proporcionar una mayor simplicidad de la herramienta subyacente para soportar el desarrollo de mashups por parte del usuario final, reducir la intervención del usuario en actividades técnicas, y facilitar la integración de recursos heterogéneos sin afectar la generación de mashup.

Por lo tanto, con base en el panorama descrito en este capítulo, la presente propuesta de investigación pretende brindar un balance entre la complejidad de la tecnología subyacente durante los procesos de integración y el nivel cognitivo de los usuarios finales para la creación de mashups en tiempo de ejecución. De manera tal, que los usuarios sin conocimientos técnicos se enfoquen en generar mashups que resuelvan sus necesidades de búsqueda y no en tareas técnicas, que los distraigan de sus objetivos reales.

2.5 Resumen

Este capítulo presentó el estado actual sobre investigaciones desarrolladas alrededor de la generación de mashups centrados en los usuarios. Primero, el lector es guiado a través de la definición formal de los conceptos elementales que soportan el presente proyecto, con el fin de evidenciar y comprender el contexto general de investigación. Posteriormente, son descritos los diferentes trabajos relacionados, organizados en dos enfoques que representan el contexto de investigación. Posteriormente, un esquema de evaluación fue definido con el ánimo de identificar brechas y aportes de los trabajos relacionados. Finalmente, se expone un análisis de los enfoques estudiados, con el ánimo de posicionar la propuesta expuesta en este trabajo monográfico.

Capítulo 3

Modelo de Mashups

3.1 Introducción

Emplear al usuario final para realizar tareas de composición a partir de recursos diversos es un problema complejo. Para lograrlo, el enfoque propuesto en este trabajo aborda tres tareas importantes: *i*) la selección de los recursos provenientes de diferentes fuentes de información; *ii*) la integración de dichos recursos, es decir, definir el flujo[1] y mapeo de los datos; y *iii*) la generación de interfaces graficas que permitan visualizar la información de forma adecuada. Así, el desafío de este proyecto radica en hacer que estas tareas sean efectuadas de forma transparente para el usuario, permitiéndole enfocarse únicamente en el cumplimiento de sus objetivos de búsqueda a través de mashups.

El presente capítulo tiene como objetivo describir el enfoque propuesto para generar de manera automática mashups de clientes. En primera instancia, se describe el modelo que soporta la solución propuesta para generar mashups de manera automática a partir de la interacción del usuario final (Sección 3.2). Seguidamente, se describen los procesos llevados a cabo en la recuperación y selección de componentes (Sección 3.3).

Los aportes principales de este capítulo son:

- Un modelo de mashups.
- Un algoritmo para generar mashups con base en el comportamiento del usuario durante su interacción con el sistema.

3.2 Formalización de la Generación de Mashups

En esta sección se exponen los aspectos que posibilitan automatizar la generación de mashups. Por lo general, las herramientas disponibles actualmente proporcionan interfaces gráficas de composición de mashups, que a través de paradigmas de programación visual, ocultan en cierta medida la complejidad del desarrollo de mashups mediante la integración manual de componentes. En consecuencia, el presente proyecto evita este proceso de integración manual, gracias a la adaptación y estandarización de los componentes, que definen tanto la generación correcta del mashup como la comunicación entre componentes. Adicionalmente, se asiste al usuario en el proceso de selección

[1] **Flujo de Datos:** Es un puente por el cual fluye información entre dos elementos de un sistema/aplicación.

de nuevos componentes por medio de descubrimiento de patrones que indican la co-ocurrencia de componentes, tomando como base las interacciones del usuario y los objetivos actuales del mashup. Teniendo en cuenta lo anterior, las siguientes secciones describen el modelo de componente y el modelo de mashups. Posteriormente, son descritos los patrones de coocurrencia de componentes.

3.2.1 Modelo de Componente

Los componentes son abstraídos y adaptados a partir de recursos disponibles en la Web, representando así el núcleo de los mashups, dado que es a través de la utilización de estos que los usuarios los generan. Es importante destacar que estos elementos al ser reutilizables e independientes, pueden ser fácilmente empleados en diversos Mashups. Estos componentes se usan para soportar el manejo de datos, la lógica de negocio y las interfaces de usuario, que constituyen al mashup [42]. A continuación, se define el concepto recurso, el cual corresponde al insumo principal de un componente:

Definición 3.1. *Recurso (Recurso Web)*

> Un recurso representa la abstracción clave de la información. Así un recurso puede ser: un documento, una imagen, un video, un servicio, una colección de otros recursos, entre otros [43].

Por ejemplo, entre los tipos de componente utilizados en investigaciones relacionadas se encuentran: Servface [33] el cual toma como componentes a servicios SOAP donde su interfaz gráfica se visualiza por medio de formularios; en Intel Mash Maker [44] los componentes son representadas por las páginas web más visitadas por el usuario; en Yahoo! Pipes [45] los componentes utilizados son recursos de tipo RSS, Atom, Feeds; entre otros. Como se puede observar, cada herramienta define el tipo de componente a utilizar, ya sea a nivel de datos, lógica de la aplicación o capa de presentación, con el fin de abstraer sus resultados adecuadamente [28].

Tomando como precedente las herramientas descritas, MashArt [46] propone un modelo para integrar diversos recursos web a través de interfaces de usuario. Esta aproximación permite usar componentes en los diferentes niveles de la aplicación sobre un mismo entorno, facilitando su integración. El modelo propuesto por Florian es conocido como composición universal, el cual establece diferencias sustanciales respecto a la composición tradicional: *i*) la sincronización[2] debe ser adoptada como paradigma de integración, en lugar de la orquestación de servicios como lo hace BPEL (*BusinessProcessExecutionLanguage*); *ii*) los eventos deben reaccionar tanto a las entradas hechas por los usuario como a la interacciones programadas; *iii*) es posible diseñar las interfaces de la aplicación compuesta, y no solo el comportamiento y la interacción entre componentes. Esto demuestra la necesidad de generar un modelo de componente basado en estados, eventos y sincronización, en lugar de hacerlo en función de las llamadas a métodos y orquestación [2].

Teniendo en cuenta lo anterior, la Figura 3.1 representa el modelo de componente a nivel de datos, que permite abstraer la lógica de la aplicación y la interfaz de usuario. Este modelo considera cuatro tipos de abstracciones: *Interfaz de Usuario*, *Parámetros*, *Eventos* y *Operaciones*.

[2] **Sincronización:** Mecanismo que organiza las interfaces de usuario respecto al cambio realizado en la lógica de negocios y los datos, en tiempo de ejecución.

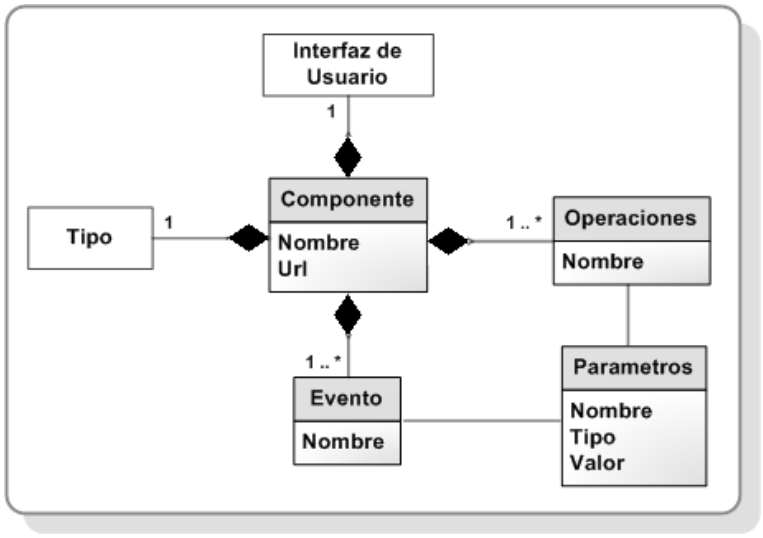


Figura 3.1: Modelo de componente [2].

A continuación, son descritos los elementos que conforman el modelo de componente acorde a [46, 2]:

Definición 3.2. *Interfaz de Usuario (UI)*

Es la representación gráfica de un componente y permite la interacción directa con el usuario. Se encarga de presentar el estado actual del componente, determinado por la configuración interna del mismo. Es así como en respuesta a las interacciones del usuario un componente puede cambiar su estado.

Definición 3.3. *Eventos*

Comunica a los componentes que conforman el mashup, los cambios de estado de un componente u otro tipo de información en el entorno, con el objetivo de mostrar datos útiles a otros componentes.

Definición 3.4. *Operaciones*

Son métodos invocados como resultado de los eventos. Estas operaciones pueden incluir funcionalidades hechas por un desarrollador de componentes, entre ellos: cambiar los estados y/o la configuración de un componente, cálculos e incluso la generación de nuevos eventos que llamen a otras operaciones. Es así como las operaciones permiten propagar la interacción del usuario desde un componente a otro, proporcionando la sincronización entre los estados de componentes (de esta manera, por ejemplo, ellos pueden mostrar información relacionada).

Definición 3.5. *Parámetros*

> Contiene los datos de configuración de un componente. Esta información es relevante puesto que permite re-ensamblar aspectos de la lógica e interfaz gráfica de un componente. Por ejemplo, existen parámetros en la interfaz de usuario que pueden ser utilizados en el diseño (Ejemplo, distribución del componente), mientras que componentes de servicio pueden necesitar datos de configuración, como el nombre de usuario y la contraseña.

A partir del modelo anterior y de acuerdo al enfoque del presente trabajo, el desarrollo de un componente adecuado debe considerar las siguientes condiciones:

- Un componente debe ser realizado por desarrolladores expertos, dado que debe cumplir con los aspectos definidos en el modelo de componente.
- La complejidad de un componente debe residir internamente y visualizarse de forma sencilla para los usuarios finales, creadores de mashups.
- Un componente no debe comprometer el funcionamiento adecuado de otros en el momento de integrarse o extraerse de un mashup.
- Los componentes deben estar basados en lenguajes, tecnologías o protocolos de comunicación estándar, que permitan a un componente ser especificado por descriptores definidos (Ej., servicios SOAP son descritos por un WSDL).

De esta manera, para el enfoque propuesto, un componente debe generar tanto la lógica del negocio como la interfaz gráfica del mismo, es decir, cada modelo debe llevar a cabo estas dos tareas durante su implementación.

3.2.2 Modelo de Mashup

El *Modelo de Mashup* (MM) abstrae los elementos que necesita considerar un desarrollador (para el enfoque proyecto, el usuario final) y/o herramienta para la generación de mashups. Estos elementos identifican implícitamente el poder expresivo de la misma, es decir, la capacidad de las herramientas de generación de mashups para: integrar diferentes tipos de componentes, definir el flujo de los datos y visualizarlos gráficamente, entre otros. Además, este modelo permite abstraer detalles de implementación a un alto nivel, que se adapte a las necesidades de integración a partir de un dominio genérico o especifico [47].

Por un lado, un MM de dominio genérico proporciona una diversidad de aplicaciones que permiten dar solución a un alto rango de posibilidades, por ejemplo: buscar restaurantes, recopilar las ultimas noticias nacionales, etc. Por otro lado, un MM de dominio específico proporciona aplicaciones dirigidas a solucionar problemas de carácter particular y se concentra en tan solo uno de ellos, por ejemplo: un usuario desea tener información de cómo aprender a interpretar el Charango.

El modelo planteado en el presente proyecto de grado se fundamenta en la aproximación expuesta por Soi [48], quien plantea un modelo de mashup general. No obstante, el modelo definido a continuación, se enfoca en generar mashups que resuelven problemas de un conjunto de dominios

específicos, tales como tareas de la vida cotidiana (Mashups de Clientes - Sección 2.2.1.1). Este modelo por su parte da la posibilidad de integrar diversos recursos de la web, independientemente de los lenguajes y tecnologías empleados para su implementación; es decir, permite integrar recursos a nivel de capa de datos hasta la capa la presentación en un mismo entorno.

A continuación, la Figura 3.2 define los elementos relevantes propuestos en el modelo de mashups.

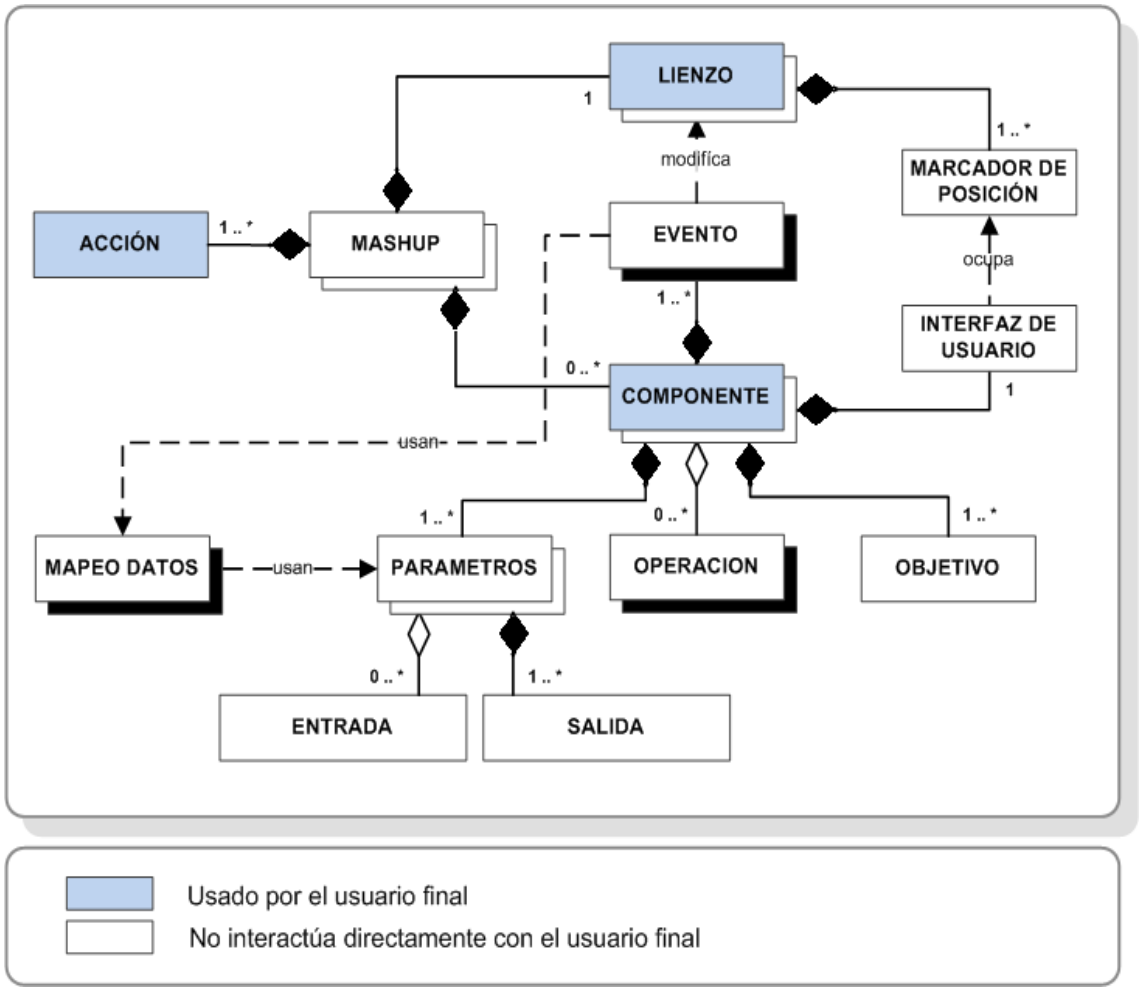


Figura 3.2: Modelo de mashup [Fuente Propia].

A partir del modelo ilustrado en la Figura 3.2, se extraen las siguientes definiciones:

Definición 3.6. *Mashup (M)*

Un mashup M centrado en las interacciones del usuario con el sistema, puede expresarse como $M =< nombre, C, L, A, ranking >$, donde: *nombre* es el nombre dado por el usuario para referirse a cada mashup, C representa el conjunto de componentes, L

define la disposición de los componentes en la interfaz gráfica del mashup –*Lienzo de integración*–, A representa el conjunto de acciones y *ranking* representa la calificación dada por el usuario al mashup.

Definición 3.7. *Objetivos (O)*

Describe un conjunto no vacío de etiquetas, correspondientes al propósito que puede satisfacer el contenido de un componente o mashup, expresado como $O = \{o_i\}_{1 \leq i \leq N}$.

Definición 3.8. *Acción (A)*

Representa la interacción del usuario con los componentes gráficos que constituyen el mashup y ejecuta los eventos que están relacionados al componente. Entre la acciones realizadas por el usuario se encuentran: compartir, abrir (observar componente), rechazar, llamar, deslizar contenido (galería de imágenes), enviar sms o correos, reproducir (visualizar videos), buscar ruta/camino (visualizar el camino para llegar al destino).

Definición 3.9. *Componente (c)*

Representa la abstracción de un recurso Web (Servicios, APIs, Datos, etc.) y se caracteriza por un nivel alto de heterogeneidad en términos de tecnologías y métodos de acceso; además, este es renderizado en una interfaz gráfica propia [28, 17]. Así, para su implementación se utilizan las especificaciones brindadas en el Modelo de Componente descrito en la Sección 3.2.1. De esta manera, un componente puede expresarse como:

$$c =< O, Ent, Sal, OP, E, UI, tipo, desc >$$

Donde: O representa el conjunto de objetivos que describen el propósito del contenido de un componente; Ent define un conjunto de entradas, $Ent = \{ent_i\}_{1 \leq i \leq N}$; Sal representa un conjunto de salidas, $Sal = \{sal_i\}_{1 \leq i \leq N}$; OP representa un conjunto de parámetros de configuración del componente, $OP = \{op_i\}_{1 \leq i \leq N}$ (Definición 3.4); E representa un conjunto de eventos (Definición 3.12); UI representa la interfaz gráfica del componente (Definición 3.2); *tipo* representa el tipo de contenido que puede visualizarse en la UI del componente y *desc* es el descriptor del componente que interactúa con el sistema (Definición 3.10).

Los componentes pueden ser catalogados según sus entradas y salidas en una de las siguientes categorías:

- Fuente: Componente que no tienen insumos, es decir, no existen parámetros de entrada pero tienen parámetros de salida.
- Sumidero: Componente que no tiene parámetros de salida pero si necesita parámetros de entrada.

- Típico: Componente que consume parámetros de entrada y producen parámetros de salida.

Definición 3.10. *Descriptor (desc - D)*

Describe cada componente, con el fin de brindar al sistema de información relevante para realizar la integración de componentes en el lienzo, tales como: el tipo, los atributos de entradas y salidas del componente. De esta manera, un componente puede expresarse como:

$$desc = \left\{ d_i \,|\, d_i \in DE\chi DS;\ DE = \bigcup_{ij} descEnt_{ij}\ ,\ DS = \bigcup_{ij} descSal_{ij} \right\}$$

Donde: DE y DS son conjuntos de descriptores de entradas y salidas de un conjunto de componentes C, definidos por:

$$DE = \{desEnt_i \,|\, desEnt_j \in tipo\_parametro\,\chi\, id\_parametro\}$$

$$DS = \{desSal_i \,|\, desSal_j \in tipo\_parametro\,\chi\, id\_parametro\}$$

Donde: *desSal* y *desEnt* son parejas ordenadas, resultado del producto cruz entre los tipos de parámetros y los identificadores del parámetros.

Definición 3.11. *Mapeo de Datos (MD)*

Representa el conjunto de las posibles conexiones entre los componentes, dado por la compatibilidad de los atributos de entradas con las salidas de otro componente. Así, dados los componentes *Cj* y *Ck*, el mapeo de las salidas de *Ck* con las entradas de *Cj*, se puede expresar como:

$$MD = \left\{ md_i \,|\, md_i \in Ent\chi Sal\ :\ Ent \bigcap Sal = \emptyset\ ,\ Ent = \bigcup_{ij} Ent_{ij}\ ,\ Sal = \bigcup_{ij} Sal_{ij} \right\}$$

Este mapeo de datos es generado automáticamente haciendo uso de los descriptores de los componentes (Definición 3.10) que son agregados al mashup.

Definición 3.12. *Evento (e)*

Extendiendo la definición 3.3, este expone los cambios de estado de uno o más componentes, generalmente definido por un desarrollador. Un evento puede ser inducido por la interacción del usuario a través de acciones sobre las interfaces de los componentes o su operación. Por ejemplo, al seleccionar un componente este puede contener información de localización, ejecutando un evento de posicionamiento en el mapa. Un evento puede expresarse como: $e =< nombre, D >$

Donde: *nombre* es el nombre que representa el evento; y D representa la tupla $<$ *parámetro*, *valor* $>$, que contienen los datos que se comparten entre los componentes

Definición 3.13. *Lienzo (L)*

Define la disposición de los componentes del mashup en la interfaz de usuario. Generalmente, la disposición de estos los elementos en un mashups está orientada a diseños genéricos, tales como Table- , List- Relative-, Linear- y Grid- Layout.

La sección siguiente contiene un análisis del modelo propuesto en el presente trabajo respecto a investigaciones relacionadas.

3.2.3 Análisis de los Modelos

Como se señaló en la sección anterior, el modelo presentado tiene como punto de referencia la propuesta de Soi [48]. No obstante, existen otros tipos de modelos de mashups, los cuales, a continuación serán contrastados con el propuesto en esta sección.

En [49] se expone un modelo semántico de servicios basado en etiquetas, el cual relaciona las entradas y salidas de sus servicios (componentes) por medio de la descripción de sus atributos, creando varios flujos de datos que son manipulados a través de un grafo acíclico dirigido. Una vez analizada la consulta del usuario, un planeador realiza el proceso de integración de los servicios basados en el grafo. De esta manera, el modelo semántico genera todas las soluciones relevantes de composiciones.

Sin embargo, el modelo propuesto en Liu, no permite agregar o remover componentes sino tan solo etiquetas, que describen una variedad de componentes no considerados por el usuario. Razón por la cual en el modelo propuesto mantiene uno de los principio expuestos en la Sección 3.2.1 "*Un componente no debe comprometer el funcionamiento adecuado de otros en el momento de integrarse o extraerse de un mashup*". Por ende, los componentes se encuentran altamente desacoplados entre sí.

Por otra parte, el modelo propuesto en Soi, caen en el error al considerar a los usuarios finales como una persona con conocimiento en composición; y en consecuencia, la definición de los conceptos del modelo exigen que el usuario se involucre directamente en procesos de composición. Es decir, este modelo define a los componentes como elementos, que pueden generar tanto la lógica de negocios como el flujo de datos y la interfaz gráfica por separado. Por lo tanto, el proceso de generación de mashup se realiza en dos etapas, como se mencionó en la Sección 2.2.2: la primera en un entorno de diseño, donde se integran los componentes y el segundo, un entorno de ejecución, donde se visualiza el resultado final de la integración.

Así, cada componente del modelo propuesto en la sección anterior, contiene los datos y la lógica necesaria para que este sea renderizado en su propia interfaz grafica, combinando los entorno de diseño y de ejecución. Además, de automatizar los procesos de integración permitiendo que usuarios finales realicen sus mashups en tiempo de ejecución, bajo los paradigmas WYSIWYG y Live Composition, Sección 2.2.2.1.

Por otra parte, el modelo propuesto en [50], los mashups son modelados como una agrupación de *Widgets*, los cuales son una extensión de la definición realizada por la *W3C*[3]. Esta extensión consiste en brindar a los Widgets, la capacidad de comunicar su estado interno por medio de

[3]**Disponible en:** http://dev.w3.org/2006/waf/widgets-land/

eventos a otros Widgets, característica similar a la definida en el enfoque del presente trabajo, a diferencia que el propuesto no está basado en la W3C. En cambio, el modelo planteado está basado en el proceso de *Integración Vertical*[4] de componentes, aproximación expuesta por Zhao en [20], de tal manera, que los flujos de datos son establecidos por los eventos que son disparados por las acciones del usuario u operaciones.

Tras el estudio de los modelos expuesto, el presente proyecto se destaca por definir un solo tipo de componente al abstraer las tres capas mencionadas anteriormente: lógica de negocios, flujo de datos e interfaz gráfica. A su vez, el modelo propuesto hace uso de los descriptores de componentes, los cuales, contienen la información necesaria para comunicar de manera automática los componentes. .Así, este enfoque permite agregar e integrar componentes sin afectar la generación del mashup, satisfaciendo las consultas de búsqueda interactivamente, proceso que se profundizará en la Sección 4.2.1.

3.3 Recomendación de Componentes

En esta sección se introducen los principales métodos empleados en la construcción de herramientas en la generación de mashups, con el fin de proporcionar el descubrimiento y recomendación de componentes, acorde a los requerimientos del usuario. A partir de estos métodos, son tomados los aspectos relevantes para el actual trabajo.

3.3.1 Recomendación Basado en la Semántica de los Parámetros de Entradas y Salidas (RDP)

Este enfoque provee recomendaciones cuando se produce un cambio en el mashup o por medio de las peticiones hechas por el usuario. El proceso de recomendación se realiza sobre la base de un modelo, bien sea probabilístico o semántico basado en etiquetas, que clasifica las salidas de un mashup en función de su popularidad y toma la similitud semántica, entre los nombres de los atributos para hacer las recomendaciones.

Para el modelo probabilístico expuesto en [40], un mashup es modelado como una composición de servicios –definida por el nombre, las entradas y salidas– y fuentes de información –entrada proporcionada por el usuario–. De esta manera, los atributos de entrada y salida empleados en las recomendación se denotan como "*concepto*". El modelo probabilístico determina las distribuciones de probabilidad de ocurrencia de los conceptos encontrados, en el repositorio que almacena los mashups previamente desarrollados y extrae los patrones de co-ocurrencia relevantes.

Para el modelo semántico de servicios basado en etiquetas utilizado en [49], se asocia un conjunto de etiquetas a los nombres de las entradas y salidas de los servicios (componentes), como anotaciones semánticas que describen las propiedades de los datos que el servicio puede consumir o producir.

Cuando el usuario desea nuevos componentes, debe expresar su objetivo a manera de etiquetas. Dado que los componentes han sido anotados previamente y clasificados bajo alguno de los grupo de etiquetas, el sistema se encarga de recuperar aquellos que responden a dichas etiquetas. Además,

[4] **Integración Vertical:** Hace referencia al proceso de integración de componentes expuesto en el tipo de mashup vertical [20], donde son agrupados los componentes similares o complementarios, presentando sus salidas gráficamente. La diferencia con este tipo de mashups, es que el modelo propuesto se define una comunicación entre componentes basada en eventos.

obtiene otro conjunto de etiquetas que puedan complementar las seleccionadas y que a su vez recuperan nuevos componentes.

Así, este enfoque se centra en el descubrimiento de nuevos componentes a partir de la similitud semántica entre los nombres de los parámetros o atributos (conceptos) de los servicios y fuentes de información.

3.3.2 Recomendación por Conocimiento de Composición (RCC)

Este enfoque considera las acciones, que son realizadas generalmente por un desarrollador en el lienzo de composición –entorno de modelado grafico– durante la etapa de diseño de la aplicación (donde son configurados y conectados los componentes, entre otras acciones). Estas acciones son representadas en diferentes clases de patrones, los cuales son llamados *Patrones de Composición*. En proyectos como MashArt [32], Baya [41] y Ommelete [51], se consideran los siguientes tipos de patrones:

- **Patrones de valor parámetro:** Este patrón representa un conjunto de asignación de valores recurrentes para los parámetros de entrada en un componente. Además, ayuda al llenado de parámetros de entrada para los componentes que requieren de una entrada en especifico.
- **Patrón de Conector:** Representa un conector recurrente entre pares de componentes. Además, Este patrón ayuda a conectar un componente recién colocado en lienzo de composición al modelo mashup parcial.
- **Patrón de Co-ocurrencia de Conectores:** Extrae un grupo de conectores que aparecen siempre en la composición. Además, este patrón incluye el mapeo de datos asociado a estos conectores, dado que los conectores informan que componentes están conectados, y que entradas y salidas están unidas.
- **Patrón de Co-ocurrencia de Componentes:** Al igual que el anterior patrón, este captura las parejas de componentes que aparecen frecuentemente. Además, asocia los componentes con conectores, parámetros y mapeo de los datos.
- **Patrón de Componente Embebido:** Captura que componentes típicamente están embebidos en otros componentes, ambos procesados por un tercer componente. Además, este patrón ayuda a modelar ciclos, que son una tarea no trivial para usuarios finales.
- **Patrón de Multi-componente:** Representa los fragmentos de modelos que son compuestos por un grupo de componentes. Además, este patrón captura patrones más complejos que los otros patrones no alcanza a descubrir.

Una vez descubiertos los patrones, estos son almacenados en una base de datos de conocimiento y son usados de forma accionable por el compositor en el lienzo de composición, es decir, los patrones son enviados de forma proactiva al lienzo sin esperar que el usuario haga la solicitud de ellos.

Así, el conocimiento en composición se centra particularmente en el descubrimiento, abstracción y reutilización, de conocimiento y composiciones existentes, que permiten que un usuario con pocas nociones de programación desarrolle una adecuada composición. Dado que, no es trivial ni intuitivo para personas del común saber que componentes son los indicados [11].

3.3.3 Análisis de los Procesos de Recomendación

En esta sección se analizaron dos de los principales enfoques, orientados a la recomendación de componentes en la generación de los mashups. Los enfoques RDP y RCC se destacan, al cambiar la tarea manual en la selección de componentes, en un proceso por poco transparente para el usuario compositor. Así, estos enfoques automatizan la tarea de recomendación de componentes, de tal forma que usuarios sin conocimientos técnicos, puedan adherir mas componentes al *Lienzo del Mashup* sin preocuparse por detalles de compatibilidad.

Por otra parte, es importante destacar que cada enfoque está relacionado estrechamente a un modelo de mashup. Donde, la selección de nuevos componentes es la asistencia que la herramienta brinda al compositor, para facilitar el proceso de generación de mashups, sea cualquier paradigma de diseño que se esté usando. Así, teniendo en cuenta el modelo propuesto en la Sección 3.2.2, el enfoque de recomendación Basado en el Conocimiento de Composición esta mas relacionado a la presente propuesta. Explícitamente, el patrón de co-ocurrencia de componentes es de especial interés para este proyecto, debido a que permite adherir nuevos componentes en el lienzo, a partir de la asociación de los componentes que aparecen más frecuentemente.

El resto de patrones de composición no son considerados en el enfoque propuesto, debido a que el modelo planteado, simplifica las tareas necesarias para la integración de componentes, dado que el usuario no tiene la necesidad de realizar tareas que demande la intervención en el proceso de integración, la cual se realiza automáticamente a través del Modelo de Integración propuesto en la Definición 4.1. A diferencia que las recomendaciones se harán cada vez que el usuario las solicite y no de forma proactiva.

A continuación, a partir de la caracterización del modelo de mashup propuesto se define el siguiente patrón de co-ocurrencia, que permite soportar el descubrimiento de componentes:

Definición 3.14. *Patrón de Co-Ocurrencia (PC)*

> Dado que cada componente se encuentra estrechamente relacionado con al menos un objetivo, este elemento permite estructurar los datos para generar las relaciones entre los componentes y los objetivos (además, entre los mismos objetivos). Así, un patrón de co-ocurrencia CP está dado por: $PC =< C, O >$
>
> Donde: C y O fueron definidas en el *MM*. Así, un patrón de coocurrencia es la representación de conocimiento útil que puede ser descubierto tanto de los mashups que están siendo creados como de aquellos mashups exitosos que han sido almacenados.

Así, dos patrones de co-ocurrencia útiles, que permiten descubrir componentes apropiados para el estado actual de un mashup, son:

Definición 3.15. *Patrón de Co-Ocurrencia de Objetivos (PCO)*

> Representa las parejas de Objetivos que aparecen frecuentemente.

$$PCO =< \emptyset, \{o_{xy}, o_{xy}\} >$$

Definición 3.16. *Patrón de Co-Ocurrencia de Objetivo-Componente (PCOC)*

Representa las parejas de objetivo-componente que aparecen frecuentemente.

$$PCOC =< \{c\}, \{o\} >$$

Por otra parte, los patrones que permite soportar la recomendación de componentes, son los siguientes:

Definición 3.17. *Recomendaciones de Componentes Basado en Objetivos Vecinos (ROV)*

Dado un componente el sistema sugiere uno nuevo por medio del PCOC. Es decir, es seleccionado el objetivo o_{xy} relacionado con el componente c_x y luego selecciona nuevo componente c_y con mayor probabilidad de co-ocurrencia con el objetivo o_{xy}.

$$ROV =< \{c_x, o_y\}, \{o_{xy}\} >$$

Definición 3.18. *Recomendaciones de Componentes Basado en Objetivos Relacionados (ROR)*

Dado los objetivos de un componente, el sistema sugiere un nuevo componente basándose en PCO y PCOC. Así, dado un objetivo o_x de un componente c_x, el sistema encuentra un objetivo o_y que esté relacionado con o_xy recupera el componente c_y.

$$ROR =< \{c_x, c_y\}, \{o_x, o_y\} >$$

3.4 Resumen

Este capítulo presentó los conceptos y enfoques, en los que se fundamenta este trabajo de investigación. Inicialmente, se abordó la descripción de un modelo que soporta la solución propuesta para generar mashups de manera automática centrado en el usuario final. De manera que cada elemento del modelo, cubre un aspecto específico en el proceso de desarrollo, de manera que permite abstraer detalles de implementación a un alto nivel. Seguidamente, se describen los procesos relevantes llevados a cabo en la recuperación y selección de componentes, en las herramientas para la generación de mashups, con el objetivo de asistir al usuario en los resultados de búsqueda y en la definición tanto de una generación correcta del mashup como la comunicación entre componentes.

Capítulo 4

Anatomía de la Solución

4.1 Introducción

Parte de la anatomía de la solución fue abordado en el Capítulo 3, donde se consiguió definir de manera formal: un modelo de Mashup que describe los detalles de implementación a un alto nivel; y Patrones de Conocimiento que exponen el descubrimiento y la recomendación de componentes, ambos abstraídos para generar mashups de manera automática, tomando como base el comportamiento del usuario y los objetivos actuales del mashup. En consecuencia, el presente capítulo complementa la anatomía de la solución. En primera instancia, la sección siguiente describe los procesos llevados a cabo en los enfoques propuestos. Por último, se presenta un estudio de caso con el fin de ejemplificar el enfoque propuesto.

4.2 Anatomía de la Solución

La Figura 4.1 ilustra la conceptualización de la solución propuesta para generar de manera automática mashups, con base en la interacción del usuario con el dispositivo móvil durante su tarea de búsqueda.

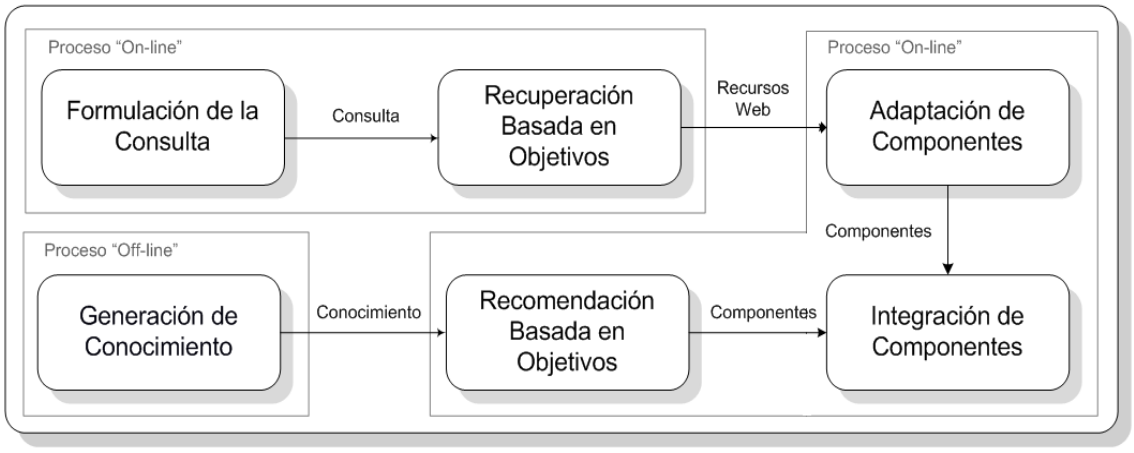


Figura 4.1: Representación Conceptual de la Solución [Fuente Propia].

Cuando un usuario formula su consulta q_0 al sistema, ésta es analizada con el fin de realizar el descubrimiento de recursos Web subyacentes. Una vez estos recursos son recuperados, se utilizan posteriormente para adaptar la información a manera de componente, que pueda ser interpretado por sistema. Sin embargo, para involucrar al usuario final en la generación de mashups, se requiere ejecutar previamente otros procesos, como: el análisis del conocimiento generado previamente (*off-line*), la recomendación de componentes en base a los eventos ejecutados por el usuario (*on-line*) y por último, la visualización de los resultados recuperados en el lienzo del mashups a partir de la interacción del usuario con el sistema (*on-line*). Así, el proceso "*on-line*" se refiere a que esta etapa genera una respuesta inmediata al involucrar al usuario final con el sistema, no obstante, el proceso "*off-line*" es un proceso que no involucra al usuario final con el sistema.

Los 3 procesos principales, que conforman la solución propuesta, son tratados con profundidad en la secciones siguientes.

4.2.1 Integración de Componentes

Tras el estudio de diferentes modelos existentes y con base al modelo propuesto en la Sección 3.2.2, este apartado presenta el proceso de integración de componentes en el mashup, a fin de eliminar la composición tradicional, y fusionar los entornos de diseño y de ejecución utilizados tradicionalmente.

Este modelo considera un paradigma diferente de desarrollo, que combina los paradigmas WYSIWYG y Live Composición (liveness nivel 4), Sección 2.2.2.1. Para facilitar el proceso de integración de componentes sobre el lienzo del mashup, sin la necesidad de definir otro entorno, donde se realice tanto la asignación de los flujos de datos y de control como la interfaz gráfica de cada componente. De esta manera, el desarrollo que permite generar automáticamente mashups, se denomina *Paradigma de Integración*.

Definición 4.1. *Paradigma de Integración*

> Lenguaje de programación visual centrado en el usuario final, cuyo objetivo busca proporcionar un solo entorno para el desarrollo y ejecución de mashups. Esto posibilita la integración transparente de componentes durante la tarea de búsqueda del usuario final, inhibiendo totalmente su participación en actividades de composición, y enfocándolo exclusivamente en información relevante para sus necesidades de información.

De esta manera, este paradigma se orienta a automatizar aquellas partes que el usuario final no está en capacidad de realizar, por ejemplo: definición de los flujo de datos y de control, mapeo de datos o asignación de eventos que permitan configurar los componentes por medio de las interacciones del usuario, generación y sincronización de los interfaces gráficas.

A continuación, el Algoritmo 1 describe el proceso que define el Paradigma de Integración:

Algoritmo 1 Proceso de integración.

Entradas:
C /*Conjunto de componentes recuperados*/
Salidas:
MP /*Según Definición 4.2*/
INICIO
$FD \leftarrow \{\}$ /*Inicialización de variable - Flujo de datos*/
$MD \leftarrow \{\}$ /*Inicialización de variable - Mapeo de datos*/
Si $MD \neq \oslash$ **Entonces:**
$MD \leftarrow MapearDatosPara(C)$ /*Paso (1)*/
Fin si
$SuscribirNotificacionesDeComponentes(C)$ /*Paso (2)*/
Si $FueEjecutadoUnEvento()$? **Entonces:**
$e \leftarrow ObtenerEventoEjecutado()$
$FD \leftarrow AsignarFDApartirDe(e,MD)$ /*Paso (6)*/
Fin si
$MP \leftarrow ReconfigurarYRenderizarComponentes(C,FD)$ /*Paso (3,4,5)*/
Retornar MP
FIN

Para proporcionar una visión general del proceso de integración de componentes propuesto, a continuación se explica el proceso en función de los conceptos expuestos en el modelo de mashup.

Asumiendo que se ha realizado una consulta inicial por el usuario final apartir de la cual se ha procedido a recuperar un grupo de recursos, los cuales han sido adaptados y descritos, ya sea por un desarrollador previamente ó de manera automática con base en las fuentes de donde provienen (Flickr, Youtube, Foursquare o Wikipeda)).

1. Se extrae cada descriptor y a partir de estos, se define el mapeo de datos para cada componente de C.

2. Cada componente es suscrito a una lista de notificaciones, encargada de publicar los eventos que ocurren en cada uno.

3. Luego, los componentes son configurados a partir de las propiedades que establecen su estado inicial, al igual que los datos para ser visualizados a través de sus GUI.

4. Después, cada componente con su interfaz gráfica es asignado a uno de los Marcador de Posición del Lienzo del mashup.

5. Posteriormente, el conjunto de C son renderizados en el Lienzo del mashup, como resultado de un modelo de mashup ejecutable y correcto.

6. Cuando un evento es activado bien sea por la interacción del usuario (Acción) o por las operaciones de otros componentes, este notifica los cambios de estado del componente al conjunto de C que constituyen el mashup.

7. Seguidamente, es seleccionado automáticamente el flujo de los datos más apropiado, teniendo en cuenta la información compartida por el evento ejecutado y el mapeo de datos realizado previamente en (1).

8. Así, interactivamente se vuelven a realizar los pasos (3), (4) y (5) en lugar de usar los parámetros para configurar los componentes.

Definición 4.2. *Modelo Parcial del Mashup (MMP)*

> Un *MMP* es una versión ejecutable y funcional preliminar del mashup, que se refina durante la interacción del usuario.

La sección siguiente describe la etapa de recomendación de componentes, con el fin de complementar el proceso de generación automática de mashups.

4.2.2 Recomendación Basado en Grafo de Conocimiento

Unas de las etapas que conlleva más labor durante la generación de mashups, es la selección adecuada de nuevos componentes. Así, teniendo en cuenta el análisis y los patrones de co-ocurrencia definidos en la Sección 4.2.3, este apartado presenta la fase de recomendación, encargado de facilitar a los usuarios finales la integración de componentes en el lienzo del mashup, sin que ellos deban realizar actividades de descubrimiento y selección de componentes.

De esta manera, las recomendaciones de componentes están basadas en un grafo, que relaciona tanto componentes con objetivos como las relaciones entre objetivos. Estas relaciones son encontradas por un algoritmo que aprendizaje no-supervisado que tiene en cuenta las interacciones del usuario y el conocimiento colectivo. Este grafo es llamado *Grafo de Conocimiento*, donde sus nodos son los objetivos y componentes; y sus aristas son las probabilidades de co-ocurrencia entre ellos. A continuación se presenta los conceptos y el proceso para la recomendación de componentes.

4.2.2.1 Grafo de Conocimiento

Los grafos son una estructura de datos para la representación de objetos y conceptos. La representación de grafos consta de dos elementos: los nodos, representan los objetos; y las aristas, son las relaciones entre dichos nodos. Una de las ventajas que los grafos ofrece, es la representación formal de sistemas que relacionan altas cantidades de datos [52], tales como: redes sociales o aplicaciones de marketing. Por ejemplo, dado una base de datos de objetos y una consulta, la tarea es recuperar uno o un grupo de objetos similares a la consulta.

Formalmente, se tiene que un grafo se representa como $G = (V, A)$, donde: V es el conjunto de vértices (también conocidos como nodos o puntos) y A es el conjunto de aristas (también conocido como arcos o líneas) [52].

A continuación se describen los tipos de grafos relevantes para el presente proyecto:

- Un grafo es dirigido, cuando todos las aristas tienen direcciones y, por lo tanto (u, v) y (v, u) se pueden distinguir. Es no dirigido, cuando no tienen dirección. De esta manera, para el presente proyecto se utiliza principalmente grafos dirigidos.

- Un grafo es etiquetado cuando se han asignado etiquetas a los vértices o aristas. Así, un grafo de este tipo se define por la tupla de G = (V, A, α, β), donde α: V $\rightarrow$ LV (LV, función de etiquetado de vértice) y β: A $\rightarrow$ LA (LA, función de etiquetado de borde).

- Un grafo es atribuido cuando se asocia un valor o peso a cada vértice o arista. Es decir, es atribuido de vértices cuando el grafo está en función de los vértices, y atribuido de aristas cuando el grafo está en función de las aristas (este tipo también es conocido como grafo ponderado).

- Un grafo es bipartito cuando los vértices pueden ser divididos en dos o más subconjuntos, tal que no haya aristas entre los vértices del mismo subconjunto.

De esta manera, la representación del grafo que almacena las relaciones es llamado *Grafo de Conocimiento* (GC), dado que este es generado tanto por una consulta donde se obtiene el conocimiento útil a partir de los mashup realizados y de las interacciones del usuario con el sistema. Asimismo, los nodos son los componentes y los objetivos; y las aristas son las relaciones encontradas entre ellos por medio del modelo LDA [53] y del algoritmo Apriori.

Por lo tanto, un GC es un grafo dirigido y con aristas-atribuidas, tal como:

$$GC = (O \bigcup C, A)$$

Donde: O representa un grupo de objetivos, C representa el conjunto de componentes y A las aristas que representan las relaciones tanto entre objetivos, como entre objetivos y componentes.

4.2.2.2 Generación de Conocimiento por medio de Aprendizaje Automático

El presente trabajo presenta dos etapas importantes para la generación de relaciones dentro del GC, de manera que sea posible inferirse automáticamente. La primera, encuentra las relaciones entre componentes y objetivos, haciendo tan solo uso de una consulta realizada en lenguaje natural. La segunda, consisten en mejorar y encontrar nuevas relaciones, tanto entre objetivos y componentes como entre objetivos.

A continuación se procederá a explicar con más detalle estas etapas, profundizando en la segunda etapa debido a que es desarrollada como parte del presente trabajo:

- **Descubrimiento de relaciones entre Componentes y Objetivos con base en la Consulta:**

Estas relaciones se generan como resultado de aplicar un modelo probabilístico conocido como LDA (*LatentDirichletAllocation*), que define una distribución conjunta sobre la consulta del usuario —realizada en lenguaje natural— a partir del cual es posible estimar la distribución de la probabilidad condicional de las objetivos sobre los recursos [53].

- **Definición y refinamiento de relaciones basado en el descubrimiento de conocimiento:**

Para llevar a cabo el descubrimiento de estas relaciones, se generaran a partir del la información recolectada por el sistema, razón por la cual es necesario usar una técnica de aprendizaje automático.

El aprendizaje automático es el campo de investigación dedicado al estudio formal de los sistemas de aprendizaje y al desarrollo de modelos apropiados para el análisis de un conjunto de datos [54]. Las técnicas de aprendizaje pueden ser clasificados en aprendizaje supervisado y no supervisado. Por un lado, el aprendizaje supervisado es una técnica utilizada para predecir el valor correspondiente de un elemento a partir de un conjunto de datos de entrenamiento. Por otro lado, el aprendizaje no supervisado se diferencia del anterior por la inexistencia de datos de entrenamiento; así, este tipo de aprendizaje, trata los objetos de entrada como un conjunto de variables aleatorias sobre el cual se construye un modelo para el conjunto de datos [54].

Así, para llevar a cabo el descubrimiento de las relaciones entre objetivos; y mejorar las relaciones entre objetivos y componentes con base en la consulta, teniendo en cuenta que no existen

datos apriori que permitan entrenar el sistema bajo una técnica de aprendizaje supervisado. A continuación, se describe el mecanismo implementado bajo el aprendizaje no-supervisado de reglas de asociación

4.2.2.3 Aprendizaje de Reglas de Asociación

Este tipo de aprendizaje automático se utiliza con frecuencia para descubrir las relaciones entre variables, patrones frecuentes, asociaciones o estructuras en colecciones de datos u otros tipos de repositorios [55]. Así, una regla de asociación establece una relación que se puede observar entre los ítems de una colección de datos.

Por ejemplo: la regla de asociación {leche, café} → {azúcar} encontrada en los datos de ventas de un supermercado, indicaría que un cliente que compra leche y café a la vez, probablemente compre azúcar también [56]. Dentro de los algoritmos más destacados en el aprendizaje de reglas de asociación se destaca el Algoritmo Apriori .

Algoritmo Apriori

A continuación se describe el algoritmo Apriori:

Algoritmo 2 Apriori.

Entradas:
C /*Conjunto de componentes de un repositorio*/
$minSupp$ /*Umbral mínimo de soporte, Definición 4.3*/
Salidas:
L /*Conjunto de elementos frecuentes de C*/
INICIO
$L1 \leftarrow EncontrarPrimerConjuntoElemtosFrecuentes(C)$ /*Obtiene un conjunto de elementos candidatos */
Para-cada l_{i-1} **en** $L1$ **Hacer:**
$Ck \leftarrow GenerarApriori(l_{i-1})$ /*Encuentra el numero de ocurrencias para cada elemento*/
Para-cada c_i **en** C **Hacer:**
$Ct \leftarrow ObtenerSubConjuntoCandidatos(Ck, c_i)$ /*Obtiene un subconjunto de elementos candidatos */
Para-cada ct_i **en** Ct **Hacer:**
$ct.contador{+}{+}$
Fin para-cada
$l_i \leftarrow \{c \in Ct \,|\, c.contador \geq minSupp\}$
Fin para-cada
$L \leftarrow \cup_i l_i$
Fin para-cada
Retornar L
FIN

El algoritmo 2 describe el proceso de asociación de elementos más frecuentes, a partir de un repositorio inicial de información. En la primera iteración del algoritmo, se obtiene un conjunto preliminar de elementos candidatos. Luego, el algoritmo explora en orden cada elemento del conjunto encontrado, para contar el número de ocurrencias de cada item. A partir de las reglas de asociación empleadas y analizando que estas cumplan con el criterio mínimo de soporte, son eliminadas las combinaciones irrelevantes. Por último, el algoritmo retorna un conjunto de elementos

que han superado el soporte mínimo.

Por medio de la implementación de este algoritmo son refinadas las relaciones entre los objetivos; y las encontradas entre los objetivos y componentes a través del modelo LDA. De esta manera, las reglas de asociación son apropiadas si se satisface el valor de umbral mínimo de soporte y confianza, descritos a continuación:

Definición 4.3. *Umbral Mínimo de Soporte (Supp)*

Se define como la proporción de transacciones T en la colección de datos, los cuales contienen tanto a un conjunto de ítems X como Y [57, 55]. Este umbral de soporte puede expresarse como:

$$supp(X \bigcup Y) = \frac{supp(X \bigcup Y)}{|T|}$$

Definición 4.4. *Umbral Mínimo de Confianza (Conf)*

Se define como la proporción de transacciones T que conteniendo a un conjunto de elementos X, también contienen al conjunto de Y [57, 55]. Este umbral de confianza puede expresarse como:

$$conf(X \bigcup Y) = \frac{supp(X \bigcup Y)}{supp(X)}$$

De acuerdo a esto, el Algorimo Apriori es el medio por el cual se procede a descubrir los patrones de co-ocurrencia PCO y PCOC (Sección 4.2.3) y persistirlos como relaciones en el GC. A continuación se describe este proceso a través del siguiente algoritmo:

Algoritmo 3 Descubrimiento de relaciones del grafo de conocimiento.

Entradas:

M `/*Conjunto de mashups generados*/`
minSoporteO `/*Minimo soporte entre Objetivos*/`
minSoporteCO `/*Minimo soporte entre Componente y Objetivo*/`
minConfianzaO `/*Minimo confianza entre Objetivos*/`
minConfianzaCO `/*Minimo confianza entre Componente y Objetivo*/`

INICIO

GO ← *ObtenerGrupoObjetivosApartirDe*(*M*) `/*Extraer los objetivos de cada mashup en grupos*/`
GCO ← *ObtenerGrupoObjetivosYComponentesApartirDe*(*M*) `/*Extraer los objetivos y componentes de cada mashup en grupos*/`
reglas ← *EncontrarRelacionesEntreObjetivos*(*GO*, *minSoporteO*, *minConfianzaO*)
GuardarRelacionesEnGrafo(*reglas*)
reglas ← *EncontrarRelacionesEntreComponentesYObjetivos*(*GCO*, *minSoporteCO*, *minConfianzaCO*)
GuardarRelacionesEnGrafo(*reglas*)

FIN

Con base en lo anterior, la sección siguiente sintetiza la propuesta para la selección adecuada de componentes durante la generación de mashups de clientes de forma automática y centrada en el usuario final.

4.2.2.4 Algoritmo de Recomendación

Haciendo uso de los tipos de recomendación ROV (Definición 3.17) y ROR (Definición 3.18), a continuación se propone un algoritmo que describe el proceso de recomendación de componentes que pueden ser incluidos en el estado actual del mashup.

Algoritmo 4 Recomendación de componentes basada en conocimiento.

Entradas:
Q_0 `/*Consulta informal según [53]*/`
Ac `/*Acción según Definición 3.8*/`
$Cent$ `/*Conjunto de componentes que constituyen el mashup actual*/`
Salidas:
$Csal$ `/*Componente recomendado*/`
INICIO
$Csal \leftarrow \{\}$ `/*Inicialización de variable*/`
$Cent' \leftarrow OrganizarPorNumeroInteracciones(Cent, Ac)$
Para-cada c_i **en** $Cent'$ **Hacer:**
$Csal$ **unir-con** $\{ ObtenerComponentesApartirdeROV(c_i)\}$ `/*Según Definición 3.17*/`
Fin para-cada
Si $Csal == \{\}$ **Entonces:**
Para-cada c_i **en** $Cent'$ **Hacer:**
$Csal$ **unir-con** $\{ ObtenerComponentesApartirROR(c_i)\}$ `/*Según Definición 3.18*/`
Fin para-cada
Fin si
Si $Csal == \{\}$ **Entonces:**
$R \leftarrow ObtenerRecursosApartirRecuperadorRecursos(Q_0)$
$Csal \leftarrow AdaptarRecursosRecuperados(R)$ `/*Retorna el conjunto de componentes a visualizar en el mashup*/`
Fin si
Retornar $Csal$
FIN

El algoritmo 4 describe el proceso de recomendación a partir de un conjunto componentes de entrada (Cent), que se encuentran en el modelo de mashup actual. En primera instancia, son organizados según el número de interacciones dando mayor importancia a aquellos que cuentan con mas interacciones. Posteriormente, se extraen los componentes a través de la recomendacion ROV, el cual realiza el proceso de recomendación a partir de los objetivos que se asocian directamente a los componentes actuales con mayor interacción. Luego, en caso que no se generen nuevas recomendaciones, se emplea las recomendaciones ROR, que recomienda a partir de la asociación de objetivos similares con los objetivos que representan a los componentes del mashup actual.

Por último, es necesario enviar una petición al módulo de *Recuperador de Recursos* (Sección 5.3.1), para solicitar por nuevos recursos acordes a la petición inicial del usuario (en lenguaje natural), en el caso que no hayan componentes para recomendar por los patrones de co-ocurrencia. En tanto existan componentes a recomendar, estos son desplegados en el lienzo del mashup.

4.2.3 Generación Automática de Mashups

Esta sección presenta el algoritmo de generación automática de mashups, que hace uso del modelo, integracion de componentes y recomendación de componentes definidos en las Secciones 3.2.2, 4.2.1 y , respectivamente. A continuación, la Figura 4.2 representa el flujo lógico del algoritmo que se

propone:

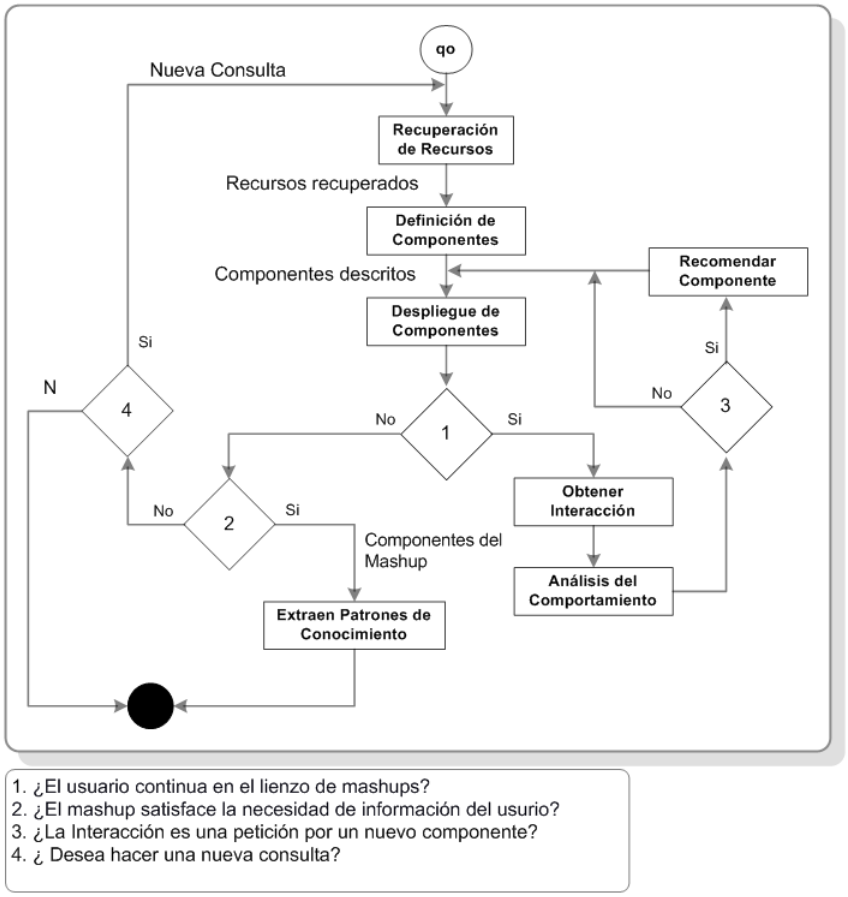


Figura 4.2: Flujo de generación de Mashups [Fuente Propia].

Cuando un usuario expresa una petición inicial Qo, el primer paso es recuperar los recursos que satisfacen dicha consulta, los cuales son adaptados y desplegados en el lienzo de integración del mashup. Una vez se visualizan los componentes en el mashup, se detecta cada interacción del usuario con el sistema, que determinan el proceso de recomendación de dichos componentes. Cabe resaltar que este proceso se realiza iterativamente hasta el punto que el usuario ha refinado su mashup, dando como resultado un recurso enriquecido que satisface las requerimientos iníciales del usuario. En el caso contrario, se da la posibilidad hacer una nueva consulta y el proceso comienza de nuevo, hasta que se cumplan los requerimientos de la búsqueda; además, se da la posibilidad de terminar el proceso de generación del mashup.

El flujo anterior se describe formalmente a través del Algoritmo 5:

Algoritmo 5 Generación automática de mashups.

Entradas:
Q_0 /*Consulta informal según [53]*/
Ac /*Acción según Definición 3.8*/
NC /*Número inicial de componentes a visualizar en el mashup*/
Salidas:
M /*Mashup final, segun Definición 3.6*/
INICIO
$MMP \leftarrow \{\}$ /*Inicialización de variable, según Definición 4.2*/
$R \leftarrow ObetnerRecursos(Q_0)$ /*Según Definición 3.1*/
$C \leftarrow ComputarRecursosRecuperados(R, NC)$ /*Adapta los recursos Web en componentes*/
$MMP \leftarrow IntegrarComponentesEnLienzo(C)$ /*Renderiza los componentes iniciales en el lienzo del mashups - Algoritmo 1*/
Mientras Ac **!=** $Stop$ **Entonces:**
$< ci, Ac > \leftarrow ObtenerAccionApartirDe(MMP)$ /*Obtiene la acción que relaciona la interacción del usuario con la interfaz grafica de un componente */
$AlmacenarYNotificarAcciones(ci, Ac)\}$ /*Almacena la acción realizada por el usuario sobre la interfaz de un componente y ejecuta los eventos asociados a dicha acción que son notificados el lienzo */
Si $NC < K$ **Entonces:**
$C \leftarrow RecomendarComponentesBasadosEnQ()$ /*Recomienda un nuevo componente apartir de la consulta [53]*/
Fin si
Si $NC > K$ **Entonces:**
$C \leftarrow RecomendarComponentesBasadosEnGrafoConocimiento()$ /*Algoritmo 4*/
Fin si
$MMP \leftarrow IntegrarComponentesEnLienzo(C)$ /*Renderiza los componentes recomendados en el lienzo del mashups - Algoritmo 1*/
Fin mientras
$M \leftarrow MMP$
Retornar M
FIN

El Algoritmo 5 describe el proceso de generación de mashups a partir de un conjunto de recursos inicial, que se obtienen del módulo "*Recuperador de recursos*" a partir de la consulta realizada por la petición de un usuario en lenguaje natural [53].

Una vez recuperado los recursos que cumplen los objetivos de la consulta Qo, se adecúa un número de componentes a partir del conjunto de recursos recuperados por medio de sus descriptores. Posteriormente, se agrega y renderiza cada componente en su interfaz propia a través del descriptor en el lienzo de composición del mashup. Posteriormente, a medida que el usuario interactúa con dichos componentes y en tanto no haya concluido el mashup, se obtienen las acciones del usuario asociadas a un componente, las cuales son almacenadas y tenidas en cuenta para recomendaciones siguientes.

Luego, son recomendados los componentes basados en los patrones de co-ocurrencia, que recuperan los componentes más adecuados para el modelo de mashup parcial. Así, cada componente recomendado es agregado y renderizado de nuevo en el lienzo de composición. Por último, una vez el usuario determina que ha finalizado su tarea de búsqueda, se termina el proceso de generación del mashup.

4.3 Estudio de caso

En esta sección, se describe un escenario que ejemplifica el entorno de generación automática de mashups a partir del comportamiento del usuario con el sistema. Para ello, imaginemos a John, una persona que trae consigo un Smartphone para realizar sus búsquedas diarias en la web. John planea visitar la ciudad de Popayán en época de celebración de semana santa y conocer los sitios turísticos de la ciudad durante su estadía. Así, antes de viajar a Popayán explora gran cantidad de información en la web y encuentra sitios interesantes.

Formalmente, lo que John desea es buscar sitios gastronómicos y turísticos dentro y alrededor de Popayán, pero encuentra información desorganizada en las diferentes páginas visitadas. Por lo tanto, él decide buscar y organizar la información a partir de cada página que satisfaga su necesidad, por lo que encuentra herramientas de desarrollo para usuarios finales, con cierta experticia en Internet, como Yahoo! pipes, entre otras que permiten la creación de nuevo contenido a partir de recursos existentes. Por lo tanto, Jhon para generar su contenido decide: primero definir el orden de las actividades que le gustaría realizar en la ciudad, segundo seleccionar las URL de los sitios vistos con mayor interés, tercero gestionar el filtro de búsqueda en la información seleccionada y por último, definir el flujo de información que desea obtener.

Sin embargo, para ser una plataforma de desarrollo orientado al usuario final, es indispensable que John conozca las funcionalidades de los componentes que emplea la herramienta, además del nivel de conocimientos requerido y la complejidad que se necesita para generar el contenido deseado, proceso que se torna inadecuado y dispendioso para usuarios básicos de Internet, debido a que ellos no están dispuestos a invertir tiempo en este proceso de búsqueda, composición y configuración de recursos, que debería realizarse generalmente.

Así, permitir que un usuario final realice una consulta como: "sitios para visitar en Popayán" y se genere contenido inmediato en una sola aplicación, simplifica notablemente actividades de recuperación y recomendación de contenido semejante a las plataformas de desarrollo web. De esta manera, John explora en la tienda de aplicaciones de Google y entre ellas, encuentra "Mobshapp" una aplicación encargada de generar y recomendar contenido —en forma de Mashups— acorde a la consulta del usuario.

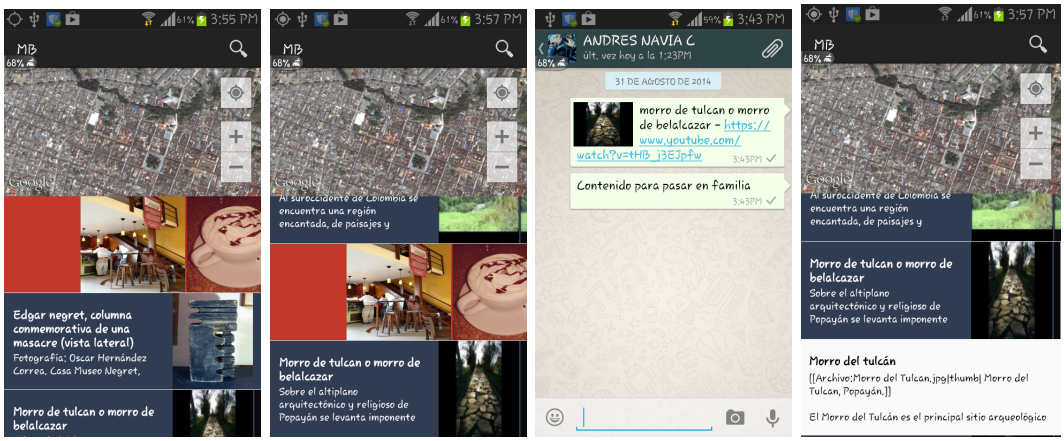
John descarga la aplicación y en una primera instancia introduce la consulta anterior, "Mobshapp" a través del descubrimiento de relaciones entre Componentes y Objetivos con base en la Consulta, entrega un conjunto de componentes que son renderizados en el lienzo del mashups, el cual puede satisfacer los escenarios de búsqueda de John, como: lugares turísticos, de comida, hoteles, religiosos, entre otros (Figura 4.3(a)).

John inicialmente rechaza información de hoteles, esta interacción indica que no es de interés lugares de hotelería para él (Figura 4.3(b)). Luego, John visualiza la información del componente "Catedral basílica de de nuestra señora de asunción" (Figura 4.3(c)), sin embargo rechaza los componentes de iglesias, indicando su bajo interés por estos recursos. Después de varias interacciones como observar y rechazar componentes, el contenido actual del mashups es el que se visualiza en la Figura 4.3(d). Luego, John realiza una llamada a un establecimiento gastronómico conocido en la ciudad a través de la aplicación (Figura 4.3(e)). Posteriormente, John desea saber cómo llegar al establecimiento al cual llamo; de esta manera, interactúa con una de las funcionalidades de la aplicación (Figura 4.3(f)). Después, John desea adicionar un nuevo componente a la búsqueda, por lo que "Mobshapp" recomienda componentes basados en el grafo de comportamiento, es decir, devuelve información apropiada a las interacciones anteriores, en este caso contenido de comida (Figura 4.3(g)). Por último, John decide eliminar información de museos indicando al sistema que

el objetivo de búsqueda para esta primera instancia, son lugares de comida.

Figura 4.3: Pantallas comportamiento primera sesión [Fuente Propia].

En una segunda instancia, John vuelve a realizar la misma consulta, pero esta vez tiene otras pretensiones con respecto a dicha consulta. “Mobshapp” recomienda y renderiza los recursos generados anteriormente junto con otros componentes relacionados a la búsqueda, como: comida, museos, campos verdes, entre otros (Figura 4.4(a)). John esta vez rechaza contenidos de museos, indicando de nuevo el no interés por estos sitios (Figura 4.4(b)). Luego, John comparte en sus redes sociales “contenido para pasar en familia”, como el “Morro de belalcázar” (Figura 4.4(c)). Nuevamente, John desea saber cómo llegar al "Morro de belalcázar"; de esta manera, se visualiza el camino como se observa en la Figura 4.3(f). Después, John elimina el contenido de comida enseñándole al sistema que cambió su interés anterior (Figura 4.4(d)). De esta manera, “Mobshapp” cambia las relaciones de los patrones de co-ocurrencia de componentes y para próximas interacciones con el sistema, como adicionar nuevo contenido, “Mobshapp” recomendara componentes basados en el comportamiento actual y devolverá información relacionada al nuevo interés de John.



(a) (b) (c) (d)

Figura 4.4: Pantallas comportamiento segunda sesión [Fuente Propia].

De esta forma, con el fin de ayudar a usuarios finales a generar su propio contenido, y su vez mitigar el crecimiento constante de la información en la web, en el ejemplo anterior, "Mobshapp" claramente mejora la experiencia de búsqueda del usuario, realizando recomendaciones con base en el análisis de su comportamiento con el sistema, a partir de lo cual es posible estimar sus objetivos de búsqueda.

4.4 Resumen

Este capítulo presentó la conceptualización de la solución propuesta para generar de manera automática mashups a partir del comportamiento del usuario final con el dispositivo móvil. Inicialmente, se abordó la descripción de los procesos llevados a cabo en el lienzo del mashup, con el objetivo de proporcionar la integración adecuada de los resultados. Finalmente, se presentó un estudio de caso con el fin de ejemplificar el enfoque propuesto.

Capítulo 5

Implementación de La Solución

5.1 Introducción

En este capítulo se detalla el proceso metodológico que soportó el diseño e implementación de "Mobshapp", un sistema para la creación de mashups de cliente centrado en el usuario final, basado en sus objetivos de búsqueda y su comportamiento con el sistema. La estructura del capítulo se organiza de la siguiente manera: En primera instancia, se describe las metodologías utilizadas para la documentación y el desarrollo (Sección 5.2). Seguido, se detalla la implementación del sistema por medio de vistas, como el modelo de datos, vista despliegue, entre otros (Sección 5.3).Por último, se presenta al lector una relación de las vistas propuestas con el fin de obtener una visión general del funcionamiento del sistema (Sección 5.4).

5.2 Metodologías Empleadas

Esta sección proporciona una introducción a las metodologías utilizadas durante la implementación del proyecto. Entre las metodologías se encuentran: para el proceso de desarrollo del sistema, *Scrum* [58] y *Test-DrivenDevelopment* [59]. Para aspectos técnicos y funcionales del sistema propuesto se siguió la metodología de *View&Beyond* [60], útil en la documentación de arquitecturas software.

5.2.1 Scrum

Scrum es una metodología de desarrollo ágil y flexible, enfocado en un proceso de desarrollo iterativo e incremental, en el que se definen y aplican un conjunto de buenas prácticas, donde un equipo de personas trabajan colaborativamente para conseguir un objetivo común. Este tipo de aproximaciones ágiles surgen para gestionar dinámicamente y coordinar procesos que garanticen el óptimo desarrollo de un proyecto. Un proyecto que emplea este tipo de metodologías se ejecuta en bloques temporales de tiempos cortos, que permita entregar una versión del sistema funcional después de cada iteración, con el propósito de obtener un software de calidad y descartar riesgos relevantes de manera anticipada. De esta manera, Scrum es una metodología adecuada que busca entregar resultados y responder a necesidades emergentes.

5.2.2 Test-DrivenDevelopment (TDD)

TDD es una práctica de programación de software que entrelaza estrictamente las tres actividades siguientes: Codificación, Testeo (pruebas de unidad) y Diseño (refactorización del código). Este proceso puede ser descrito en los siguientes pasos:

- Escribir de manera sencilla las prueba de unidad, describiendo un requerimiento del programa.
- Ejecutar la prueba, si la prueba falla es debido a que el programa le falta dicho requerimiento o la prueba puede estar errónea.
- Escribir el código necesario para que la prueba sea correcta, no tiene que ser el código que mejor implemente el requerimiento probado.
- "*Refactorizar*" el código teniendo en cuenta los criterios simplicidad, los cuales son: *i*) el código es verificado por pruebas automatizadas, *ii*) no hay duplicación en el contenido del código, *iii*) el código expresa separadamente una idea o responsabilidad distinta, *iv*) el código es compuesto por el mínimo número de componentes (clases, métodos y líneas) compatibles con los anteriores tres criterios.
- Repetir este proceso y de esta forma acumular una gran cantidad de pruebas que permitan comprobar unitariamente cada requerimiento del programa.

5.2.3 View And Beyond (V&B)

V&B tiene como objetivo guiar en el proceso de documentación de arquitecturas de sistemas software, que facilite la comprensión de un sistema por parte de los diversos lectores, haciendo hincapié en la utilización del concepto de vistas como el principio para documentar una arquitectura. Una vista puede representar diversos tipos de arquitecturas a partir de estructuras, relaciones, comportamientos, interfaces, entre otros. No obstante, esta metodología no se enfoca en realizar todas las vistas posibles para describir una documentación adecuada, sino en describir las vistas más relevantes que representen el sistema, siempre y cuando facilite la comprensión por parte de los lectores. De esta manera, V&B es un metodología que busca que un grupo de personas que integran el proceso de desarrollo de un sistema, pueda analizar y comprender las técnicas y funcionalidades del mismo a partir de vistas relevantes.

5.3 Vistas del Sistema

Esta sección describe el proceso de desarrollo del presente proyecto, siguiendo la metodología de documentación introducida en la primera sección, a partir de la especificación de tres tipos de vistas: módulos, proporciona a sus lectores un modelo coherente del sistema, el cual abstrae en unidades de implementación de software para proveer un conjunto de responsabilidades. La vista de componentes y conectores, contiene elementos que constituyen características de ejecución por medio de subsistemas (tales como clientes, servidores, y repositorios de datos) y relaciones (tales como protocolos, flujos de información, entre otros). La vista de asignación, describe la comunicación entre la arquitectura, el entorno software y físico sobre el cual se despliega, implementa y se ejecuta el sistema.

Las vistas mencionadas evidencian la siguiente organización:

- *Presentación Principal:* Expone inicialmente una gráfica de la arquitectura general o parcial del sistema, en la cual se visualizan sus elementos y las relaciones entre los mismos.
- *Catálogo de Elementos:* Detalla los elementos representados en el punto anterior, (presentación principal). Así, en esta sección puede incluir los elementos y sus propiedades, relaciones y sus propiedades, interfaces de los elementos y comportamiento de los elementos.
- *Diagrama de Contexto:* Permite definir el alcance de una vista, indicando cómo el sistema representado en la vista se relaciona con su entorno. Por lo general, en esta parte también se expone y describe un diagrama.
- *Guía de Variabilidad:* Son puntos susceptibles a cambios en la arquitectura, de manera que se expone los puntos que en la arquitectura puede sufrir cambios y/o alteraciones. Algunos de los puntos de variación son las siguientes: sustitución de elementos, parametrización, composición de elementos, plantillas, entre otros.
- *Decisiones Arquitecturales y de Implementación:* Brinda una explicación sobre la decisión a nivel de diseño e implementación de una arquitectura.

5.3.1 Módulos

Este tipo de vista abstrae en un alto nivel los módulos de implementación que constituyen el sistema, mediante la especificación de la siguiente vista:

5.3.1.1 Vista de Descomposición

Esta vista describe la organización del sistema en *módulos* y *sub-módulos*, con el fin de aclarar la interacciones del sistema [60].

Presentación Principal

La Figura 5.1 expone un diagrama de alto nivel, que representa la vista general y los módulos que lo constituyen.

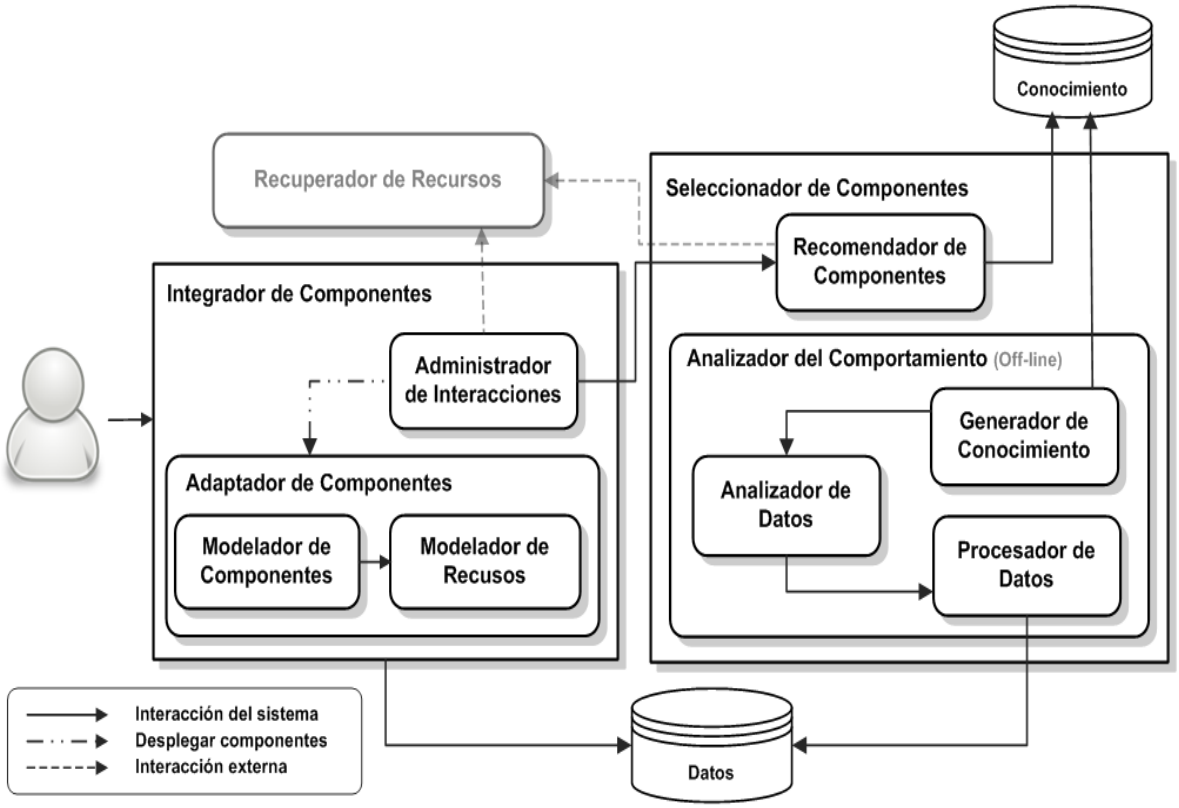


Figura 5.1: Vista de descomposición [Fuente Propia].

Catálogo de Elementos

Recuperador de Recursos: Implementa la solución propuesta en el trabajo de grado denominada "*Estimación del Intento de Búsqueda del Usuario para la Recuperación de Recursos en la Web*" elaborada por Ing. Luis Javier Suarez Meza para optar al título Magíster en Ingeniería Telemática [61]. Este módulo se encarga de recuperar los recursos con base en las intenciones y objetivos identificados a partir del análisis de la consulta del usuario.

Integrador de Componentes: Implementa la conceptualización de la solución propuesta en la Sección 3.2.2 (Modelo de Mashup), con el objetivo de generar de manera automática mashups a partir del Modelo de Componente (Sección 3.2.1) de forma transparente para el usuario. Este módulo soporta el siguiente conjunto de sub-módulos:

- ***Adaptador de Componentes:*** Tiene como objetivo generar la comunicación, eventos, configuraciones iníciales y operaciones entre los componentes que constituyen el mashup, adaptar la vista gráfica de cada recurso provenientes de la web y garantizar la comunicación coherente entre componentes.
  - ***Modelador de Recursos:*** Responsable de abstraer la información de cada de recurso Web proveniente del módulo *Recuperador de Recursos* a través de un descriptor, para que se pueda comunicar correctamente con otros recursos utilizados en el mashup.

  - ***Modelador de Componentes:*** Encargado de implementar la conceptualización de la solución propuesta en la Sección 3.2.1 (Modelo de Componente). De manera que un recurso descrito por el *Modelador de Recursos* pueda ser representado mediante un componente. También, proporciona los mecanismos para que el usuario pueda interactuar con los componentes del sistema a través de la interfaz gráfica.

- ***Administrador de Interacción:*** Encargado de detectar y analizar las acciones del usuario con los componentes que constituyen el mashup, además de persistirlas en la *Base de Datos*.

Seleccionador de Componentes: Implementa la conceptualización de la solución propuesta en la Sección 4.2.2, al proporcionar al usuario los componentes que más se adecuen a sus requerimientos. Los elementos que soportan este módulo son:

- ***Recomendador de componentes:*** Encargado de preguntar al repositorio de conocimiento por nuevos componentes, haciendo uso de los patrones de recomendación (Definición 3.17 y 3.18). Este analiza los componentes que pueden ser recomendados y decide si son adecuados —por ejemplo, no reenvía los componentes que ya han sido enviados—. Dado el caso que hayan recomendaciones, se realiza la petición por nuevos componentes al módulo de *Recuperador de Recursos*.

- ***Procesador de datos:*** Adecua la información que se ha persistido en el repositorio de datos, rechazando los datos que pueden generar corrupción en el proceso de análisis. Además, convierte la información en un formato correcto que recibirá el modulo *Analizador de Datos*.

- ***Analizador del Datos:*** Este módulo analiza los mashups que están siendo y fueron desarrollados previamente, encontrando las relaciones definidas en la Sección 4.2.3. Así, a partir de estas relaciones: *i*) son generadas las reglas que definen las asociaciones entre objetivos o *ii*) sean diferenciados los tipos de componentes que han sido más utilizados en los mashups o que están siendo desarrollados. Para el primero, son tomados los mashups mejor renqueados y extraídas las reglas de asociación entre objetivos por medio del Algoritmo Apriori, descrito en la Sección 4.2.2.3. El segundo, es realizado por medio de un mapeo sobre la Base de Datos.

- ***Generador de Conocimiento:*** Examina las reglas de asociación encontradas por el *Analizador de Datos* y las persiste como relaciones en el *Grafo de Conocimiento* descrito en la Sección 4.2.2.1. Además, tiene la capacidad de modificar las reglas de asociación encontradas y que estas mejoren a medida que el sistema vaya creciendo, lo cual implica mejores recomendaciones.

Datos: Repositorio que almacena los recursos y componentes modelados, así como las interacciones del usuario con el sistema. Estos datos son analizados por medio del *Generador de Conocimiento*, el cual sirve como insumo para la recomendación de componentes en el *Lienzo de Integración*.

Conocimiento: Repositorio que almacena el *Grafo de Comportamiento* definido en la Sección 4.2.2.1. Este contiene los objetivos y los componentes y sus respectivas relaciones a manera de Grafo.

Diagrama de Contexto

No aplica.

Guía de Variabilidad

Los puntos susceptibles a cambiar son: el *Modelador de Recursos*. De acuerdo a la solución propuesta en la Sección 3.2.1, el modelador es proclive a cambiar a partir de los adaptadores empleados para abstraer la información de los recursos Web provenientes del módulo *Recuperador de Recursos*.

Decisiones Arquitecturales y de Implementación

Los módulos y sub-módulos que componen el sistema fueron implementados en Java[1] y RubyOnRails[2]. Java es un lenguaje de programación orientado a objetos, robusto, interpretado, de distribución libre e independiente de la arquitectura hardware. Java es seleccionado para el presente proyecto debido a que es un lenguaje multiplataforma, que se puede implementar en diferentes sistemas operativos como dispositivos móviles y soporta un conjunto de librerías, que facilitan la programación a través de interfaces gráficas y para la manipulación de información en el módulo Integración de Componentes.

Por último, RubyOnRails es un framework para aplicaciones Web de código abierto, bajo el paradigma *Modelo Vista Controlador* (MVC) y escrito en lenguaje de programación Ruby. Este lenguaje permite realizar meta-programación de la cual Rails hace uso. Este framework combina la simplicidad con la posibilidad de desarrollar aplicaciones con menos código y con un mínimo de configuración en comparación con otros frameworks. RubyOnRails es seleccionado para el presente proyecto, debido a que permite generar servicios RESTfull de forma sencilla y confiable en la comunicación entre los módulos de Integración de Componentes y Selección de Componentes.

5.3.2 Componentes y Conectores (C&C)

Este tipo de vista describe el sistema a partir de la especificación de elementos que presentan características de ejecución, tales como procesos, clientes, servidores, y registros de datos, los cuales a su vez incluyen elementos de interacción o comunicación, tales como protocolos y flujos de información. Esta sección se dedica a detallar la Vista de Call-Return del sistema, que define una separación del sistema en un conjunto de elementos, cada uno de los cuales ofrece una funcionalidad específica y bien definida [60].

5.3.2.1 Vista Call-Return

Esta vista describe un modelo computacional en el que los elementos proporcionan un conjunto de funcionalidades que pueden ser invocadas por otros elementos, y los conectores son responsables de transmitir tanto la petición desde el solicitante al proveedor, como los resultados retornados.

[1] **Disponible en:** http://www.oracle.com/es/technologies/java/overview/index.html
[2] **Disponible en:** http://rubyonrails.org/

Presentación Principal

La Figura 5.2 expone un diagrama Call-Return de alto nivel del sistema.

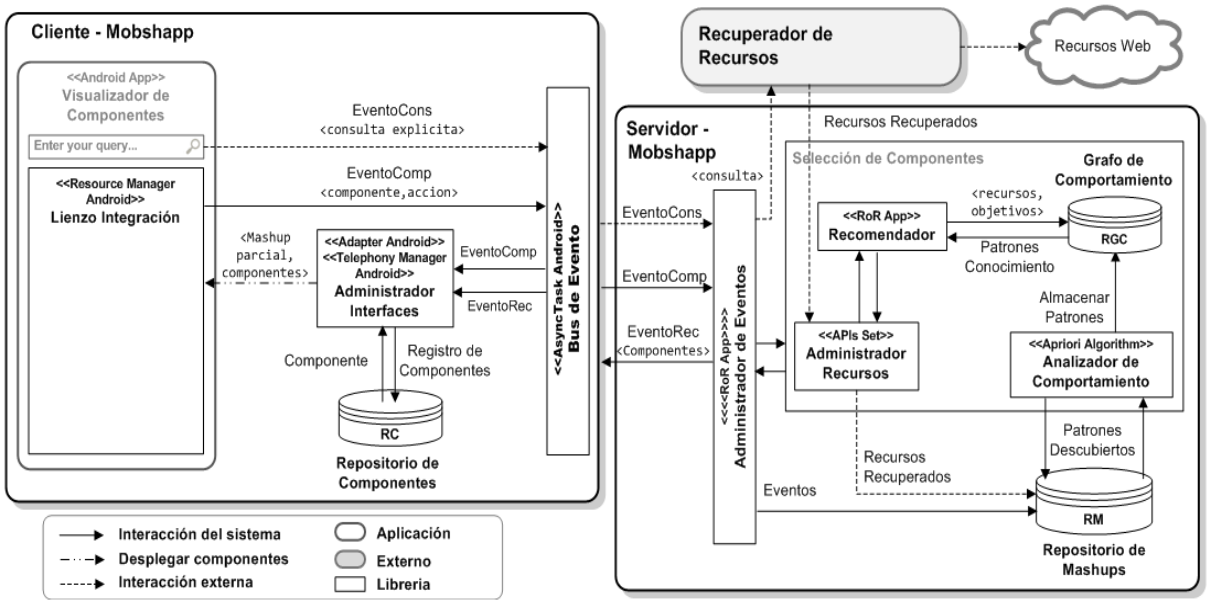


Figura 5.2: Vista Call-Return (Arquitectura general del sistema) [Fuente Propia].

Catálogo de Elementos

La arquitectura propuesta consta de un esquema cliente-servidor, donde el cliente corresponde a una aplicación móvil encargada de detectar el comportamiento del usuario, generar automáticamente el mashup y presentar las nuevas recomendaciones de contenido. Por otro lado, el servidor se encarga de adaptar los recursos web y generar una descripción para cada uno, procesar la información obtenida desde el cliente y ejecutar los mecanismos propuestos para generar las respectivas recomendaciones.

Cliente:

- ***Administrador de Interfaces:*** Encargado de "*renderizar*" automáticamente la interfaz gráfica de los componentes que constituyen el mashups y de reorganizar los componentes en el *Lienzo de Integración*, a partir de las acciones ejecutados por el comportamiento —interacciones— del usuario o internos de la aplicación.

- ***Lienzo de Integración:*** Interfaz gráfica de usuario que contiene la disposición de los componentes que constituyen el mashup actual,acorde a la consulta e interfaz donde interactúa el usuario final, por medio de sus acciones.

- ***Repositorio de Componentes (RC):*** Corresponde al repositorio local del dispositivo móvil, en donde se almacenan los componentes que constituyen cada mashup actual y los que hayan satisfecho los requerimientos de un usuario. Esta información es borrada del disposi-

tivos móvil cada vez que el usuario termine la sesión del sistema, debido a las restricciones de memoria que pueden darse en estos dispositivos.

Servidor:

- ***Administrador de Recursos:*** Encargado de recibir los recursos provenientes del *Recuperador de Recursos* y de persistirlos. Este recibe las interacciones hechas por el usuario sobre los componentes en el *Lienzo de Integración* y según la interacción, consulta al *Recomendador* por nuevos componentes.
- ***Recomendador:*** Encargado de sugerir componentes a partir de las acciones realizas por el usuario y los componentes que constituyen el mashup actual a través de los patrones de recomendación (Definición 3.17 y 3.18).
- ***Analizador de Comportamiento:*** Este módulo realiza un proceso de data mining con el fin de descubrir conocimiento potencial dentro de los mashups previamente almacenados en el repositorio de mashups. Este módulo procesa la información iterativamente y de manera off-line, mejorando la calidad del conocimiento extraído.
- ***Repositorio de Mashups:*** Corresponde al repositorio donde se almacena toda la información relacionada a los mashups y al proceso de generación de los mismos.
- ***Grafo de comportamiento (GC):*** Corresponde al repositorio donde se almacena todo el conocimiento extraído por el *Analizador de Comportamiento.*
- ***Bus de Eventos y Administrador de Eventos:*** Estos elementos son los responsables de controlar los eventos entre el cliente y servidor, respectivamente los generados por la interacción del usuario con el sistema. Para este caso, se definen los siguientes tipos de eventos: Evento de consulta, de comportamiento y de recomendación:
  - ***Evento de consulta (EventoCons):*** Representa la consulta inicial realizada por el usuario y contiene la expresión de la consulta.
  - ***Evento de comportamiento (EventoComp):*** Representa la interacción del usuario con un componente o el mismo sistema. Este contiene información del componente relacionado con la interacción y el tipo de acción ejecutado.
  - ***Evento de recomendación (EventoRec):*** Este es generado por la interacción del usuario a través del evento de comportamiento y representa la respuesta del servidor cuando se requiere una nueva recomendación, es decir, contiene la información del nuevo componente a recomendar con su respectivo descriptor para ser desplegado en el *Lienzo de Integración.*

Diagrama de Contexto

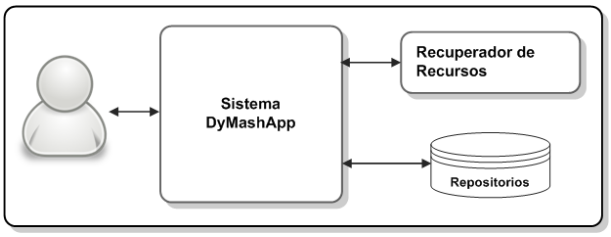


Figura 5.3: Diagrama de Contexto – Vista CallReturn.

Guía de Variabilidad

Los puntos susceptibles a cambiar fueron descritos en la Sección 5.3.1.1.

Decisiones Arquitecturales y de Implementación

Como se describió en la Sección 5.3.1, los elementos que componen el sistema fueron implementados en Java y RubyOnRails, respectivamente en el cliente y servidor. Agregando a lo anterior, el esquema del cliente es implementado a través del sistema operativo Android, debido a que es considerado uno de los sistemas operativos más utilizados sobre teléfonos inteligentes, tabletas, entre otros.

A continuación se listan las librerías más representativas, utilizadas en este trabajo de investigación. Por una parte, en el desarrollo de los elementos del cliente se destacan:

- ***Resource Manager Android:*** Gestiona todos los elementos que forman parte de una aplicación y que están fuera del código; por ejemplo, imágenes o lienzos de diseño dinámicos. Específicamente, el Resource Manager apoya la agregación de elementos en el *Lienzo de Integración*, con el fin contener la disposición de los componentes a partir de las interacciones del usuario.

- ***Telephony Manager Android:*** Proporciona funcionalidades internas del sistema operativo, tales como llamadas, enviar SMS/MMS, entre otros. Específicamente, Telephony Manager apoya la implementación en el elemento *Administrador de Interfaces*, con el fin de proporcionar funcionalidades a las vistas graficas de los componentes.

- ***Google APIs:*** Google Play Service contempla a Google APIs, al proporcionar un kit de desarrollo y una interfaz de programación, para permitir a las aplicaciones hacer uso de funcionalidades que se integran directamente con los servicios de Google; por ejemplo, el servicio de localización, Youtube, entre otros. Específicamente, Google APIs apoya la implementación en el elemento *Administrador de Interfaces*, con el fin de proporcionar las vistas graficas de los componentes relacionados a los servicios de Google.

- ***AdapterAndroid:*** Un Adapter es el encargado de proporcionar una vista gráfica a cada ítem de un conjunto de datos y además, actúa como puente entre la vista y los datos subyacentes al proveer acceso a la información de los elementos visualizados. Específicamente, Adapter apoya la implementación en el elemento *Administrador de Interfaces*, al proporcionar las vistas graficas de los componentes que constituyen el mashup.

- El elemento ***Repositorio de Componentes*** utiliza el almacenamiento externo del esquema cliente a través de un formato ligero para el intercambio de datos, conocido como JSON (*JavaScriptObjectNotation*). La simplicidad de JSON ha dado lugar a la divulgación de su uso, como alternativa a XML y bases de datos.

Por otra parte, en el desarrollo de los elementos del servidor se destacan:

- ***Apriori:*** Encuentra de manera eficiente las reglas de asociación dentro de grandes conjuntos de transacciones. Esta librería es una interfaz Ruby desarrollada en C que implementa este algoritmo de forma rápida debido a que utiliza un árbol de prefijos, que permite organizar los contadores usados para los conjuntos de elementos. Para lo cual se realizó una contribución a la comunidad de código abierto, que consiste en la actualización de la misma, con el fin de ser usada en las versiones de Ruby 2.x.

- ***Neography:*** Genera una capa de abstracción a la API de Neo4j, permitiendo utilizar desde Ruby gran mayoría de sus funcionalidades. También se hizo una contribución a la comunidad de desarrolladores, añadiendo una nueva característica (adherir propiedades a las relaciones únicas cuando se realizan los baches de transacciones) con sus respectivas pruebas de unidad.

- ***Sidekiq, Sidetiq y Redis-semaphore:*** Permite ejecutar procesos en segundo plano por medio del uso de Múltiples-Hilos de forma eficiente en memoria, además de ejecutar en forma periódica y evitar inconsistencia de datos a la hora de ser procesados. Específicamente, estas librería apoya la implementación en el módulo Analizador de Conocimiento.

- El elemento ***Repositorio de Mashups*** utiliza MongoDB, un sistema de base de datos —multiplataforma— orientado a documentos, conocido como base de datos no relacional. MongoDB evita la estructura de datos relacional tradicional, basada en tablas, inclinándose a favor de documentos como JSON, con esquemas dinámicos. Esta característica, hace que la integración de datos en ciertos ámbitos (como BigData), sea eficiente.

- El elemento ***Grafo de Comportamiento*** es implementado en Neo4j, una base de datos de código abierto orientada a grafos, implementada en Java. De esta manera, Neo4j es un motor de persistencia transaccional —transacciones ACID— que almacena los datos estructurados en grafos, en lugar de tablas. Neo4j realiza los proceso de forma rápida a la hora de encontrar información relacionada, por medio de un lenguaje dedicado para hacer sencillas las consultas y permite almacenar grandes cantidades de nodos, relaciones y propiedades. Además, a través de su interfaz REST permite fácilmente ser accedida desde otras plataformas o sistemas.

5.3.3 Asignación

Este tipo de vista describe el mapeo entre la arquitectura general, el entorno software y físico sobre el cual se despliega y ejecuta el sistema.

5.3.3.1 Vista de Despliegue

Esta vista describe un diagrama de despliegue del sistema, cuyo propósito es distribuir los elementos descritos en la vista de Componentes y Conectores, en equipos o dispositivos físicos [60].

Presentación Principal

La Figura 5.4 expone un diagrama de despliegue en UML del sistema.

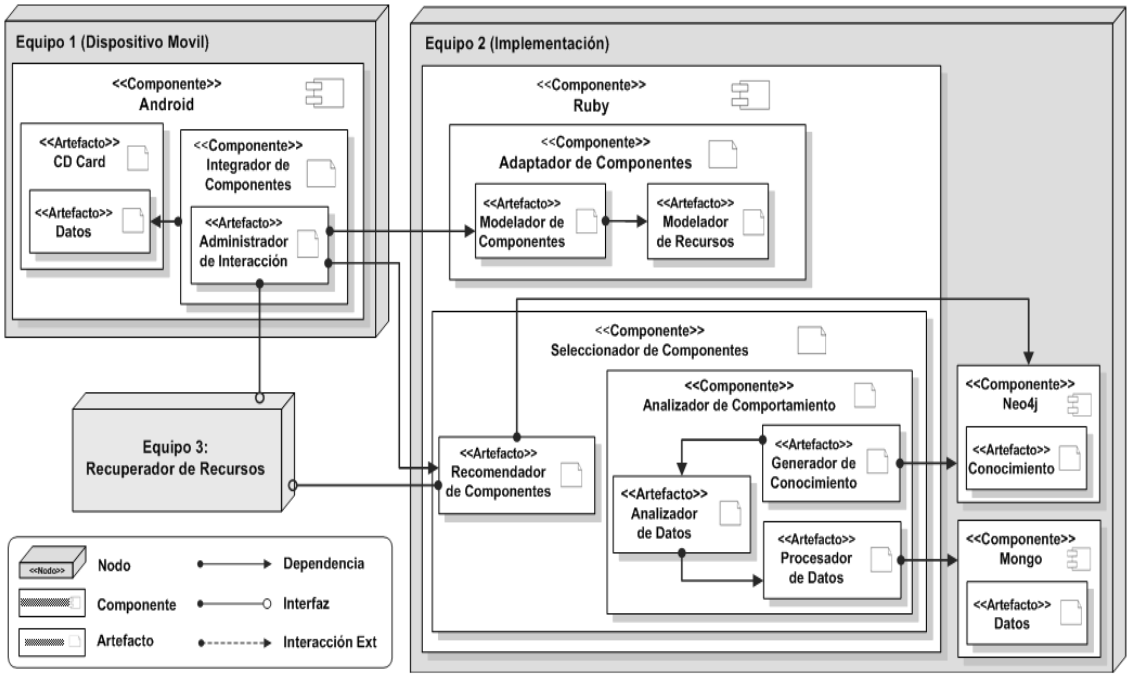


Figura 5.4: Vista de descomposición en capas [Fuente Propia].

Catálogo de Elementos

Para llevar a cabo los procesos descritos en la vista de Componentes y Conectores en la Sección 5.3.2, es necesario distribuir los elementos del sistema en equipos que respondan a la ejecución de las diferentes funcionalidades de los esquemas en el cliente y servidor, respetivamente. Por lo tanto, para el presente proyecto de investigación, se cuenta con dos equipos que presentan las características expuestas en el Cuadro 5.1.

Cuadro 5.1: Características de equipos

Características	**Equipo 1**	**Equipo 2**
Tipo de Equipo	Dispositivo Móvil	PC Escritorio
Sistema Operativo	Android v4.2 JellyBean	Linux Ubuntu 13.10
Versión Kernel Linux	>= 2.6.25	3.11.0-14-generic
Memoria Ram	2GB	8GB
Procesador	quad-core Snapdragon S4 Pro - 1.5GHz	Intel Core i5
Disco Duro	16GB memoria interna	1T

Diagrama de Contexto

No aplica.

Guía de Variabilidad

La implementación del sistema propuesto no contempla dispositivos móviles con versiones del sistema operativo Android inferiores a 3.3, debido a la falta de almacenamiento externo. Además, existen puntos susceptibles a cambiar en cuanto a la librería de soporte para fragmentos, dado que es usada para construir interfaces de usuario dinámicas en sistemas operativos de Android inferiores a 4.0.

Decisiones Arquitecturales y de Implementación

Los dispositivos móviles son terminales pequeños con algunas limitaciones en cuanto a capacidades de procesamiento, conexión a redes de internet y memoria con respecto a equipos de cómputo. De esta manera, el equipo 1 se dejó lo suficientemente ligero para brindar una adecuada visualización del entorno de diseño y de ejecución, mientras que el equipo 2 se encarga de las tareas con mayor procesamiento de carga, tales como: adaptación de componentes y procesos de selección de componentes donde hay una gran cantidad de operaciones, procesos periódicos y off-line que haría que el dispositivo móvil se bloqueara o agotara su batería rápidamente.

5.4 Relación Entre Vistas

Esta relación permite a un lector comprender las asociaciones relevantes entre las vistas, permitiendo obtener una visión general del funcionamiento del sistema [60]. Dado que todas las vistas de una arquitectura describen el mismo sistema, dos vistas pueden encontrar varios puntos en común.

El Cuadro 5.2 representa la asociación entre módulos, componentes y equipos de las vistas de Descomposición, de Conectores y Componentes (C&C), y de Asignación, respectivamente. La forma como se debe comprender dicho cuadro es, por ejemplo: el componente *Analizador de Comportamiento* (de la vista de C&C) es implementado por el módulo *Generador de Conocimiento* (de la vista de Descomposición) que a su vez, se encuentran desplegados en el *Equipo 2* (de la vista de Asignación del sistema).

Cuadro 5.2: Relación entre vistas de Descomposición, Conectores y Componentes, y de Asignación.

Vistas			
Módulos	**Sub-módulos**	**C&C**	**Asignación**
Integrador de Componentes	Adaptador de Componentes	Administrador de Interfaces	Equipo 1
	Modelado de Componentes	Administrador de Recursos	Equipo 2
	Modelado de Recursos		
	Administrador de Interacción	Bus de evento	Equipo 1
		Administrador de Eventos	Equipo 2
	Datos	Repositorio de Componentes	Equipo 1
		Repositorio de Mashups	Equipo 2
Seleccionador de Componentes	Recomendador de Componentes	Recomendador	Equipo 2
	Generador de Conocimiento	Analizador de Comportamiento	
	Analizador de Datos		
	Procesador de Datos		
	Conocimiento	Grafo de Comportamiento	
Recuperador de Recursos	Recuperador de Recursos Recuperador de Recursos	Recuperador de Recursos	Equipo 3

5.5 Resumen

Este capítulo contiene la síntesis documental de la arquitectura que define el sistema propuesto en esta investigación. La arquitectura comprende los artefactos software desarrollados, los cuales se describen mediante View&Beyond (V&B). Otras de las metodologías que acompañaron el presente proceso de investigación, son: Scrum, que guía el proceso de desarrollo y Test-DrivenDevelopment (TDD), que guía el proceso de pruebas. De acuerdo a V&B, en este capítulo la descripción arquitectónica se conforma de tres estilos de vistas: de Módulos, la cual ofrece un modelo mental coherente del sistema completo y su propósito; de Componentes y Conectores, contiene elementos que presentan características de ejecución, interacción y/o comunicación; y de Asignación, describe el mapeo entre la arquitectura y el entorno software y físico sobre el cual se despliega y ejecuta la plataforma. Al final se expone la relación entre vistas, con el fin de brindar una visión completa de la articulación de los diversos elementos que conforman el sistema.

Capítulo 6

Evaluación y Resultados

6.1 Introducción

Este capítulo tiene como propósito evaluar dos aspectos: *i*) evaluar la relevancia del mashup generado por el sistema y *ii*) el grado de aceptación del sistema por parte de un usuario final, ambas pruebas realizadas sobre un entorno de prueba. De esta manera, el contenido de este capítulo está dedicado a detallar la evaluación del sistema, la cual se fundamenta en la metodología DESMET. La estructura del capítulo se organiza de la siguiente manera: En primera instancia se describe la metodología utilizada (Sección 6.2). Seguido se describe la definición del método de evaluación empleado a partir de un conjunto de características (Sección 6.3). Posteriormente se detallan las actividades de evaluación (Sección 5.4). Luego se detallan los resultados conseguidos en las pruebas realizadas (Sección 5.5). Por último, se realiza un análisis de los resultados obtenidos (Sección 5.6).

6.2 Metodología de Evaluación

El objetivo de esta sección es dar conocer los métodos de evaluación, el proceso que se debe llevar a cabo para seleccionar el método apropiado y realizar la prueba pertinente para el presente proyecto, a partir de la metodología propuesta por Kitchenhamen [62], conocida como DESMET.

DESMET es un proyecto realizado con el propósito de ayudar a las organizaciones a planificar y ejecutar evaluaciones sobre sus procesos, métodos y herramientas de ingeniería de software. Esta metodología considera los siguientes tres tipos de evaluación:

- **Cualitativas u objetivas**: Tiene como fin establecer la capacidad de un método o herramienta software, es decir, qué tan bien se adapta un método o herramienta a las necesidades de una organización o de personas particulares. Así, los evaluadores miden el grado en que el método o herramienta proporciona las características necesarias con base en una opinión personal.

- **Cuantitativas:** Tiene como objetivo establecer los efectos medibles de la utilización de un método o herramienta software en términos medibles. Los efectos medibles se basan generalmente en la reducción del tiempo de producción, reconstrucción, mantenimiento y costos. En otras palabras, la evaluación cuantitativa permite determinar si los beneficios esperados son realmente entregados a la organización.

- **Hibrida:** Finalmente, los métodos híbridos son aquellos que involucran tanto los elementos objetivos como subjetivos.

Por otra parte, DESMET propone las siguientes tres formas de organizar un proceso de evaluación:

- **Experimento formal:** Sugiere dos grupos para llevar a cabo la evaluación, experimentales (usuarios potenciales de la herramienta) y de evaluación (usuarios para realizar la evaluación). Este tipo de evaluación solicita a los usuarios que exploren diferentes tareas a través del método o herramienta bajo evaluación.

- **Estudio de caso:** Consiste en la evaluación de un método o herramienta de investigación aplicado sobre un proyecto real o escenario de estudio.

- **Encuesta:** Similar al experimento formal, ya que es realizada por la apreciación de los usuarios con y sin experiencia en el tema de investigación, para que proporcionen información sobre los métodos o herramienta software en evaluación.

De esta manera, la metodología DESMET define nueve métodos de evaluación diferentes que son descritos a través del Cuadro 6.2 de la Sección 6.3.2, que muestra un resumen de las condiciones favorables y desventajas de cada uno de los métodos propuestos. DESMET define el siguiente conjunto de criterios que permiten seleccionar el método de evaluación adecuado de acuerdo a las necesidades específicas de cada proyecto:

- Contexto de evaluación
- Naturaleza del impacto esperado
- Naturaleza del objeto de evaluación
- Alcance del impacto del objeto de evaluación
- Madurez del objeto de evaluación
- Tiempo requerido para entender los principios delineados por el objeto de evaluación
- Tiempo requerido para llegar a ser un experto en el uso del objeto de evaluación
- Capacidad de medición de la organización llevar a cabo la evaluación

6.3 Análisis Métodos de Evaluación

6.3.1 Escenario de Estudio

El objeto de evaluación se efectuó a través de tres escenarios, que buscan ubicar a un turista en tres problemas diferentes, los cuales debe solucionar mediante la aplicación que implementa el enfoque propuesto en este trabajo de grado. Los escenarios propuestos se centran en: lugares turísticos, hospedaje y restaurantes en la ciudad.

A continuación, se expone el funcionamiento del sistema independiente de los escenarios propuestos. Específicamente, cada escenario describe el flujo expuesto en la Figura 4.2.

Inicialmente, el turista emite una consulta *q0* que refleja los objetivos de búsqueda, por ejemplo *q0 = { lugares turísticos }*. Cuando se expresa una petición *q0*, el primer paso es recuperar un

conjunto de recursos (*Recuperador de Recursos*, Sección 5.3.1) acorde a los objetivos de búsqueda, que luego son desplegados en el lienzo de composición del mashup.

Una vez ejecutada una acción (Definición 3.8) sobre el componente seleccionado por el usuario, se generan dos eventos: Por un lado, el primer evento es analizado para proporcionar recomendaciones acordes a la acción y el componente. Por otro lado, el segundo evento es analizado para rende-rizar automáticamente el lienzo del mashup según la acción. El flujo anterior está descrito formalmente por el Algoritmo 5.

Por otro lado, debido a que no existen datos apriori, se procede a generar un conjunto de 100 registros de mashups a través de las interacciones realizadas por usuarios expertos con el sistema, esto sin hacer uso del GC, solamente por medio de las recomendaciones hechas por la consulta en lenguaje natural (CsC aproximando a 1000). Luego, se inicia un proceso Off-line en la generación de conocimiento (Seccion 4.2.2.4), con el fin de generar las primeras relaciones entre objetivos y objetivos-componentes, donde fueron generadas 1271 relaciones y 126 nodos como se observa en la figura 6.1.

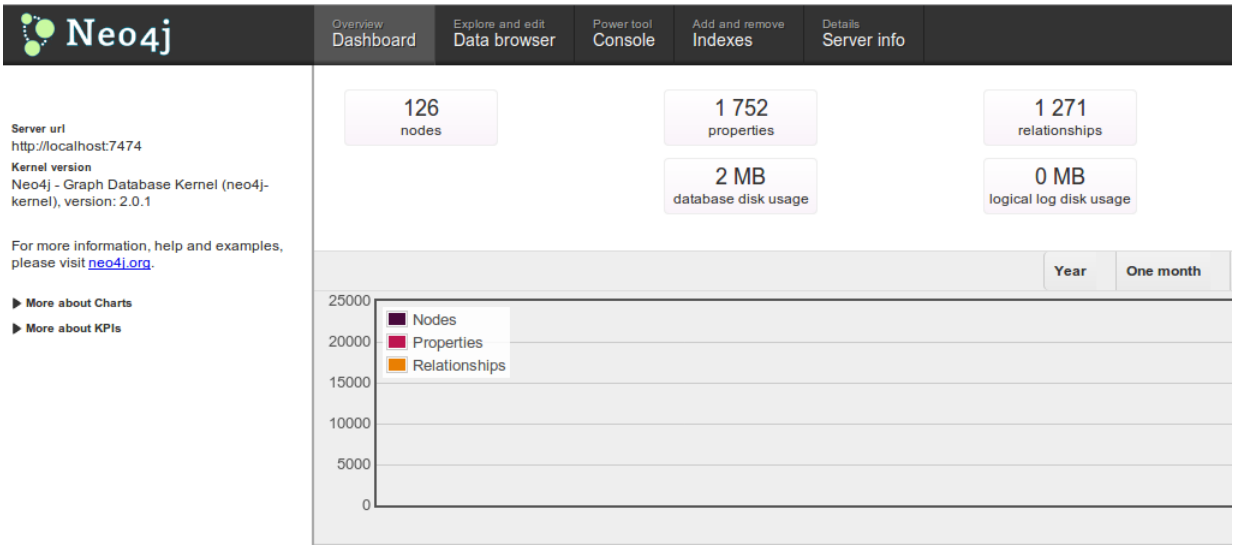


Figura 6.1: Relaciones y nodos generados inicialmente.

En principio, a través del escenario de estudio se busco determinar los parámetros apropiados para: *i*) generar las reglas de asociación, las cuales definen las relaciones del grafo y *ii*) realizar las recomendaciones de componentes más apropiados. Específicamente, se requiere encontrar los valores del Umbral de Soporte (US), Umbral de Confianza (UC), mínima probabilidad entre componentes-objetivos (PCO) y mínima probabilidad entre objetivos (POs). Para determinar estos valores, se siguió con la ejecución de un conjunto de experimentos —de ensayo y error—, donde el Grafo de Conocimiento sobre los datos previamente generados por expertos, fue ajustado variando los valores de US, UC, PCO y POs. De tal manera, que los componentes recomendados son los más apropiados.

El valor inicial de componentes a visualizar en el mashup (Cs), se fija basando se en el número de componentes según *Chowdhury* en [41], son los que conforman en promedio un mashup. El Cuadro 6.1 expone la configuración inicial de los parámetros para los escenarios de estudio:

Cuadro 6.1: Parámetros de configuración de los escenarios de estudio

Abreviatura	Parámetro	Cantidad	Descripción
Cs	Componentes	12	Número inicial de componentes a visualizar en el mashup.
CsC	Componentes basado en la Consulta	15	Número de componentes a recomendar basados en la consulta en lenguaje natural antes de recomendar por medio del grafo de comportamiento.
US	Umbral Soporte	0.5	Definición 4.3
UC	Umbral Confianza	0.5	Definición 4.4
PCO	Mínima Probabilidad entre Componentes-Objetivos	0.6	Probabilidad mínima que debe existir entre un componente y objetivo como parámetro para generar una consulta.
POs	Mínima Probabilidad entre Objetivos	0.6	Probabilidad mínima que debe existir entre objetivos como parámetro para generar una consulta.

6.3.2 Selección del Método de Evaluación

El objetivo de esta sección es identificar el contexto y la selección del método de evaluación adecuado para el presente proyecto. La caracterización del contexto se realiza a partir del conjunto de criterios propuestos en la sección 6.2. Por lo tanto, en primera instancia, es necesario explicar que el objeto de evaluación es la **Herramienta de Generación Automática de Mashup de Clientes en Dispositivos Móviles Orientado al Usuario Final**.

- **Contexto de evaluación:** Comprobar que el objeto de evaluación es una opción adecuada para que usuarios finales creen mashups.
- **Naturaleza del impacto esperado:** La naturaleza del impacto esperado es cualitativo, debido a que se desea capturar la apreciación del usuario en términos de la aceptación de los recursos visualizados y no en términos de valores medibles, con lo cual se pretende que usuarios finales puedan generar mashups de clientes.
- **Naturaleza del objeto de evaluación:** El objeto de evaluación es una herramienta de generación de mashups de clientes para usuarios finales.
- **Alcance del impacto del objeto de evaluación:** Efecto aplica a la creación de otros productos.
- **Madurez del objeto de evaluación:** El objeto de evaluación se encuentra listo para ser utilizado en diversas actividades que un usuario final realiza en el día a día.
- **Tiempo requerido para entender los principios delineados por el objeto de evaluación:** El tiempo requerido es bajo (máximo 15 min), debido a que los evaluadores no deben poseer conocimientos avanzados en la generación de mashups de clientes.

- **Tiempo requerido para llegar a ser un experto en el uso del objeto de evaluación:** El tiempo requerido es medio (30 minutos), pues las acciones son bastante intuitivas y no se requiere un conocimiento avanzado o experticia por parte del usuario final para generar un mashup que satisfaga sus requerimientos.
- **Capacidad de medición de la organización llevar a cabo la evaluación:** En este caso la organización que evaluará el objeto de evaluación cuenta con dos grupos de usuarios finales, ambos grupos tienen la capacidad de realzar todas las tareas de evaluación, dado no necesitan de conocimiento específico relacionado al tema de investigación.

Una vez realizado el análisis del contexto, el siguiente paso es seleccionar con mayor claridad el método de evaluación a emplear. Desde el punto de vista de los diferentes métodos de evaluación. El Cuadro 6.2 resume las condiciones que favorecen y no el uso de cada método.

Cuadro 6.2: Seleccion metodologia evaluación

Método de Evaluación	**Condiciones Favorables del método**	**Condición Presente**		**Porcentaje**
		Si	**No**	
Experimento Cuantitativo	Beneficios claramente cuantificables		X	16.67 %
	Disponibilidad del personal para participar en el experimento		X	
	Método/herramienta relacionada con una sola tarea/actividad		X	
	Beneficios directamente medibles del resultado de una tarea		X	
	Tiempo de aprendizaje relativamente corto	X		
	Deseo de realizar evaluaciones del método/herramienta independientes del contexto		X	
Estudio de Caso Cuantitativo	Beneficios cuantificables en un solo proyecto.		X	0 %
	Beneficios cuantificables antes del retiro del producto.		X	
	Procedimientos de desarrollo estables.		X	
	Personal con experiencia en mediciones.		X	
	Plazos de evaluación proporcionales con el tiempo de desarrollo de los proyectos de tamaño normal.		X	
Encuestas Cuantitativas	Beneficios no cuantificables en un solo proyecto.	X		66.6 %
	Existencia de una base de datos de logros de proyecto incluyendo: productividad, calidad, datos del método/herramienta.		X	
	Proyectos con experiencia en el uso del método/herramienta.	X		

Análisis de Característ. Por Chequeo	Amplio número de métodos/herramientas a evaluar.		X	50 %
	Periodos de tiempo cortos para realizar la evaluación.	X		
Análisis de Características por Estudio de Caso	Beneficios difíciles de cuantificar.	X		40 %
	Beneficios observables en un solo proyecto.	X		
	Procedimientos de desarrollo estable.		X	
	Población de usuarios del método/herramienta limitado.		X	
	Plazos de evaluación proporcionales con el tiempo de desarrollo de los proyectos de tamaño normal.		X	
Análisis de Características por Experimento	Beneficios difíciles de cuantificar.	X		100 %
	Beneficios directamente observables del resultado de una tarea.	X		
	Tiempo de aprendizaje relativamente corto.	X		
	Población de usuarios del método/herramienta muy variados.	X		
Análisis de Características por Encuesta	Beneficios difíciles de cuantificar	X		100 %
	Población de usuarios del método/herramienta muy variado.	X		
	Beneficios no observables en un solo proyecto.	X		
	Proyectos con experiencia en el uso del método/herramienta, o proyectos preparados para aprender sobre el método/herramienta.	X		
Análisis de Efectos Cualitativos	Disponibilidad de opiniones de expertos en evaluaciones de métodos/herramientas(similares)	X		25 %
	Ausencia de procedimientos de desarrollo estables		X	
	Requerimientos para combinar y relacionar métodos/herramientas		X	
	Interés en la evaluación de métodos/herramientas genéricas		X	
Benchmarking	Método/herramienta enfocado en máquina, no en humanos		X	50 %
	Salidas del método capaces de ser clasificadas en términos de algún "buen" criterio	X		

Después de revisar las condiciones favorables presentes y no presentes en cada uno de los métodos propuestos, se puede observar que los métodos de evaluación se reducen a dos: *Análisis de Características por Experimento y por Encuesta.*

Por lo tanto, para seleccionar adecuadamente el método fueron considerados los siguientes factores:

- Grupo de usuarios evaluadores
- Tiempo necesario para ejecutar la evaluación
- Nivel de confianza que un usuario pueda tener en los resultados de la evaluación
- Costo de la evaluación

En el análisis de características por experimento, la evaluación se basa en usuarios potenciales del sistema, el tiempo es corto (semanas), el riesgo es bajo y el costo es alto. En cuento al análisis de características por encuesta, una diferencia radical es que el tipo de evaluadores son usuarios que no representan a usuarios potenciales, el tiempo estimado es medio (varios meses), el riesgo es medio y el costo de igual forma es medio.

En este punto, teniendo en cuenta las consideraciones expuestas hasta el momento, el método de evaluación seleccionado es el *Análisis de Características por Encuesta*, debido a que el principal inconveniente radica en que no se cuenta con usuarios expertos y como se mencionó en el análisis del contexto, se pretende capturar la apreciación del usuario en términos de los recursos recomendados en la generación del mashups de clientes orientados a usuarios finales. A continuación, se presenta el plan de evaluación propuesto con base en la encuesta cualitativa.

6.4 Planeación de la Evaluación

DESMET recomienda organizar la evaluación como un experimento formal. De esta manera se definen dos tipos de pruebas: La primera, es una evaluación de relevancia a partir de los mashups generados por parte del usuario. La segunda, es una evaluación del producto orientada a determinar el grado de aceptación del objeto de evaluación por parte de usuarios finales. A continuación se describe la caracterización de la evaluación.

6.4.1 Caracterización de los Participantes

Para llevar a cabo la evaluación se considera solo el rol de usuario final. Este rol es asumido por personas cuyo perfil es idóneo para el presente experimento. A continuación se presenta la información general de los participantes de esta evaluación.

La evaluación contó con 30 participantes entre 20 y 26 años de edad, de la ciudad de Popayán y entre ellos, personas pertenecientes al programa de ingeniería electrónica y telecomunicaciones de la Universidad del Cauca.

Se citaron a los colaboradores en diferentes momentos a las instalaciones de la Universidad del Cauca y a un recinto privado, donde se realizaron una exposición breve del proyecto de grado, sus objetivos y la arquitectura del sistema. Para cada evaluación se expone el criterio de conformidad a evaluar y los factores a tener en cuenta, además los evaluadores pueden realizar preguntas, sugerencias o comentarios que son consignados.

6.4.2 Evaluación de Relevancia

Esta evaluación consiste en determinar la relevancia de los mashups generados por el sistema. Para comenzar, el objeto de evaluación es permitir que usuarios sin conocimientos en composición o creación de mashups, tengan la posibilidad de hacerlo para satisfacer sus necesidades de información.

A través de la evaluación, el presente proyecto busca confirmar o rechazar la siguiente hipótesis:

- *H01: Los usuarios finales consideran que los mashups generados por Mobshapp son relevantes para su tarea de búsqueda.*

Como el objetivo de la hipótesis es medir la relevancia de los mashups, se consideran las calificaciones (rankings) que el usuario ha asignado a cada uno de los mashups generados. La Figura 6.2 presenta un ejemplo de esta calificación.

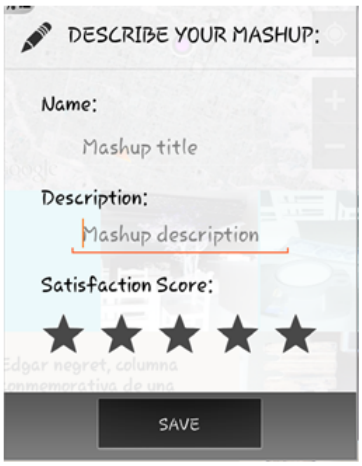


Figura 6.2: Interfaz de calificación de mashups.

6.4.3 Evaluación del Producto

Esta evaluación consiste en determinar el grado de aceptación del objeto de evaluación por parte de usuarios finales. Para comenzar, se seleccionan las características relevantes a tener en cuenta en la evaluación. Esta selección es una actividad subjetiva y depende de la experiencia de los usuarios finales, además de los puntos de vista e intereses de las personas a evaluar.

A través de esta evaluación, el presente proyecto busca comprobar o refutarla siguiente hipótesis:

- *H02: Los usuarios evidencian una opinión positiva con respecto al cumplimiento de sus necesidades de información (requerimientos) por medio del uso de Mobshapp.*

Ya que el objetivo de la hipótesis es medir el grado de percepción sobre el objeto de evaluación, se emplea una encuesta con el fin de conocer aspectos relevantes sobre si los usuarios son capaces de generar mashups de forma intuitiva, siendo conscientes que con sus interacciones mejoran el proceso de generación de mashups.

De esta manera, la encuesta (ver ANEXO B) está conformada por 8 preguntas que se componen de: Una descripción de la característica a evaluar, una estimación sobre su grado de importancia, una escala de evaluación y un espacio para observaciones; tal como se muestra en el Figura 6.3:

1. Uso intuitivo de la herramienta	
Descripción: Esta característica mide el grado de facilidad e intuición de la herramienta para permitir que usuarios sin conocimientos técnicos generen sus propios mashups a través de ella, sin la necesidad de invertir mucho tiempo indagando cómo funciona la herramienta o que alguna persona instruya en su uso.	
Pregunta: En qué grado considera que Mobshapp facilita el proceso de generación de mashups?	
Grado de importancia: Altamente deseable.	
Escala:	
Ninguno	
Muy poco	
Parcialmente	
En alto Grado	X
Completamente	
Observaciones:	

Figura 6.3: Ejemplo encuesta.

Esta encuesta se fundamenta en las evaluaciones expuesta por Inostroza [63] y Omaña [64]. Por una lado, Inostroza utiliza una evaluación Heurística para dispositivos móviles, en el cual se hace una inspección minuciosa a las interfaces y/o sistemas, con el fin de asegurarse que los elementos se adaptan al contexto. Por otra parte, Omaña busca integrar diversos modelos de calidad como la ISO9126, a partir de una evaluación al producto y al proceso de desarrollo; en el cual, la evaluación del producto tiene en cuenta aspectos de la funcionalidad, fiabilidad, usabilidad, eficiencia, entre otro.

6.5 Interpretación de Resultados

Las dos evaluaciones son conducidas a través de una Prueba t; la cual discrimina dos tipos: pareada y desapareada. La primera consiste en una prueba de mediciones repetitivas, por ejemplo: un mismo grupo que ha sido evaluado en dos ocasiones diferentes. La segunda radica en una prueba de mediciones aleatorias, independientes e idénticamente distribuidas, por ejemplo: es evaluado el efecto de un tratamiento médico sobre un grupo de personas.

Para estas evaluaciones, la prueba t desapareada es la más adecuada puesto que consiste en una prueba de mediciones aleatorias, donde el grupo de sujetos de estudio fueron seleccionados independientemente. De esta manera, las evaluaciones contaron con 30 participantes teniendo en cuenta la teoría de pequeñas muestras, este número genera una distribución de muestreo adecuada, debido a que, cuando este numero de muestras se aproximan a 30 o tienden a número infinito, la dispersión correspondiente a la curva t disminuye aproximándose a la curva estándar, razón por la cual, se conoce como curva z a la curva t cuando N tiende a infinito.

Para este tipo de análisis es necesario aclarar primero los conceptos de: *Alfa*, *p-Valor* y *Significancia Estadística*.

Alfa: Representa el nivel de significancia, es decir, el nivel de alfa es la probabilidad de rechazar la hipótesis nula cuando la hipótesis nula es verdadera. Este concepto tiene un valor comprendido entre [0,1]. Además, el valor de alfa está asociado a un nivel de confianza de la prueba. A continuación se enumeran algunos niveles de confianza con sus valores correspondientes de alfa:

- Para obtener resultados con un nivel de confianza del 90 %, el valor de alfa es 0,1 (1-0,90 = 0,10).
- Para obtener resultados con un nivel de confianza del 95 %, el valor de alfa es 1-0,95 = 0,05.
- Por lo general, el valor de alfa puede adoptar un rango diverso de valores. Por lo tanto, para resultados con un nivel de confianza C %, el valor de alfa es: 1 - C/100.

p-Valor (p): El p-Valor es la probabilidad de obtener un resultado estadístico de la prueba, bajo el supuesto que la hipótesis nula es verdadera. A continuación, se expone la interpretación del p-Valor con base en un nivel de significancia del 10 %:

- $p < 0.01$: evidencia una muy fuerte suposición en contra de la hipótesis nula.
- $p = 0.01$: evidencia una fuerte suposición en contra de la hipótesis nula.
- $0.01 < p < 0.05$: evidencia una suposición moderada en contra de la hipótesis nula.
- $0.05 < p < 0.1$: evidencia una baja suposición en contra de la hipótesis nula.
- $p > 0.1$: ninguna suposición en contra de la hipótesis nula.

Significancia Estadística: Para determinar si un resultado observado es estadísticamente significativo, se comparan los valores de alfa y p. Lo que conlleva a dos posibilidades:

- Si p es menor o igual a alfa: En este caso se rechaza la hipótesis nula. Cuando esto ocurre se dice que el resultado es estadísticamente significativo.
- Si p es mayor que alfa. En este caso, no se rechaza la hipótesis nula. Cuando esto ocurre se dice que el resultado no es estadísticamente significativo.

Considerando un nivel de significancia de 0.05 (correspondiente al de mayor aceptación en la academia y la industria), a continuación se presentan los resultados conseguidos en cada evaluación descrita en la sección anterior.

6.5.1 Evaluación de Relevancia

Dado que los valores de calificación se encuentran en el rango de 1 a 5 puntos, se procede a transformar la escala propuesta a valores legibles para las personas, como se presenta en el Cuadro 6.3:

Cuadro 6.3: Equivalencia numérica de las escalas - Evaluación de Relevancia.

Calificación	Escala
5	Muy de acuerdo
4	De acuerdo
3	Ni de acuerdo ni en desacuerdo
2	En desacuerdo
1	Muy en desacuerdo

Se han tomado estos valores de tal forma que la opción "*Ni de acuerdo ni en desacuerdo*" sea un valor neutro (3), las opciones "*Muy de acuerdo*" y "*De acuerdo*" reflejen resultados positivos (>3), y las opciones "*En desacuerdo*" y "*Muy en desacuerdo*" reflejen resultados negativos (<3).

En primera instancia se visualizan las frecuencias de las calificaciones obtenidas para cada mashups generado.

Posteriormente, se realiza una estadística descriptiva a través del Cuadro 6.4, con el fin de describir apropiadamente las características relevantes de esta evaluación. Sin embargo, aunque hay tendencia a generalizar todas las muestras, las primeras conclusiones obtenidas tras un análisis descriptivo, corresponden a un resultado preliminar y aproximado.

Cuadro 6.4: Estadística descriptiva - Evaluación de Relevancia.

Escala	Frecuencia	Porcentaje	Porcentaje Acumulado
Muy de acuerdo	5	16.7	16.7
De acuerdo	14	46.7	63.4
Ni de acuerdo ni en desacuerdo	11	36.6	100.0
En desacuerdo	0	0.0	100.0
Muy en desacuerdo	0	0.0	100.0
TOTAL	30	100.0	

Cuadro 6.5: Tabla de Frecuencias - Evaluación de Relevancia.

Escala	Calificaciones
Muy de acuerdo	5
De acuerdo	14
Ni de acuerdo ni en desacuerdo	11
En desacuerdo	0
Muy en desacuerdo	0
Media	3.8
Moda	4
Desviación Típica	0.714

De acuerdo al valor de la media, la calificación de los mashups tiende a 3.8 correspondiente a la escala "De acuerdo". Para roborar dicho valor, la moda indica que la mayoría de usuarios

consideró que estaba "De acuerdo" con el resultado presentado en los mashups, es decir, los resultados satisfacen en gran proporción los requerimientos.

El valor de la desviación típica nos indica la dispersión de las opiniones de los usuarios finales, según el Cuadro6.5, se generaron opiniones similares en la evaluación.

De esta manera, teniendo en cuenta que tres (3) corresponde al punto neutro de la evaluación y que no existieron valores menores a tres (3), se formula la siguiente hipótesis estadística.

- *H01n (Hipótesis nula): La relevancia promedio de los mashups generados es igual a 3.*
- *H01a (Hipótesis alternativa): La relevancia promedio de los mashups generados es mayor a 3.*

Para validar la hipótesis nula, se obtuvieron los resultados que se observan en el Cuadro 6.6.

Cuadro 6.6: Prueba T - Evaluación de Relevancia.

	Valor de Prueba = 3					
	t	**gl**	**p**	**Dm**	**Ic = 95 %**	
					Inf.	**Sup.**
Calif.	6,134	29	0,0000011	0.8	0,533	1.066
gl: Grado de libertad. Ic: Intervalo de confianza. Dm: Diferencia de medias.						

De acuerdo a los valores presentados en el Cuadro 6.6, tenemos que p es menor a 0.05 (nivel de significancia). Esto significa, que el promedio de evaluación de los mashups generados por usuarios finales son estadísticamente significativos y se rechaza la hipótesis nula (H1n). Para una perspectiva más general, la Figura 6.4 grafica el porcentaje de frecuencia de las calificaciones:

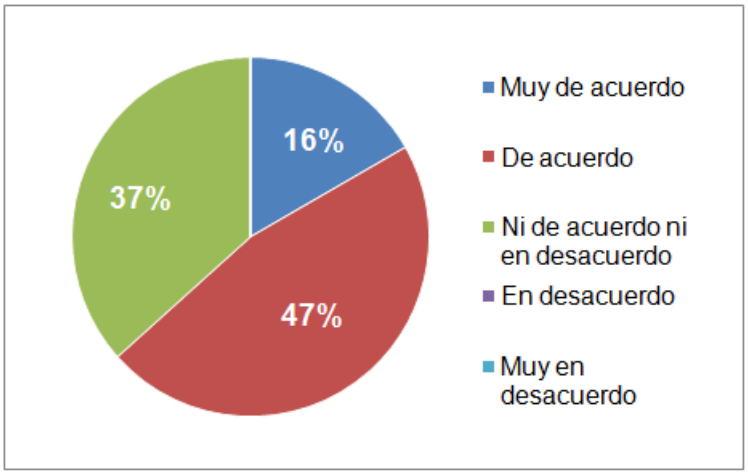


Figura 6.4: Porcentaje de frecuencia - Evaluación de Relevancia.

Respecto a la totalidad de las calificaciones conseguidas, el 16 % corresponde a la opción "*Muy de acuerdo*", el 47 % a "*De acuerdo*" y el 37 % a "*Ni de acuerdo ni en desacuerdo*". Por lo tanto,

los resultados positivos en la evaluación constituyen un 63 %, las opiniones neutras constituyen el 37 % restante, y no se presentaron opiniones negativas. Se puede considerar que la evaluación en general resultó exitosa.

6.5.2 Evaluación del Producto

En primera instancia se visualizan las frecuencias de los resultados obtenidos en cada pregunta de la encuesta.

Cuadro 6.7: Tabla de frecuencia pregunta 1.

Escala	Frecuencia	Porcentaje (%)	Porcentaje Acumulado (%)
Completamente	7	23.3	23.3
En Alto Grado	18	60.0	83.3
Parcialmente	5	16.7	100.0
Muy poco	0	0	100.0
Ninguno	0	0	100.0
TOTAL	30	100.0	

Cuadro 6.8: Tabla de frecuencia pregunta 2.

Escala	Frecuencia	Porcentaje (%)	Porcentaje Acumulado (%)
Completamente	2	6.7	6.7
En Alto Grado	15	50.0	56.7
Parcialmente	13	43.3	100.0
Muy poco	0	0.0	100.0
Ninguno	0	0.0	100.0
TOTAL	30	100.0	

Cuadro 6.9: Tabla de frecuencia pregunta 3.

Escala	Frecuencia	Porcentaje (%)	Porcentaje Acumulado (%)
Completamente	10	33.3	33.3
En Alto Grado	16	53.4	86.4
Parcialmente	4	13.3	100.0
Muy poco	0	0.0	100.0
Ninguno	0	0.0	100.0
TOTAL	30	100.0	

Cuadro 6.10: Tabla de frecuencia pregunta 4.

Escala	Frecuencia	Porcentaje (%)	Porcentaje Acumulado (%)
Completamente	5	16.7	16.7
En Alto Grado	17	56.7	73.4
Parcialmente	8	26.6	100.0
Muy poco	0	0.0	100.0
Ninguno	0	0.0	100.0
TOTAL	30	100.0	

Cuadro 6.11: Tabla de frecuencia pregunta 5.

Escala	Frecuencia	Porcentaje (%)	Porcentaje Acumulado (%)
Completamente	7	23.3	23.3
En Alto Grado	16	53.4	76.7
Parcialmente	7	23.3	100.0
Muy poco	0	0.0	100.0
Ninguno	0	0.0	100.0
TOTAL	30	100.0	

Cuadro 6.12: Tabla de frecuencia pregunta 6.

Escala	Frecuencia	Porcentaje (%)	Porcentaje Acumulado (%)
Completamente	6	20.0	20.0
En Alto Grado	14	46.7	66.7
Parcialmente	10	33.3	100.0
Muy poco	0	0.0	100.0
Ninguno	0	0.0	100.0
TOTAL	30	100.0	

Cuadro 6.13: Tabla de frecuencia pregunta 7.

Escala	Frecuencia	Porcentaje (%)	Porcentaje Acumulado (%)
Completamente	4	13.3	13.3
En Alto Grado	20	66.7	80.0
Parcialmente	6	20.0	100.0
Muy poco	0	0.0	100.0
Ninguno	0	0.0	100.0
TOTAL	30	100.0	

Cuadro 6.14: Tabla de frecuencia pregunta 8.

Escala	Frecuencia	Porcentaje (%)	Porcentaje Acumulado (%)
Completamente	16	53.3	53.3
En Alto Grado	14	46.7	100.0
Parcialmente	0	0.0	100.0
Muy poco	0	0.0	100.0
Ninguno	0	0.0	100.0
TOTAL	30	100.0	

A continuación, para realizar algunos análisis se requiere una transformación de la escala de evaluación propuesta a valores numéricos, como se presenta en el Cuadro 6.15.

Cuadro 6.15: Equivalencia numérica de las escalas - Evaluación del Producto.

Escala	Calificación
Completamente	5
En Alto Grado	4
Parcialmente	3
Muy poco	2
Ninguno	1

Se han tomado estos valores de tal forma que la escala "*Parcialmente*" exprese un grado de aceptación neutra (3), las escalas "*Completamente*" y "*En alto Grado*" reflejen un alto grado de aceptación (>3) y por último, las opciones "*Muy poco*" y "*Ninguno*" reflejen un grado de aceptación bajo (<3) sobre la herramienta.

Así, a partir de las frecuencias obtenidas en las calificaciones de cada pregunta y la equivalencia numérica de las escalas, se realiza una estadística descriptiva a través del Cuadro 6.16 con el fin de describir apropiadamente las características relevantes de esta evaluación.

Cuadro 6.16: Estadísticas descriptivas – Evaluación de Producto.

Escala	P1	P2	P3	P4	P5	P6	P7	P8
Completamente	7	2	10	5	7	6	4	16
En Alto Grado	18	15	16	17	16	14	20	14
Parcialmente	5	13	4	8	7	10	6	0
Muy poco	0	0	0	0	0	0	0	0
Ninguno	0	0	0	0	0	0	0	0
Media	4.06	3.63	4.2	3.9	4	3.86	3.93	4.53
Moda	4	4	4	4	4	4	4	5
Desviac. Típica	0.63	0.52	0.66	0.66	0.69	0.73	0.58	0.50

De acuerdo al valor de la media, se observa que la pregunta P2 presenta una opinión menos favorable, con una media de 3.63 correspondiente a la escala "*En alto Grado*". No obstante, la mayoría de preguntas presentan una opinión más favorables, con una media con valor superior a 3.9. Para roborar, la moda indica que la mayoría de usuarios tienen una opinión favorable "*En Alto Grado*", es decir, consideran adecuada la herramienta para la generación de mashups. El valor de la desviación típica nos indica la dispersión de las opiniones de los usuarios sobre las aspectos evaluados, según el Cuadro 6.16 se generaron opiniones más dispares en la pregunta P6.

De esta manera, teniendo en cuenta que tres (3) corresponde al punto neutro, se formulan las siguientes hipótesis estadísticas:

- *H02n (Hipótesis nula): El grado de aceptación promedio de Mobshapp es igual a 3.*
- *H02a (Hipótesis alternativa): El grado de aceptación promedio de Mobshapp es mayor a 3.*

Para validar la hipótesis nula, se obtuvieron los resultados que se observan en el Cuadro6.16 y considerando un nivel de significancia de 0.05, se consiguieron los resultados que se observan en el Cuadro 6.17.

Cuadro 6.17: Prueba T - Evaluación de Producto.

Calif.	Valor de Prueba = 3					
	t	gl	p	Dm	Ic = 95 %	
					Inf.	Sup.
P1	9.133	29	0.0001	1.07	0.83	1.31
P2	5.641	29	0.0001	0.63	0.40	0.86
P3	9.893	29	0.0001	1.20	0.95	1.45
P4	7.449	29	0.0001	0.90	0.65	1.15
P5	7.918	29	0.0001	0.97	0.72	1.22
P6	6.50	29	0.0010	0.87	0.59	1.14
P7	8.764	29	0.0001	0.93	0.72	1.15
P8	16.55	29	0.0001	1.53	1.34	1.72

gl: Grado de libertad.
Ic: Intervalo de confianza.
Dm: Diferencia de medias.

De acuerdo a los valores presentados en el Cuadro 6.17, se tiene que p es menor a 0.05 (nivel de significancia). Esto significa, que el grado de aceptación promedio sobre las preguntas evaluadas por usuarios finales son estadísticamente significativos y se rechaza la hipótesis nula (H1n).

Así, para una perspectiva más general de los resultados la Figura 6.5 representa una distribución de las evaluaciones y la Figura 6.6 grafica el porcentaje de frecuencia de las calificaciones:

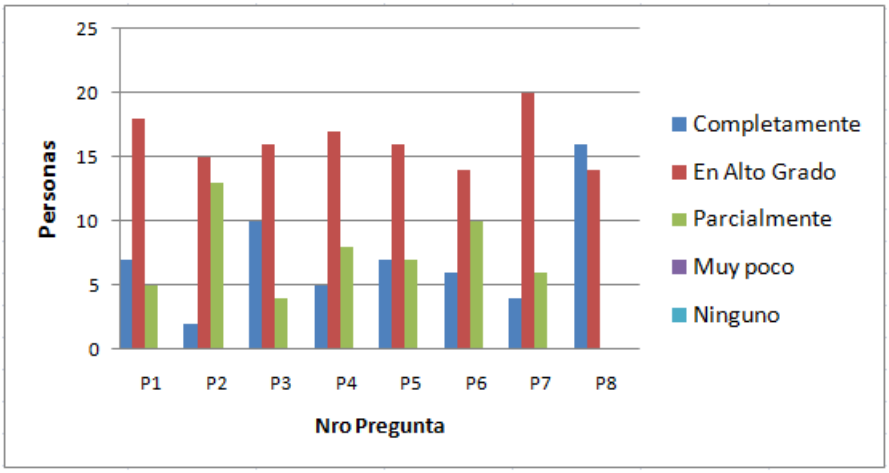


Figura 6.5: Distribución de las calificaciones – Evaluación de Producto.

A partir de la figura anterior, se obtienen las siguientes observaciones:

- La pregunta con opinión menos favorable, se presento en la P2, indicando que la ayuda contextual no es adecuada y del agrado para los usuarios finales. No obstante, la mayoría de preguntas presentan una opinión más favorable.
- En la P6, podemos observar una dispersión de las opiniones de los usuarios respecto a las otras preguntas.
- En la P1, se observa que los usuarios consideran adecuada la herramienta para la generación de mashups.
- En términos generales, los resultados obtenidos en las preguntas P3, P4, P5, P7 y P8 fueron los resultados que buscamos, lo cual me indica, que los tipos de acciones influyeron en las recomendaciones y por ende, facilito la tarea de búsqueda del usuario.

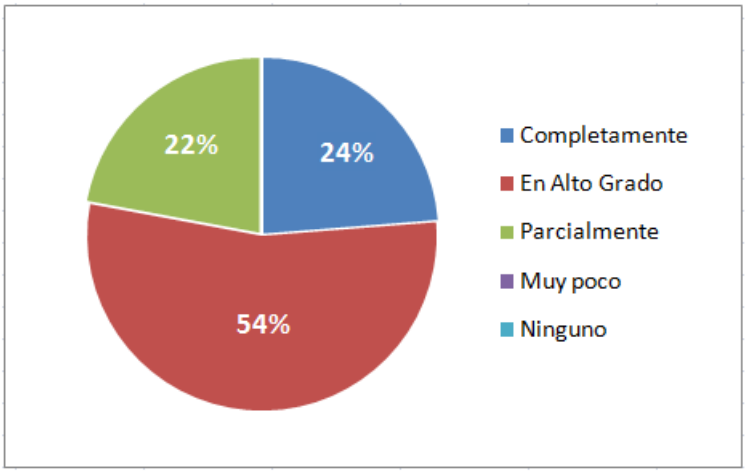


Figura 6.6: Porcentaje de frecuencia - Evaluación de Producto.

Respecto a la totalidad de las calificaciones conseguidas, Figura 6.6: el 54 % corresponde a la opción "En Alto Grado", el 24 % a "Completamente" y el 22 % a "Parcialmente". Por lo tanto, los resultados positivos en la evaluación constituyen un 78 %, las opiniones neutras constituyen el 22 % restante, y no se presentaron opiniones negativas. Teniendo en cuenta los resultados conseguidos, se puede considerar que la herramienta tiene una alto grado de aceptación sobre la mayoría de usuarios.

6.6 Resumen

En este capítulo se abordó una descripción detallada del proceso de evaluación llevado a cabo para el sistema propuesto. Inicialmente, se presentó la metodología de evaluación escogida, que toma como base un conjunto de criterios para seleccionar el método de evaluación más apropiado a partir del análisis del contexto del sistema.

Se realizaron dos tipos de evaluaciones con la colaboración de un grupo de usuarios finales: La primera consistió en una prueba del sistema, que permitió conocer el grado de relevancia de los mashups generados; y la segunda evaluación consistió en una apreciación del producto, que permitió determinar el grado de aceptación de la herramienta para desarrollar mashups. Así, para cada prueba se consigna tanto los aspectos metodológicos de la evaluación efectuada, como el análisis de los resultados obtenidos.

Capítulo 7

Conclusiones y Trabajos Futuros

7.1 Introducción

En la actualidad existe un crecimiento en el desarrollo de mashups a través de herramientas Web, incentivando a usuarios finales no a solo consumir contenido sino a crear sus propias aplicaciones, lo cual crea un ecosistema propicio para el desarrollo y consumo de aplicaciones. Después de haber realizado un análisis detallado de las diferentes plataformas y modelos para la creación de mashups, es evidente que estos aún no se centran en los usuarios finales, debido a que requieren ejecutar actividades que un desarrollador realiza comúnmente.

Con el fin de presentar una solución a este problema, el enfoque propuesto en este trabajo abordo dos temáticas: la creación de mashups y recomendación de recursos. El primero se desarrolló, a partir del análisis de la categorización y generación de mashups. Y el segundo, se desarrolló a partir del estudio de los mecanismos de recomendación propuestos en los trabajos relacionados, que expusieron premisas importantes en esta área. Este análisis, permitió proponer un modelo y un paradigma para la generación de mashups, que permiten la integración automática de componentes; y un mecanismo de recomendación a través de patrones de conocimiento de composición (patrones de co-ocurrencia) basados en los objetivos de los recursos. Agregando a lo anterior, las temáticas mencionadas toman como base el comportamiento del usuario con el sistema durante el desarrollo del mashup.

Así, este capítulo describe inicialmente las conclusiones a las que se llegó con el desarrollo del proyecto y finalmente, propone algunos trabajos futuros que señalan el camino que aún resta por recorrer en este ámbito de estudio.

7.2 Conclusiones

Al finalizar este proyecto se desarrollo un prototipo denominado *Mobshapp*, el cual se basa en técnicas, algoritmos y paradigmas de diseño mencionadas al interior de este documento. Mobshapp se destaca frente a los procesos de generación de mashups estudiados, debido a la integración de los entornos de desarrollo y de ejecución; los cuales realizan sus procesos coordinadamente en función de las acciones ejecutadas por los usuarios finales, permitiendo identificar los objetivos de su búsqueda y proporcionando recomendaciones iterativas. Además, este prototipo proporciona simplicidad y facilidad de uso a usuarios finales para generar mashups a través de su interacción con el sistema, obviando procesos propios de composición (recuperación, selección, integración y

mapeo de datos).

A continuación se describen las principales conclusiones obtenidas a partir de la ejecución del presente proyecto.

- Hasta la fecha no existe una estandarización hacia la categorización de los mashups y de sus herramientas, debido a que no hay una concepción única en los parámetros, que defina dicha clasificación.

- A partir del análisis de la categorización y generación de mashups, se determinó que la mayoría de herramientas se encuentran orientadas a usuarios con conocimiento en composición.

- La mayoría de herramientas actuales son de tipo Semiautomática-Asistida, es decir, este proceso automatiza parcialmente actividades propias de la composición; tales como: seleccionar, adaptar y ensamblar componentes de tal forma que proporciona asistencia y orientación en el proceso de desarrollo a través de actividades de recomendación de componentes y patrones de composición. Sin embargo, en este tipo de procesos las personas intervienen en los flujos de integración, añadiendo requerimientos en el proceso de desarrollo y validación de los resultados finales.

- Generalmente, las plataformas presentan dos entornos: el primero de diseño y el segundo de ejecución. Sin embargo, en el primer entorno, las herramientas caen en el error de incorporar al usuario, sin conocimientos técnicos, en tareas de selección de componentes y en la definición de un flujo de control, actividades no propicias para ellos.

- Usuarios finales no están interesados en realizar procesos complejos para llevar a cabo tareas de búsqueda que den solución a su problema particular; debido a que, cada vez que ellos cambien sus requerimientos se verán obligados a realizar modificaciones en el entorno de diseño del mashup y desplegar nuevamente los cambios en el entorno de ejecución.

- La mayoría de herramientas hacen uso de los dispositivos móviles como entorno de ejecución y no como entorno de diseño.

- Para alcanzar un proceso transparente en la generación de mashups orientado al usuario final, es importante que las herramientas busquen los mecanismos, que permitan que personas sin conocimientos técnicos, desarrollen sus propias aplicaciones a través de la reutilización de conocimiento.

- Dado que los mashups son aplicaciones basados en componentes, es necesario definir un modelo de componente universal que permita representar cada recurso de la Web. Además, la complejidad de cada componente debe llevarse a cabo internamente, de tal manera que los usuarios finales que hacen uso de este, puedan hacerlo de forma intuitiva.

- La adaptación de la interfaz gráfica del mashup final, conocida como sincronización, se consigue a través de la implementación de eventos que representan las interacciones del usuario. Así, el concepto de evento permite comunicar componentes y su estado dentro de la aplicación, además de definir los posibles flujos de datos a partir de la interacción del usuario, proporcionando mecanismos que simplifiquen los procesos propuestos por otros enfoques, como BPEL.

- Los procesos de mapeo y el flujo de datos deben ser automatizados, para permitir que usuarios finales participen en el proceso de desarrollo de mashups de clientes.
- La implementación de un sistema de recomendación a través de la interacción del usuario y los objetivos que describen a los componentes en uso, permiten que el usuario no se desenfoque de su tarea de búsqueda.
- El enfoque propuesto sobre dispositivos móviles, facilita la tarea de búsqueda y visualización del contenido de los recursos, que satisfacen los requerimientos de búsqueda. Esto, debido a la naturaleza ubicua de los dispositivos móviles, que permiten desarrollar mashups paulatinamente, es decir, en cualquier momento y espacio donde se presente una necesidad de información.
- A través de las interacciones del usuario con el sistema es posible reducir la ambigüedad de las consultas realizadas en lenguaje natural y determinar un dominio de búsqueda específico, que satisfaga los objetivos del usuario.
- Las pruebas de relevancia realizadas durante la generación de mashups, mostraron que el 54 % correspondiente a la escala "De acuerdo", refleja que las recomendaciones sugeridas por Mobshapp, fueron acordes a los objetivos de búsqueda. No obstante, solo el 13 % de las calificaciones reflejan que los resultados cumplieron completamente sus objetivos. Por lo tanto, se puede considerar que la evaluación en general resultó exitosa.
- Las pruebas del producto se realizaron una vez terminada la evaluación de relevancia. De acuerdo al resultado de la encuesta realizada, el grado de aceptación promedio de los usuarios es bastante favorable, en cuanto a la facilidad e intuición de la herramienta para permitir que usuarios sin conocimientos técnicos, generen sus propios mashups.
- Usuarios finales consideran en alto grado útiles los tipos de acciones propuestos en Mobshapp, para llevar a cabo su tarea de búsqueda. Por lo tanto, se evidencia que a través de las interacciones del usuario, se permiten inferir en los requerimientos y refinamientos de la consulta en lenguaje natural durante el proceso de generación de mashups.
- En resumen, el resultado de las evaluaciones del producto corroboran los resultados generados en la prueba de relevancia durante la generación de mashups, considerando adecuada el enfoque propuesto en el presente trabajo.
- Mobshapp permite que el usuario ejecute sus tareas de búsqueda, sin tener que preocuparse por otras actividades.

7.3 Trabajos Futuros

En resumen, esta investigación ha aportado soluciones al problema de generar mashups orientados a usuarios sin conocimientos técnicos, a través de la integración automática de recursos Web a partir de una consulta inicial en lenguaje natural, toman como base las interacciones del usuario. Además, el enfoque propuesto combina los entornos de diseño y ejecución con el fin de interactuar con los resultados que satisfagan los requerimientos de búsqueda, sin tener que volver a diseñar los mashups una vez ejecutados. En este sentido, con relación al campo de estudio de esta investigación, se propone los siguientes trabajos futuros:

- Extender el desarrollo de mashups tanto de clientes como empresariales.
- Realimentar el sistema de recomendación a través de la detección de emociones, como parámetros de entrada, de tal manera que permitan identificar los requerimientos de búsqueda.
- Incorporar consultas transaccionales en el proceso de desarrollo de mashups.
- Incorporar información de redes sociales en los mecanismos de recomendación propuestos, debido a que solo se contempla las redes sociales para realizar el registro en el sistema y con el objetivo de proporcionar recomendaciones en el contexto y en la información social.
- Actualmente Mobshapp, solo contempla 4 tipos de repositorios de información para soportar el sistema de recomendación. Por lo tanto, es posible mejorar las recomendaciones con la extensión a otros tipos de repositorios.
- Extender el soporte de Mobshapp a otras plataformas móviles. Se plantea la necesidad de migrar la aplicación a otras plataformas móviles, como pueden ser iOS, Windows Phone, entre otros. Esto con el fin de observar los efectos que puede generar el tipo del mercado o plataforma, sobre el sistema propuesto.
- Evaluar la herramienta propuesta en un entorno real, con el fin de entrenar el sistema de recomendación a través de un conjunto considerable de usuarios finales.

Referencias

[1] S. Pietschmann, T. Nestler, and F. Daniel, "Application composition at the presentation layer: Alternatives and open issues," *ACM*, vol. 3, pp. 333–434, nov 2010.

[2] F. Daniel, S. Soi, and F. Casati, "From mashup technologies to universal integration: Search computing the imperative way." in *SeCO Workshop*, ser. Lecture Notes in Computer Science, S. Ceri and M. Brambilla, Eds., vol. 5950. Springer, 2009, pp. 72–93.

[3] R. Trapero Burgos, D. Suárez Fuentes, J. M. del Alamo Ramiro, A. L. Martín, Y. S. Martín García, I. Ordás Arnal, l. Martínez Reol, and J. C. Yelmo García, "Next generation mashups: Cómo crear mis propios servicios en un mundo convergente." in *Actas XVIII Jornadas Telecom I+D*, vol. 3. Bilbao: Universidad del Pais Vasco, 2008, pp. 1–4.

[4] J. C. Yelmo, J. M. del Álamo, R. Trapero, and Y.-S. Martín, "A user-centric approach to service creation and delivery over next generation networks," *Comput. Commun.*, vol. 34, pp. 209–222, 2011.

[5] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds. Springer, 2011, no. 978-0-387-85819-7.

[6] L. J. Suarez Meza, "Service sonsumption survey," vol. 3, pp. 2–14, 2011.

[7] L. J. S. Meza, L. A. R. Potosí, J. C. Corrales, and O. R. Rocha, "Wishful and wisdom aware composing - a full user-centric approach to create ngm." in *WEBIST*. SciTePress, 2012, pp. 803–806.

[8] M. Lennighan. (2011, Mayo) Number of phones exceeds population of world. [Online]. Available: http://www.totaltele.com/view.aspx?ID=464922

[9] E. Agichtein, E. Gabrilovich, and H. Zha, "The social future of web search: Modeling, exploiting, and searching collaboratively generated content," *IEEE Data Eng. Bull.*, vol. 32, p. 10, 2009.

[10] D. Florian, Y. Jin, B. Boualem, F. Casati, M. Matera, and S.-P. Regis, "Understanding ui integration: A survey of problems, technologies, and opportunities," *IEEE Internet Computing*, vol. 11, no. 3, pp. 59–66, 2007.

[11] D. Florian, C. Rodriguez, S. Roy Chowdhury, H. R. Motahari Nezhad, and F. Casati, "Discovery and reuse of composition knowledge for assisted mashup development," in *Proceedings of the 21st international conference companion on World Wide Web*, ser. WWW '12 Companion. New York, NY, USA: ACM, 2012, pp. 493–494.

[12] L. Nieto Peñalver, "Metodología de diseño para el desarrollo de mashups semánticos," Master's thesis, Universidad Nacional de La Plata, 2012.

[13] A. V. Riabov, E. Boillet, M. D. Feblowitz, Z. Liu, and A. Ranganathan, "Wishful search: interactive composition of data mashups," in *Proceedings of the 17th international conference on World Wide Web*, ser. WWW '08. New York, NY, USA: ACM, 2008, pp. 775–784.

[14] O. A. H. Hassan, L. Ramaswamy, and J. A. Miller, "Enhancing scalability and performance of mashups through merging and operator reordering," in *2010 IEEE International Conference on Web Services*. IEEE, 2010, pp. 171–178.

[15] S. Soi, "International doctorate school in information and communication technologies," Ph.D. dissertation, International Doctorate School in Information and Communication Technologies (DISI) - University of Trento, 2013.

[16] N. Zang and M. B. Rosson, "What's in a mashup? and why? studying the perceptions of web-active end users," *Visual Languages and Human-Centric Computing, IEEE Symposium on*, vol. 0, pp. 31–38, 2008.

[17] V. Hoyer and M. Fischer, "Market overview of enterprise mashup tools," in *ICSOC*, ser. Lecture Notes in Computer Science, A. Bouguettaya, I. Krüger, and T. Margaria, Eds., vol. 5364, 2008, pp. 708–721.

[18] B. Beemer and D. Gregg, "Mashups: A literature review and classification framework," *Future Internet*, vol. 1, no. 1, pp. 59–87, 2009.

[19] D. Florian, A. Koschmider, T. Nestler, M. Roy, and A. Namoun, "Toward process mashups: Key ingredients and open research challenges," in *Proceedings of the 3rd and 4th International Workshop on Web APIs and Services Mashups*, ser. Mashups '09/'10. New York, NY, USA: ACM, 2010, pp. 9:1–9:8.

[20] Z. Zhao, S. Bhattarai, J. Liu, and N. Crespi, "Mashup services to daily activities: end-user perspective in designing a consumer mashups," *ACM New York NY*, vol. 3, pp. 222–229, 2011.

[21] S. Aghaee, C. Pautasso, and A. De Angeli, "Natural end-user development of web mashups," in *Visual Languages and Human-Centric Computing (VL/HCC), 2013 IEEE Symposium on*, Sept 2013, pp. 111–118.

[22] L. Chen, N. Shadbolt, C. A. Goble, F. Tao, S. J. Cox, C. Puleston, and P. R. Smart, "Towards a knowledge-based approach to semantic service composition," in *International Semantic Web Conference*, 2003, pp. 319–334.

[23] S. Roy Chowdhury, F. Daniel, and F. Casati, "Efficient, interactive recommendation of mashup composition knowledge," in *Proceedings of the 9th international conference on Service-Oriented Computing*, ser. ICSOC'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 374–388.

[24] V. Tietz, G. Blichmann, S. Pietschmann, and K. Meißner, "Task-based recommendation of mashup components." in *ICWE Workshops*, ser. Lecture Notes in Computer Science, A. Harth and N. Koch, Eds., vol. 7059. Springer, 2011, pp. 25–36.

[25] T. Fischer, F. Bakalov, and A. Nauerz, "An overview of current approaches to mashup generation," in *Fifth Conference Professional Knowledge Management: Experiences and Visions, March 25-27, 2009 in Solothurn, Switzerland*, ser. LNI, K. Hinkelmann and H. Wache, Eds., vol. 145. GI, 2009, pp. 254–259.

[26] A. Koschmider, V. Torres, and V. Pelechano, "Elucidating the mashup hype: Definition, challenges, methodical guide and tools for mashups," in *2nd Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web*, 2009.

[27] D. D. Hoang, H.-y. Paik, and B. Benatallah, "An analysis of spreadsheet-based services mashup," in *Proceedings of the Twenty-First Australasian Conference on Database Technologies - Volume 104*, ser. ADC '10. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2010, pp. 141–150.

[28] S. Aghaee and C. Pautasso, "An evaluation of mashup tools based on support for heterogeneous mashup components," in *Proceedings of the 11th international conference on Current Trends in Web Engineering*, ser. ICWE'11, vol. 7059. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 1–12.

[29] M. Mostarda and D. Palmisano, "Mu: an hybrid language for web mashups," in *International World Wide Web Conferences Steering Committee (IW3C2)*, ser. IW3C2, Madrid, Spain, 2009.

[30] S. Aghaee, M. Nowak, and C. Pautasso, "Reusable decision space for mashup tool design," in *Proceedings of the 4th ACM SIGCHI symposium on Engineering interactive computing systems*, ser. EICS '12. New York, NY, USA: ACM, 2012, pp. 211–220.

[31] E. Maximilien, H. Wilkinson, N. Desai, and S. Tai, "A domain-specific language for web apis and services mashups," in *Service-Oriented Computing ICSOC 2007*, ser. Lecture Notes in Computer Science, B. Kramer, K.-J. Lin, and P. Narasimhan, Eds. Springer Berlin Heidelberg, 2007, vol. 4749, pp. 13–26. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-74974-5_2

[32] D. Florian, F. Casati, B. Benatallah, and M.-C. Shan, "Hosted universal composition: Models, languages and infrastructure in mashart," in *Proceedings of the 28th International Conference on Conceptual Modeling*, ser. ER '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 428–443.

[33] M. Feldmann, T. Nestler, K. Muthmann, U. Jugel, G. Hübsch, and A. Schill, "Overview of an end-user enabled model-driven development approach for interactive applications based on annotated services," in *Proceedings of the 4th Workshop on Emerging Web Services Technology*, ser. WEWST '09. New York, NY, USA: ACM, 2009, pp. 19–28.

[34] T. Nestler, M. Feldmann, G. Hubsch, A. Preussner, and U. Jugel, "The servface builder - a wysiwyg approach for building service-based applications," in *Proceedings of the 10th International Conference on Web Engineering*, ser. ICWE'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 498–501.

[35] R. Tuchinda, P. Szekely, and C. A. Knoblock, "Building mashups by example," in *Proceedings of the 13th international conference on Intelligent user interfaces*, ser. IUI '08. New York, NY, USA: ACM, 2008, pp. 139–148.

[36] P. Baglietto, F. Cosso, M. Fornasa, S. Mangiante, M. Maresca, A. Parodi, and M. Stecca, "Always-on distributed spreadsheet mashups," in *Proceedings of the 3rd and 4th International Workshop on Web APIs and Services Mashups*, ser. Mashups '09/'10. New York, NY, USA: ACM, 2010, pp. 8:1–8:8.

[37] S. Aghaee and C. Pautasso, "Live mashup tools: Challenges and opportunities," San Francisco, CA, USA, May 2013.

[38] S. R. Chowdhury, C. Rodríguez, F. Daniel, and F. Casati, "Wisdom-aware computing: on the interactive recommendation of composition knowledge," in *Proceedings of the 2010 international conference on Service-oriented computing*, ser. ICSOC'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 144–155.

[39] C. Viedma, "Mobile web mashups: The long tail of mobile applications," Master's thesis, Communication Systems, School of ICT, 2010.

[40] H. Elmeleegy, A. Ivan, R. Akkiraju, and R. Goodwin, "Mashup advisor: A recommendation tool for mashup development," in *Proceedings of the 2008 IEEE International Conference on Web Services*, ser. ICWS '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 337–344.

[41] S. R. Chowdhury, C. Rodríguez, F. Daniel, and F. Casati, "Baya: assisted mashup development as a service," in *WWW (Companion Volume)*, 2012, pp. 409–412.

[42] S. Pietschmann, V. Tietz, J. Reimann, C. Liebing, M. Pohle, and K. Meissner, "A metamodel for context-aware component-based mashup applications," *ACM New York*, vol. 3, pp. 413–420, 2010.

[43] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, 2000, aAI9980887.

[44] R. Ennals, E. Brewer, M. Garofalakis, M. Shadle, and P. Gandhi, "Intel mash maker: Join the web," *SIGMOD Rec.*, vol. 36, no. 4, pp. 27–33, Dec. 2007.

[45] *Yahoo Pipes*. [Online]. Available: http://pipes.yahoo.com/pipes/

[46] F. Daniel and M. Matera, "Turning web applications into mashup components: Issues, models, and solutions." in *ICWE*, ser. Lecture Notes in Computer Science, M. Gaedke, M. Grossniklaus, and O. Díaz, Eds., vol. 5648. Springer, 2009, pp. 45–60.

[47] M. Imran, S. Soi, F. Kling, F. Daniel, F. Casati, and M. Marchese, "On the systematic development of domain-specific mashup tools for end users." in *ICWE*, ser. Lecture Notes in Computer Science, M. Brambilla, T. Tokuda, and R. Tolksdorf, Eds., vol. 7387. Springer, 2012, pp. 291–298.

[48] F. Daniel, M. Imran, F. Kling, S. Soi, F. Casati, and M. Marchese, "Developing domain-specific mashup tools for end users," in *Proceedings of the 21st International Conference Companion on World Wide Web*, ser. WWW '12 Companion. New York, NY, USA: ACM, 2012, pp. 491–492.

[49] X. Liu, Q. Zhao, G. Huang, H. Mei, and T. Teng, "Composing data-driven service mashups with tag-based semantic annotations," in *Proceedings of the 2011 IEEE International Conference on Web Services*, ser. ICWS '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 243–250.

[50] S. Wilson, F. Daniel, U. Jugel, and S. Soi, "Orchestrated user interface mashups using w3c widgets," in *Proceedings of the 11th International Conference on Current Trends in Web Engineering*, ser. ICWE'11. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 49–61.

[51] O. Chudnovskyy, T. Nestler, M. Gaedke, F. Daniel, J. I. Fernández-Villamor, V. Chepegin, J. A. Fornas, S. Wilson, C. Kögler, and H. Chang, "End-user-oriented telco mashups: the omelette approach," in *Proceedings of the 21st international conference companion on World Wide Web*, ser. WWW '12 Companion. New York, NY, USA: ACM, 2012, pp. 235–238.

[52] J. C. Corrales, "Behavioral matchmaking for service retrieval," Ph.D. dissertation, University of Versailles Saint-Quentin-en-Yvelines, January 2008.

[53] L. J. S. Meza, "Estimación del intento de búsqueda del usuario para la recuperación de recursos en la web," Master's thesis, Universidad del Cauca, Mayo 2014.

[54] Z. Ghahramani, "Unsupervised learning," in *Advanced Lectures on Machine Learning*, ser. Lecture Notes in Computer Science, O. Bousquet, U. von Luxburg, and G. RÃ€tsch, Eds. Springer Berlin Heidelberg, 2004, vol. 3176, pp. 72–112.

[55] P. D. McNicholas, T. B. Murphy, and M. O'Regan, "Standardising the lift of an association rule," *Comput. Stat. Data Anal.*, vol. 52, no. 10, pp. 4712–4721, Jun. 2008.

[56] J. Li, X. Zhang, G. Dong, K. Ramamohanarao, and Q. Sun, "Efficient mining of high confidence association rules without support thresholds," in *Principles of Data Mining and Knowledge Discovery*, ser. Lecture Notes in Computer Science, J. Å»ytkow and J. Rauch, Eds. Springer Berlin Heidelberg, 1999, vol. 1704, pp. 406–411.

[57] Q. Zhao and S. S. Bhowmick, "Association rule mining: A survey."

[58] K. S. Rubin, *Essential Scrum: A Practical Guide to the Most Popular Agile Process*, 1st ed. Addison-Wesley Professional, 2012.

[59] Beck, *Test Driven Development: By Example*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002.

[60] P. Clements, D. Garlan, F. Bachmann, J. Ivers, J. Stafford, L. Bass, and P. Merson, *Documenting Software Architectures: Views and Beyond*, 2nd ed. Addison-Wesley Professional, 2010.

[61] L. J. S. Meza, "Estimación del intento de búsqueda del usuario para la recuperación de recursos en la web," Master's thesis, Universidad del Cauca, Mayo 2014.

[62] B. Kitchenham, S. Linkman, and D. Law, "Desmet: a methodology for evaluating software engineering methods and tools," *Computing Control Engineering Journal*, vol. 8, no. 3, pp. 120–126, June 1997.

[63] R. Inostroza, C. Rusu, S. Roncagliolo, C. Jimenez, and V. Rusu, "Usability heuristics for touchscreen-based mobile devices," in *Information Technology: New Generations (ITNG), 2012 Ninth International Conference on*, April 2012, pp. 662–667.

[64] M. S. Omaña Montoya, "Modelo de calidad basado en caracteristicas para la seleccion de un sistema de gestion de aprendizaje," Master's thesis, Universidad Católica Andres Bello, Mayo 2009.

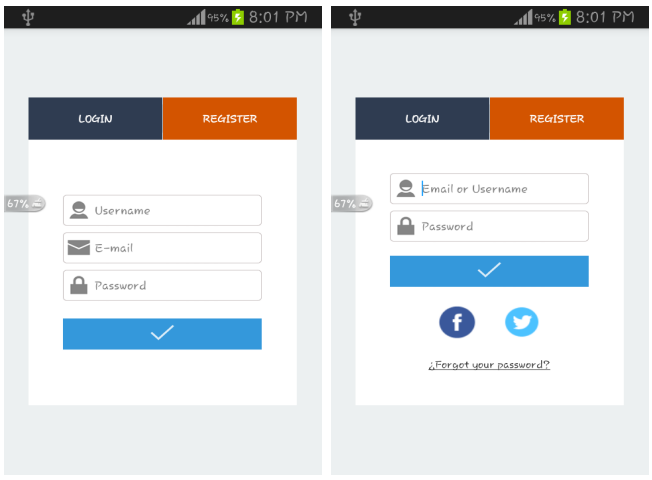
Anexo A

Diseño Herramienta - Generación Automática De Mashups En Dispositivos Móviles

A continuación se describe el diagrama de navegabilidad realizado para la implementación de la interfaz gráfica de usuario de la herramienta, el cual, fue desarrollado mediante una aplicación móvil bajo la plataforma Android.

A.1 Pantalla 1 - Sesión del Sistema

Interfaz gráfica que solo se muestra cuando se abre por primera vez o se cierra sesión en la aplicación móvil, en la cual se realiza el registro (Figura A.1(a)) e inicio de sesión (Figura A.1(b)) de usuarios mediante el sistema propuesto o validación de usuario a través de las redes sociales, como Facebook o Twitter.



(a) (b)

Figura A.1: Interfaz grafica – Sesión del Sistema.

A.2 Pantalla 2 - Consulta de Búsqueda

Interfaz gráfica en cual se ingresa la consulta de búsqueda en lenguaje natural por el usuario (Figura A.2).

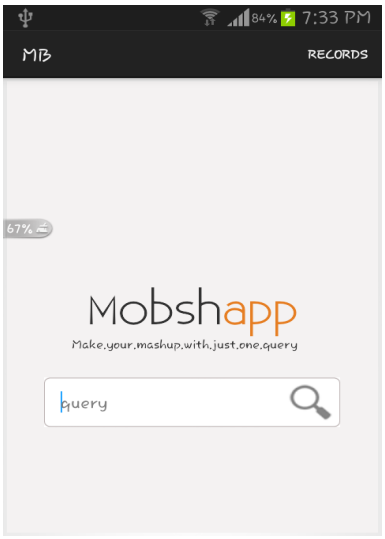
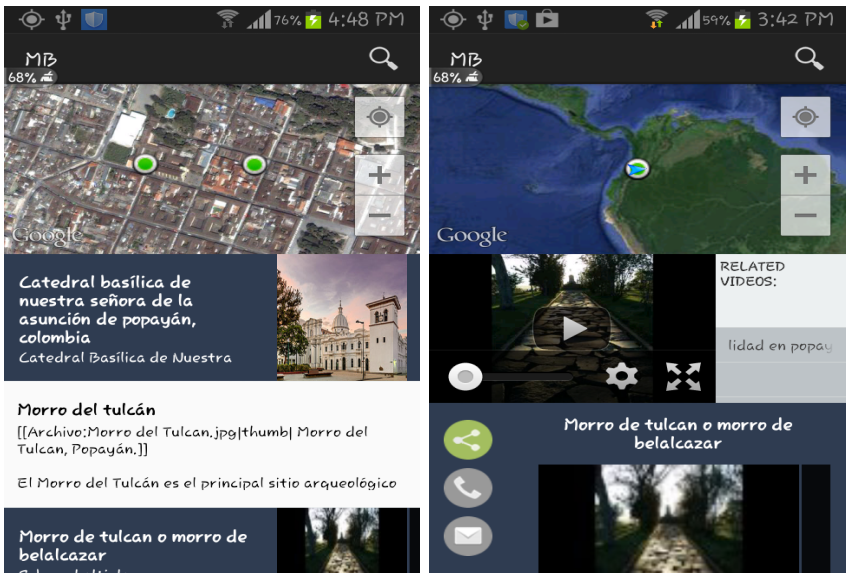


Figura A.2: Interfaz grafica – Consulta de Búsqueda.

A.3 Pantalla 3 - Recomendación de Contenidos

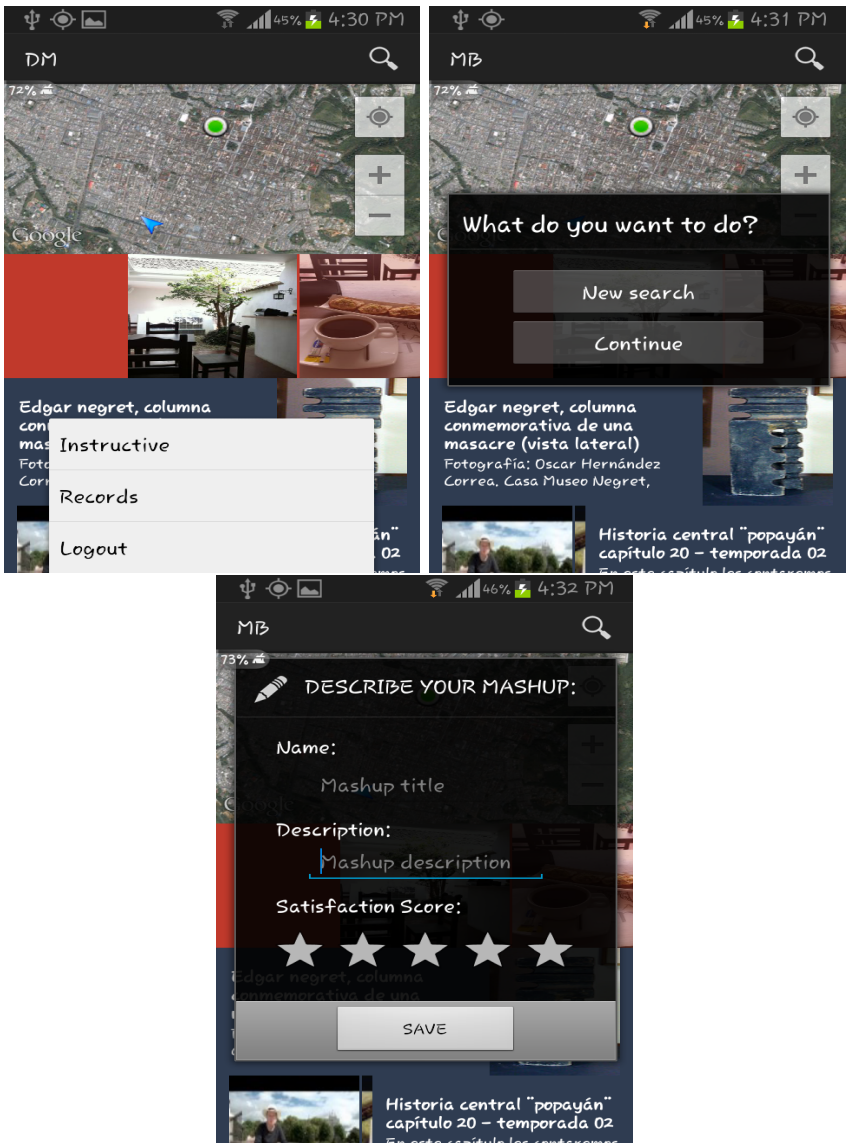
Interfaz gráfica en la cual se visualizan los resultados del mashup generado a partir de una consulta de búsqueda ingresada en la pantalla 2. Esta pantalla contiene 3 componentes principales como se observa en la Figura A.3, descritos a continuación: En el primero se visualiza las ubicaciones geográficas de cada componente. El segundo contiene un grupo de videos relacionados a cada componente. El tercero abarca un conjunto de componentes que representan el resultado de la consulta de búsqueda a través de contenidos de imagen, informacional o compuesto.



(a) (b)

Figura A.3: Interfaz grafica – Recomendación de Contenido.

Además, la pantalla consta de un menú de opciones (Figura A.4(a)), que describen las siguientes funcionalidades: La primera opción se encuentra visual, en el cual se le propone al usuario continuar (refinar los resultados a través de una consulta de búsqueda nueva – Figura A.4(b)) o terminar (los resultados satisfacen la consulta inicial por lo cual se visualiza una evaluación de satisfacción – Figura A.4(c)) el mashup en proceso. La segunda opción se encuentra oculta, en cual se puede acceder al tutorial de ayuda la herramienta.

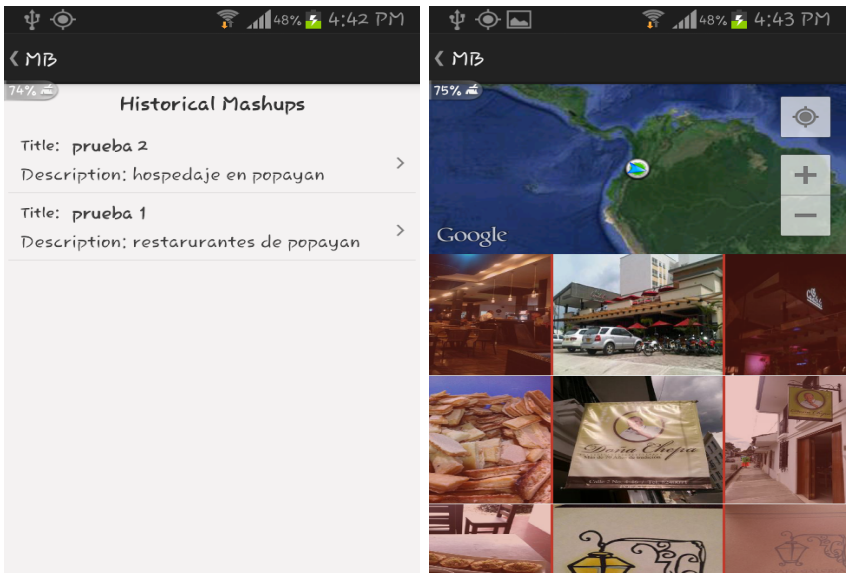


(a) (b) (c)

Figura A.4: Interfaz grafica – Recomendación de Contenido.

A.4 Pantalla 4 - Historial de Mashups

Interfaz gráfica en cual se visualiza un historial de los mashups creados que satisficieron los requerimientos de búsqueda de un usuario. Esta pantalla se representa a través de dos interfaz: En la primera se visualiza una lista donde se describe cada mashup generado (Figura A.5(a)). En la segunda se visualiza el mashups seleccionado de la lista mencionada, con el fin de mantener disponible la información al usuario (Figura A.5(b)).



(a) (b)

Figura A.5: Interfaz grafica – Historial de Mashups.

Anexo B

Evaluación Del Producto – Generación Automática De Mashups En Dispositivos Móviles

Nombre Evaluador:

Fecha Evaluación:

El cuestionario evalúa el grado de aceptación del objeto de evaluación desarrollado por parte de usuarios finales a partir de características de funcionalidad, para el trabajo de grado titulado "GENERACION AUTOMATICA DE MASHUPS EN DISPOSITIVOS MOVILES". Señale su respuesta como se indica en el ejemplo a continuación:

1. TITULO-PREGUNTA	
DESCRIPCIÓN: xxxxx	
PREGUNTA: xxxxx	
GRADO DE IMPORTANCIA: xxxx	
ESCALA DE EVALUACIÓN:	
Completamente	X
En alto Grado	
Parcialmente	
Muy Poco	
Ninguno	
OBSERVACIÓN: xxxx	

A continuación se presentan las encuestas:

1. Uso intuitivo de la herramienta	
DESCRIPCIÓN: Esta característica mide el grado de facilidad e intuición de la herramienta para permitir que usuarios sin conocimientos técnicos generen sus propios mashups a través de ella, sin la necesidad de invertir mucho tiempo indagando cómo funciona la herramienta o que alguna persona instruya en su uso.	
PREGUNTA: En qué grado considera que Mobshapp facilita el proceso de generación de mashups?	
GRADO DE IMPORTANCIA: Altamente deseable.	
ESCALA DE EVALUACIÓN:	
Completamente	
En alto Grado	
Parcialmente	
Muy Poco	
Ninguno	
OBSERVACIÓN:	

2. Ayuda Contextual	
DESCRIPCIÓN: Esta característica mide el grado de contextualización proporcionado por la herramienta, para aclarar sus funcionalidades además de brindar una guía paso a paso del proceso de desarrollo de mashups.	
PREGUNTA: En qué grado considera que la ayuda proporcionada por Mobshapp, contextualiza al usuario para realizar el proceso de desarrollo de mashups?	
GRADO DE IMPORTANCIA: Deseable.	
ESCALA DE EVALUACIÓN:	
Completamente	
En alto Grado	
Parcialmente	
Muy Poco	
Ninguno	
OBSERVACIÓN:	

3. Tipos de Acciones	
DESCRIPCIÓN: Esta característica mide el grado en el cual los tipos de acciones propuestas, tales como: llamar al establecimiento de interés del contenido sugerido; observar, enviar, agregar y eliminar contenido sugerido por la herramienta, son útiles para realizar tareas de la vida cotidiana.	
PREGUNTA: En qué grado considera que los tipos de acciones propuestas en Mobshapp son útiles para llevar a cabo su tarea de búsqueda?	
GRADO DE IMPORTANCIA: Altamente deseable.	
ESCALA DE EVALUACIÓN:	
Completamente	
En alto Grado	
Parcialmente	
Muy Poco	
Ninguno	
OBSERVACIÓN:	

4. Recomendación de Componentes	
DESCRIPCIÓN: Esta característica mide el grado en el cual la funcionalidad de recomendación cumple los requerimientos de la consulta en lenguaje natural y las interacciones del usuario durante el proceso de generación de mashups.	
PREGUNTA: En qué grado considera que las recomendaciones por Mobshapp fueron útiles durante el proceso de generación de mashups?	
GRADO DE IMPORTANCIA: Altamente deseable.	
ESCALA DE EVALUACIÓN:	
Completamente	
En alto Grado	
Parcialmente	
Muy Poco	
Ninguno	
OBSERVACIÓN:	

5. Tarea de Búsqueda	
DESCRIPCIÓN: Esta característica mide el grado en el cual la tarea de búsqueda es llevado a cabo en su totalidad a través de la herramienta, proporcionando al usuario los componentes finales que constituyen el mashup y representan el resultado de los requerimientos e interacciones del usuario.	
PREGUNTA: En qué grado considera que los objetivos de la tarea de búsqueda son logrados satisfactoriamente por medio de los componentes que constituyen el mashups final?	
GRADO DE IMPORTANCIA: Altamente deseable.	
ESCALA DE EVALUACIÓN:	
Completamente	
En alto Grado	
Parcialmente	
Muy Poco	
Ninguno	
OBSERVACIÓN:	

6. Interfaz de Usuario	
DESCRIPCIÓN: Esta característica mide el grado en el cual la herramienta cuenta con una interfaz grafica adecuada, con el fin que usuarios finales puedan realizar sus tareas a través de una interfaz grafica simple y practica, lo cual implica que esta no los distraiga de su objetivo de búsqueda.	
PREGUNTA: En qué grado considera que la interfaz grafica de Mobshapp, proporciona una adecuada visualización y disposición de los elementos necesarios para el desarrollo de mashups?	
GRADO DE IMPORTANCIA: Altamente deseable.	
ESCALA DE EVALUACIÓN:	
Completamente	
En alto Grado	
Parcialmente	
Muy Poco	
Ninguno	
OBSERVACIÓN:	

7. Tiempos de Respuesta	
DESCRIPCIÓN: Esta característica mide el grado de la percepción en relación a la rapidez en la entrega de los resultados, respecto a las funcionalidades de recuperación y despliegue de los componentes durante el proceso de generación de mashups.	
PREGUNTA: Que tan rápido considera se comporta las funcionalidades de recuperación y despliegue de los componentes durante el proceso de generación de mashups por Mobshapp?	
GRADO DE IMPORTANCIA: Deseable.	
ESCALA DE EVALUACIÓN:	
Completamente	
En alto Grado	
Parcialmente	
Muy Poco	
Ninguno	
OBSERVACIÓN:	

8. Reutilización de Componentes	
DESCRIPCIÓN: Esta característica mide el grado de reutilización de los componentes en el cual pueden ser recomendados por la herramienta, con el objetivo que otros usuarios puedan satisfacer los requerimientos durante el proceso de generación de mashups.	
PREGUNTA: Que tan adecuado considera que los componentes puedan ser reutilizados en otras composiciones?	
GRADO DE IMPORTANCIA: Deseable.	
ESCALA DE EVALUACIÓN:	
Completamente	
En alto Grado	
Parcialmente	
Muy Poco	
Ninguno	
OBSERVACIÓN:	