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

WORKFLOW AD-HOC CON GEOLOCALIZACIÓN Y MICRO BLOGGING

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERA CIVIL EN COMPUTACIÓN

MARCELA LORETO ROMERO CISTERNA

PROFESOR GUÍA:
NELSON BALOIAN TATARYAN

MIEMBROS DE LA COMISIÓN:
CLAUDIO GUTIÉRREZ GALLARDO
MARÍA CECILIA BASTARRICA PIÑEYRO

SANTIAGO DE CHILE
2017

RESUMEN DE LA MEMORIA PARA OPTAR AL TITULO DE: Ingeniero civil en Computación

POR: Marcela Loreto Romero Cisterna

FECHA: 03/08/2017

PROFESOR GUIA: Nelson Baloian Tataryan

WORKFLOW AD-HOC CON GEOLOCALIZACIÓN Y MICRO BLOGGING

Hoy en día muchas empresas modelan y definen sus procesos de negocio para utilizar motores Workflow. Hay ocasiones en donde es necesaria variar la forma en la que se ha definido inicialmente un proceso, por lo que basta con modificar la definición inicial de éste para seguir operando, pero, ¿Qué pasa cuando la definición de un proceso cambia contantemente? Las organizaciones que están orientadas al éxito dentro de los entornos empresariales deben ser capaces de cambiar rápida y continuamente. Es aquí donde se introduce el concepto de Workflow ad-hoc.

El presente trabajo muestra la implementación de una aplicación multiplataforma que pretende dar solución a la necesidad inminente de controlar un proceso, del cual no es posible saber su definición y que, además, el saber la ubicación del lugar físico donde se debe ejecutar cada tarea definida, aporta información y gran valor al proceso mismo.

En la solución ofrecida se integra la utilización de Patrones Workflow, Google maps como GIS, BPM y servicio microblogging. Con ayuda de estas herramientas el usuario podrá comunicarse por medio de mensajes cortos a través de un Servicio microblogging utilizando una sintaxis que emula los Patrones de Workflow, pudiendo o no visualizar la ubicación de los flujos en un mapa de Google maps, controlando todo lo anterior con la ayuda de las herramientas que nos provee un motor BPM.

AGRADECIMIENTOS

Ha sido un largo camino por el cual me han acompañado muchas personas, a las que me gustaría mencionar en este trabajo de título.

Quiero agradecer el apoyo incondicional de mi familia, especialmente el de mi madre. Gracias mamita linda por todo el apoyo que me has dado ☺

También quiero agradecer a mi futuro marido Nicolás, que ha estado siempre conmigo, en las buenas y en las malas. ¡Te amo!

En cuanto a mi viaje por la U, me gustaría agradecer en primera instancia a Jeremías Garay. Quizá nunca lea esto, pero quiero agradecer de todos modos por su amistad y por la ayuda que me brindo en los ramos Físicos.

También quiero agradecer a mi amigo Claudio Rojas. Sin su compañía la U no habría sido lo mismo. Gracias Clau ☺

Agradezco el compañerismo de nuestro grupo de memoristas Juan Pablo Arancibia y Alfonso Cornejo. ¡Tenía que nombrarlos!

Y finalmente, agradecer el gran apoyo que me ha dado mi jefe Guillermo Quezada. ¡El mejor jefe!

Agrego agradecimientos a nuestro departamento de Ciencias de la Computación. La calidad humana que tiene su personal y profesores es invaluable.

¡Muchas gracias por permitirme ser una profesional y acompañarme en mi camino!

TABLA DE CONTENIDO

INDICE DE FIGURAS E IMÁGENES

1. INTRODUCCIÓN

Toda Empresa crea planes y proyectos para abordar nuevos mercados, clientes, o productos. Estos planes de negocios generalmente tienen un alto grado de dependencia de las TI.

Fusionar en un solo mapa, la estrategia de negocio, la implementación, la información y los aspectos de infraestructura es un error común. La correcta arquitectura implica la separación de las preocupaciones, en esta separación la estrategia de negocio se resuelve en forma concreta diseñando los Procesos de Negocio. Para abordar este aspecto con una estrategia, metodologías y buenas prácticas nace un concepto llamado **BPM** (**BUSINESS PROCESS MANAGEMENT**) que significa Gestión de Procesos de Negocios, o también Gestión de Procesos Empresariales, donde debemos tener en cuenta que el objetivo de un proceso de negocio estará altamente relacionado con la estrategia de negocio y su dominio.

Muchas organizaciones diseñan, automatizan e implementan estos procesos de negocio basándose en motores Workflow.

Un **MODELO WORKFLOW** (WFM) es predefinido para cada proceso de negocio, conformado por un número fijo de actividades y un conjunto de condiciones las cuales unen estas actividades (1). Una aplicación de **WORKFLOW** automatiza la secuencia de acciones, actividades o tareas utilizadas para la ejecución del proceso, incluyendo el seguimiento del estado de cada una de sus etapas y el aporte de las herramientas necesarias para gestionarlo.

De manera similar, se ha descubierto la necesidad y utilidad de georreferenciar la información, lo cual muchas empresas han explotado como nicho de negocio. Sin ir muy lejos, Esri Chile[1] es una de estas empresas que ofrece servicios y soluciones integrales en el ámbito de las geo-tecnologías.

[1] http://www.esri.cl/

En su página .com es posible ver su producto **GeoPlanner for ArGis**, producto que, además de mezclar Workflow con GIS permite diseñar escenarios, entender el impacto de cada diseño, mejorar el análisis espacial y comparar escenarios alternativos.

Claramente, no se puede suponer que la ubicación de las actividades puedan ser las mismas ni menos que la definición del proceso será estática durante toda la vida del negocio. Satisfaciendo ésta última premisa se creó el término de **Workflow ad-hoc**.

Ya en el año 2000 se sabían las ventajas de **Workflow ad-hoc**. La principal ventaja es que los procesos de negocio pueden ser creados flexible y dinámicamente, además de la habilidad que tienen para enfrentar los cambios en las reglas de negocio y definiciones de servicio, implementaciones o, incluso, localizaciones (1).

Si a esto agregamos lo beneficioso que sería, para muchos procesos, agregar información geográfica para las actividades pertenecientes a éstos, tenemos una poderosa herramienta.

Un sistema **BMP georreferenciado** integrando las funcionalidades de **GIS** (Geographical Information Systems) y **BPM** (Business Process Management) podría ser de gran ayuda en procesos de negocio en los cuales optimizar las localizaciones de cada actividad es relevante.

Algo similar se ha visto, con antelación, en la intersección entre **BPM** y **micro blogging**. Esta joven combinación ha sido bautizada como **Social BPM**.

Social BPM es una disciplina que combina técnicas tradicionales de gestión de procesos de negocio con herramientas sociales y las tecnologías de la Web 2.0, para facilitar los esfuerzos de mejora de negocio. Un ejemplo de esto es **Tweetflows**, una plataforma liviana que soporta la coordinación de procesos de negocio utilizando **Twitter** (2) (3). Es tal el impacto que ha presentado micro blogging a través de su representante **Twitter** que, según los datos

publicados por Flurry, durante el año 2013, **TWITTER** fue unos de los principales propulsores para el incremento en el uso de dispositivos móviles el cual fue de un 115%.

Esto último es un punto a favor para que el desarrollo de la solución propuesta sea multi-dispositivos y que no se centre solo en un tipo de éstos, además de avalar la geolocalización por la utilización de dispositivos móviles y la necesidad de saber la ubicación desde donde se están realizando las acciones.

1.1. Definiciones

En esta sección se detallan los términos que se utilizaran a lo largo del presente documento en conjunto con su respectiva definición:

- API: Acronimo de **Application Programming Interface**. Es el conjunto de funciones o métodos que ofrece cierta aplicación para ser utilizado por otro software como una capa de abstracción.
- AngularJS: Framework javascript de código abierto, cuyo autor es Google. Utiliza el patrón MVC (**Modelo Vista Controlador**) ayudando en la gestión de aplicaciones de una sola página.
- BPM: Siglas en ingles de **Business Process Management**. Gestión de procesos de negocio es una metodología y disciplina de gestión de procesos. Su objetivo es mejorar el desempeño y la optimización de los procesos de negocio de una organización.

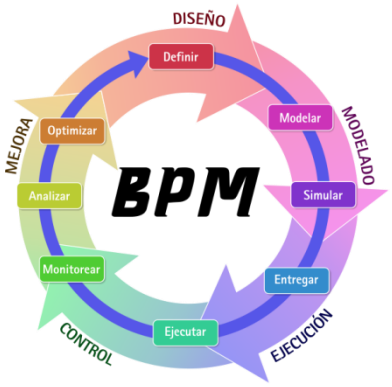


Figura 1: Ciclo de vida de BPM

- CouplingServer: Servidor de objetos acoplados. Proveedor la comunicación entre los distintos actores participantes de los procesos.
- DOM: En inglés **Document Object Model**. Se define como una **API** que proporciona un conjunto estándar de objetos para representar documentos HTML y XML, un modelo de cómo pueden combinarse dichos elementos y una interfaz para acceder y manipular a éstos mismos.

- Flow: Acción a ser ejecutada en un proceso. Análogo a las tareas de un proceso.
- GIS: Siglas en ingles de **Geographic Information Systems**. Sistema diseñado para gestionar información geográfica.
- Microblogging: Servicio que permite a sus usuarios enviar y publicar mensajes breves. El principal propulsor de este concepto es **Twitter**.
- Proceso: Conjunto de acciones y reglas a ser ejecutadas.
- Tweet: Mensaje corto escrito por los usuarios.
- Tarea: Unidad de trabajo en una actividad sin un proceso. En este trabajo se usara mayormente como sinónimo de **Flow**.
- Sequence Flow: Es una secuencia lineal usada para mostrar el orden de las actividades que componen un proceso.
- Spring MVC:
- Workflow: Orquestador y patrones repetibles de actividades de negocio habilitadas por la organización sistemática de recursos dentro de los procesos que transforma materiales, provee servicios o procesa información.
- Workflow ad-hoc: Es un orquestador de procesos en el cual cuya serie de actividades no pueden ser predefinidas. En este tipo de procesos, los usuarios deben estar dispuestos a decidir qué hacer y cuando hacerlo. Deben además estar dispuestos a asignar actividades a otros usuarios, de manera de generar interacción entre varios usuarios.

1.2. Motivación

Muchos negocios requieren integrar procesos y manejar información geográfica. Por ejemplo, en Nueva Zelanda es común contratar compañías de basura privadas, por lo que cada semana, o en una determinada fecha, un camión va y recolecta la basura.

Por supuesto, estos contratos están constantemente siendo creados y cancelados. Esto significa que las rutas de recolección tienen que ser

redefinidas al menos una vez semanal. Además, ya que la competencia es feroz y los costos con el combustible, los recursos humanos y el mantenimiento debe mantenerse baja, las rutas de recolección tienen que ser optimizadas de forma permanente.

Los **SISTEMAS DE INFORMACIÓN GEOGRÁFICA** (**GIS**) pueden ayudar a definir y optimizar las rutas de recolección de una manera visual. Pero una ruta de recolección también puede ser vista como un proceso de negocio ad-hoc, que comprende un conjunto de tareas de recolección de basura. El término ad-hoc destaca la naturaleza volátil de los procesos de negocio, que requieren constante redefinición.

Otro ejemplo, en donde ya se está utilizando la combinación Workflow + GIS es en manejo de catástrofes. La **FEDERAL EMERGENCY MANAGEMENT AGENCY** (**FEMA**) es una asociación estadounidense que brinda apoyo a la ciudadanía y a las agencias de primera respuesta para trabajar en conjunto frente a situaciones catastróficas. **FEMA** tiene a su disposición E-Task, sistema Workflow que permite rastrear recursos naturales (agua, nieve, etc.) y envíos de ayuda para actuar contra una situación catastrófica. Este proyecto se compone de cuatro workflows principales con capacidades de reporte y de búsqueda. La pregunta es, ¿Qué pasaría si en este mismo escenario se utilizará un motor workflow ad-hoc?

Otro escenario posible es el rescate de mascotas. Se ha visto un aumento significativo en las causas animalistas. Esto se refleja en el contenido actual de redes sociales e incluso en noticieros por televisión abierta, en donde no es extraño ver noticias de rescate de mascotas. Los grupos animalistas deben hacer una serie de tareas para poder rescatar una mascota, entre ellas recogerlo, llevarlo al veterinario inmediatamente (esto a razón que no contagien a las demás mascotas rescatadas con enfermedades letales), llevarlo a un hogar temporal, etc, hasta la tarea final que podría ser llevarlo a un hogar definitivo en donde sea adoptado. Cada una de estas tareas tiene

una geolocalización diferente por lo que sería de gran ayuda registrar estas actividades en un proceso con el fin de saber en qué estado va cada caso de mascota. En este caso sería perfectamente también factible aplicar **GIS** y **SOCIAL BPM**.

Un sistema BPM georreferenciado integra la funcionalidad proporcionada por el **SIG** y **BPM** lo que proveería apoyo a las personas que gestionan los escenarios descritos anteriormente y muchos otros escenarios similares, tales como el mantenimiento de infraestructuras, rondas policiales y manejo forestal, por mencionar algunos.

Los procesos o **WORKFLOWS AD-HOC** son procesos especiales que pueden permitir identificar las desviaciones de una definición formal de un proceso, guardando registro de estas desviaciones. Ejemplos en donde es posible aplicar **WORKFLOW AD-HOC** incluye equipos consultores, equipos de marketing, equipos de diseño, pequeños equipos de ingenieros, equipos de mitigaciones de crisis (4).

1.3. Objetivos

El objetivo general de este trabajo de título es diseñar e implementar un sistema de workflow ad-hoc, que permita crear procesos, en donde las tareas (**FLOW** de ahora en adelante) podrán ser creadas por los usuarios de acuerdo a las necesidades que se vayan presentando con el fin de lograr el objetivo por el cual ha sido creado el proceso. Todo esto será implementado bajo la plataforma de **GOOGLE MAPS**, junto con **MICRO BLOGGING** y patrones de diseño de **WORKFLOW**, utilizando una sintaxis de expresiones regulares para identificar el tipo de patrón, las tareas y los usuarios a los cuales se les ha asignado determinado **FLOW.**

Los objetivos específicos que se desea lograr al finalizar el trabajo de título son:

- Entender **COPUPLINGSERVER**.

- Diseñar la estructura de datos con la que se manejarán los Usuarios, Procesos, Flows y georreferencias.
- Administrar grupos de usuarios, integrando control de acceso a usuarios a través de permisos, permitiendo a usuarios participar de múltiples procesos a la vez. Un usuario será parte de un proceso en los siguientes escenarios:
  - Es el usuario creador del Proceso
  - Un usuario ya perteneciente al proceso le ha asignado un **FLOW**.
- Diseñar la interfaz de usuario para definir cómo será la usabilidad que experimentarán los usuarios.
- Permitir la comunicación entre los usuarios, miembros de un Proceso, a través de **MICRO BLOGGING**. Los usuarios también podrán crear **FLOWS** de los procesos por medio de texto, diferenciando los **FLOWS** por medio de expresiones regulares para emular los **PATRONES DE WORKFLOW** presentados en la tabla de la Figura 2 (5). Además se creará un nuevo Patrón, el que se utilizará para indicar que se ha finalizado la realización de una tarea asignada.

Patrón	Descripción
Comenzar (Start)	Patrón de workflow creado para iniciar un proceso.
Secuencial (Sequence)	Se utiliza para indicar que la tarea sigue de manera secuencial a la tarea anterior, luego de que ésta fue completada.
Paralela (And Split)	Se utiliza cuando varias actividades pueden realizarse concurrentemente o en paralelo.
Sincronización a (And Join)	Punto de convergencia de múltiples tareas. En este caso se menciona que se ha terminado la tarea para que el coordinador del proceso haga el ***merge***.
Sincronización b (And Join)	Otra manera de converger tareas en paralelo. En este caso sólo se informa el término de la tarea al usuario que se encuentra realizando la otra tarea en paralelo con el fin de comunicar que el flujo se ha unificado nuevamente.
Cerrar (Close)	Se utiliza para continuar el flujo, como punto de convergencia, luego de tener dos o más caminos excluyentes entre sí.

Figura 2: Patrones de Workflow a utilizar

- Lograr un buen manejo de los objetos creados, de manera colaborativa, entre los usuarios participantes de un **Proceso**, y geo-localizada, a través de una interfaz que provee la **API** de **Google Maps**.
- Implementar filtro por usuario, es decir, que escogido un usuario sea posible saber cuáles han sido sus aportes a lo largo del proceso.
- Implementar filtro por **Flow**, es decir, si se selecciona un **Flow** de una lista, indicar visualmente, cuales son las geolocalizaciones asociadas a ese **Flow**.
- Utilizar algún framework para javascript como **BackBones** o **AngularJs**.
- Implementar un sistema de autocompletado tipo twitter y facebook para los nombres de proceso, nombres de usuario y nombre de flows.
- Añadir eventos a Geo-localizaciones para que al hacer *click* sobre ellas muestren información relacionada con el **Flow**.

1.4. Alternativas analizadas, alternativas escogidas

Desde sus inicios se consideró aplicar patrones de workflow para plantear la solución al problema de **procesos ad-hoc**. Uno de los principales problema recaía en cómo plantear una manera sencilla de utilizar los patrones de **workflow**, ya que si se utilizaba los veinte patrones existentes (5), en su cabalidad, los usuarios se abstendrían de utilizar la herramienta por incluir conceptos muy técnicos.

Estructuralmente, se analizó la posibilidad de utilizar un framework **Javascript**. Inicialmente se había pensado en **BackBones**, por ser uno de los primeros frameworks de este tipo, pero pronto se cambió de **AngularJs** ya que tiene muy buena documentación y la introducción a comenzar a escribir líneas de código es relativamente sencilla. A diferencia de **BackBones**, **AngularJs** ya provee la funcionalidad de objeto observable que en **BackBones** hay que configurar desde cero, además de que **AngularJs** es un proyecto de Google, por lo que se pensó tendría mejor acople con **Google Maps**, lo que resulto en un acierto.

Por otro lado, se analizó moderar o no moderar la generación del proceso con el fin de controlar la convergencia de flujos y, en especial, el cierre del proceso. En este caso se analizaron dos escenarios:

- Sólo se permite cerrar el proceso cuando todas las tareas hayan sido concretadas por alguno de los usuarios.
- Permitir cerrar un proceso en donde no se haya concluido una o muchas tareas.

Al ser un proceso de naturaleza ad-hoc es posible que no todas las tareas creadas sean obligatorias para la correcta ejecución del proceso, por lo que se optó por la segunda opción en donde se permite al usuario cerrar el proceso independiente de si hay tareas no concluidas, para, de cierta manera, simular la opción cancelar proceso que existente dentro de los patrones de workflow.

A su vez, los **GIS** presuponen dos componentes fundamentales: la transformación de datos y visualización.

Con respecto a la geolocalización, se pensó inicialmente en incluir las coordenadas dentro del mensaje o tweet, pero luego se desistió. Esto porque se utiliza la interfaz de **GOOGLE MAPS** para marcar las geolocalizaciones por lo que se decidió utilizarla también, en el caso que el usuario requiriera consultarla. Otra razón que motivo a tomar esta decisión fue que los mensajes quedarían demasiado extensos, de incluir una geolocalización que incluya muchos puntos del mapa en su definición.

1.5. Descripción General de la Solución

En este trabajo se desarrolló una herramienta de apoyo a la gestión de los procesos generados de manera ad-hoc, es decir, sin previa definición.

Para ello se definió que el orquestador de procesos se diseñaría utilizando los patrones de **WORKFLOW** y que, este último, se utilizaría de forma ad-hoc.

Con el fin de no ser tan técnicos en la utilización de **WORKFLOW**, ya que los usuarios que utilicen la aplicación no deben requerir necesariamente conocer dichos patrones, se simplificó la nomenclatura quedando como se muestra en la siguiente tabla.

Patrón	Expresión regular	Descripción
Start	#.*	Creación de un proceso
Sequence	%1 #.*[*|@.* ^.*]	Creación de un Tweet secuencial
Paralel	%2 #.*[*|@.* ^.*]	Creación de flujo paralelo a la rama inicial
Join a	%3a #.*[*|^.*]	Volver a unir rama paralela a rama principal
Join b	%3b #.*[*|@.*]	Volver a unir rama paralela a rama principal
Finish	%11 #.*	Cierre de proceso

Figura 3: Definición de patrones de workflow utilizados

Para comprender mejor la nomenclatura, en la siguiente tabla se detalla un ejemplo de cada patrón utilizado.

Patrón	Ejemplo
Start	#rescate gatito sobre el árbol de Ñuñoa
Sequence	%1 #rescate @usuario1 ^pedir_ayuda_a_bomberos
Paralel	%2 #rescate @usuario2 ^pedir_apoyo_redes_sociales_y_prensa
Join a	%3a #rescate ^pedir_apoyo_redes_sociales_y_prensa
Join b	%3b #rescate @usuario1
Finish	%11 #rescate

Figura 4: Ejemplo de utilización de Patrones Workflow

Luego de utilizando estos patrones por medio de mensajes cortos, uno de los aspectos relevante a considerar, es poder llevar a cabo un **BPM** en donde las

actividades se realizarán con distintas localizaciones, por lo que se requirió de un sistema de comunicación para la coordinación. Bajo este mismo anhelo, nació **MOBIZ**, aplicación construida como trabajo de memoria por el exalumno Diego Aguirre del **DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN** de la **UNIVERSIDAD DE CHILE**, que permite el trabajo concurrente de varios actores en el modelamiento de procesos sobre plataformas móviles. Construido con HTML5, posee la ventaja de ser soportado por múltiples plataformas, vale decir, ser accedida desde cualquier dispositivo moderno sin importar su sistema operativo (6).

Para la implementación de **MOBIZ** se creó el framework llamado **COUPLINGSERVER**, en donde se implementó todo lo referente a la comunicación entre los usuarios, logrando abstraer completamente la aplicación de la comunicación.

Utilizando la herramienta antes mencionada, se desarrolló una aplicación que nos permite una comunicación concurrente para la gestión de procesos personalizados utilizando los patrones de **WORKFLOW** de manera ad-hoc.

Dado que **COUPLINGSERVER** funciona a nivel de Cliente, la solución implementada se desarrolló en su mayor parte, también por el lado del cliente. Esto quiere decir que, si tengo un grupo de usuarios ejecutando un proceso ad-hoc, son ellos mismos quienes, de manera local, almacenan la información enviada y recibida en la memoria local del dispositivo por el cual se están comunicando.

Por todo lo anterior se propone, como trabajo de título, un sistema de **WORKFLOW AD-HOC** en donde se mezclan los beneficios y las bondades de **BPM**, **GIS** y **MICRO BLOGGING** sobre dispositivos móviles partiendo de la base que provee **COUPLINGSERVER**, ayudando a abstraer la comunicación.

1.6. Resultados de la implementación para resolver el problema

Si bien la implementación solo fue una prueba de concepto, hubiese sido más real y enriquecedor poder probar la aplicación en un escenario real y no en un escenario simulado. Se escogió un escenario conocido para, sobre eso, fundamentar el apoyo que podría haber brindado una aplicación con las características descritas a lo largo de este informe.

Como bien se verá en el capítulo 5. Validación de la Solución, los actores del escenario escogido escogieron utilizar una plataforma como Twitter para la comunicación, lo que avala en cierto modo el acierto del uso de Microblogging como medio de comunicación para resolver el problema de llevar a cabo un proceso. Además de que el escenario era un escenario inesperado e irrepetible, por lo que era de necesidad un proceso de contingencia que pudiera apoyar a enfrentar la situación.

2. MARCO TEORICO

2.1. Estado del Arte

Hasta ahora, la necesidad de poder soportar **MODELOS WORKFLOW** para procesos no repetibles, o en constante modificación, se ha mirado desde otros puntos de vista, como en **E-SCIENCE**, en donde no se sabe cuándo se debe repetir un sub-conjunto de actividades de un proceso, haciendo que el proceso en si sea cada vez diferente al anterior. Por ello, como el problema es una necesidad puntual, se mira desde el punto de vista de la recursión y de una nueva ejecución de una parte del proceso (7).

Lo más cercano a lo que se quiere lograr en este trabajo de título se ha descrito como **TWEETFLOWS** (2) (3). **TWEETFLOWS** es un lenguaje liviano para la creación de procesos de forma móvil. Su objetivo es proveer un lenguaje el cual siga los principios de SOA aplicados al dominio móvil. Pero queda en la similitud ya que en **WORKFLOW AD-HOC**, entre los objetivos, no se persigue proveer un lenguaje bajo los principios de SOA, sino más bien basarnos en el modelamiento de los procesos que varían en su definición, de forma de poder definir el flujo que se genera según la instancia.

La muy conocida empresa **BIZAGI** también propone una herramienta para solucionar la problemática que proponen los procesos sin definición previa. Con ayuda de un patrón de **PROCESO AD-HOC**, se usa un evento disponible para todos los usuarios en cualquier lapso de tiempo del proceso, para crear y asignar tareas para sí mismos u otros actores del proceso, similar a una fábrica de tareas. En la imagen de la Figura 5 es posible ver la definición del proceso donde se encuentra la tarea que permite crear nuevas tareas llamado **PERFOM AD HOC TASK**. Para persistir la información de la tarea, se crea un formulario que recibe los datos de la nueva tarea a crear.

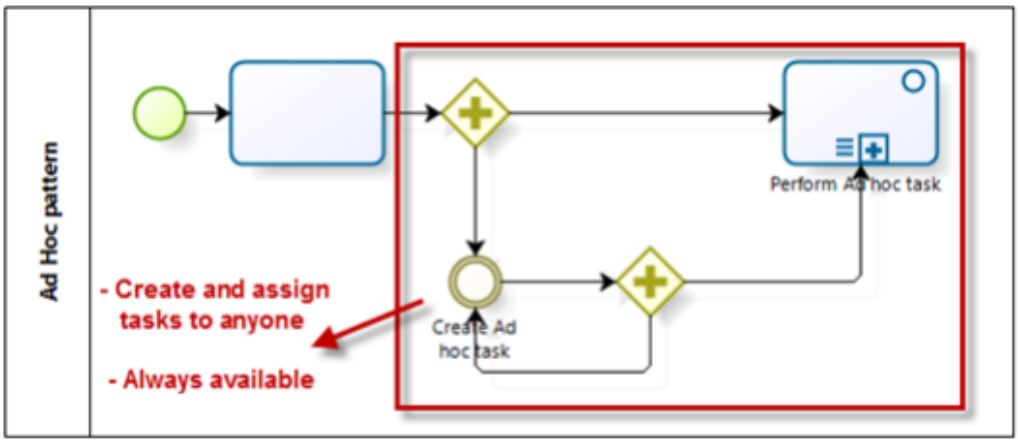


Figura 5: Proceso en donde se mantiene siempre disponible la tarea para crear taras de manera ad-hoc, valga la redundancia.

Si bien se resuelve la problemática para un **WORKFLOW AD-HOC**, no se acerca a lo que desea solucionar en el presente trabajo de memoria, ya que se busca simplificar el lenguaje por el cual se crean las tareas.

Otra herramienta existente en el mercado es **AD-HOC WORKFLOW**, herramienta desarrollada por el grupo **COMALA TECH** que ofrece modelar el proceso de documentar una wiki a través de un **WORKFLOW AD-HOC**. Cada documento creado tiene un conjunto de tareas que son asignadas a los usuarios. Cuando los usuarios cumplen con esas tareas lo informan por medio de notificaciones. La sintaxis usada es por medio de notación @ para referirse a los usuarios más texto. Ej: @martin revisar por favor.

Si bien **AD-HOC WORKFLOW** es la herramienta que su nombre describe, se ha ceñido sólo para solucionar un tipo de problemática y no abarca más que eso: gestionar documentos.

2.1.1. Trabajos relacionados

1. TWEETFLOWS: es un lenguaje especializado utilizado para **WORKFLOWS** sobre teléfonos móviles con **TWITTER** como canal de comunicación. Actualmente se encuentra realizando un estudio público con la ayuda de la **APP**, la cual filtra **TWEETS** por comandos **TWEETFLOWS** (2) (3).
2. Ad-hoc Workflow por Comala Tech: Permite administrar y controlar la documentación, aprobando, asignando y notificando tareas, ayudando

a crear, revisar y publicar contenido. Específicamente, entre sus funcionalidades cuenta con:

- Asignar contribuyentes y usuarios quienes se encargarán de revisar y dar de alta una tarea a tarea.
- Crear tareas para el equipo
- Publicar sólo contenido aprobado
- Administrar fases de documentación
- Generar informes de cumplimiento
- Habilitar flujos de trabajo para tareas o sub-procesos repetitivos

3. Procesos ad-hoc de Bizagi: Es una suit que incluye dos productos complementarios, **BIZAGI MODELADOR** de procesos y la **SUITE DE BPM**. Dentro de estos productos se provee una herramienta para crear procesos ad hoc a través de un patrón especial (ver Figura 5). Se debe diseñar un proceso en donde se mantenga siempre una tarea de usuario disponible, la cual sirve para crear las tareas ad-hoc. Para ello se debe ingresar os datos de la tarea a ser creada como el nombre de la tarea, el usuario al que se le asignará dicha tarea y la fecha de creación. Con el fin de persistir esta información, existe una estructura especial en donde el detalle de la tarea queda guardado permanentemente (Ver Figura 6).

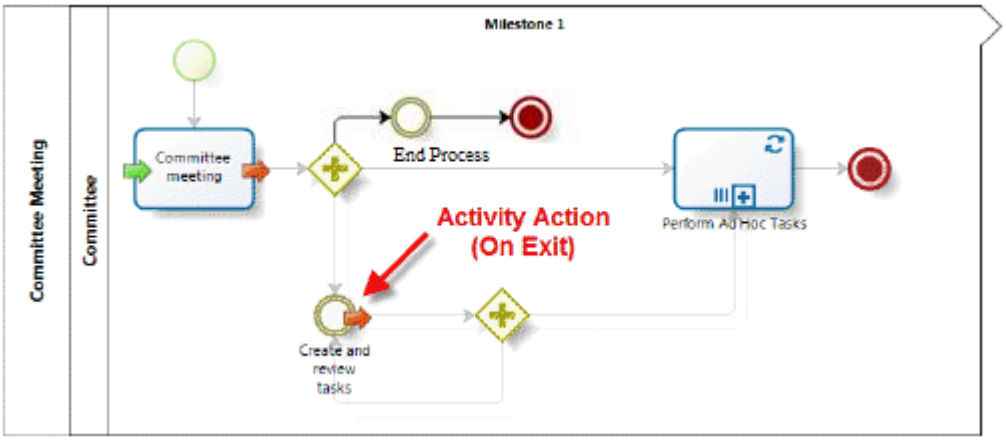


Figura 6: Estructura que persiste la información de las tareas creadas de manera ad-hoc en Bizagi.

4. Caramba: Se define como un Sistema de colaboración de procesos que soporta procesos ad hoc y colaborativos en equipos virtuales (8). Caramba podría ser una solución similar a la que se busca llegar en el presente trabajo de título, pero cabe destacar que no provee automatismo si se producen desviaciones a un proceso previamente modelado, ocurriendo inconsistencias. Aun así, el registro de las actividades (control y datos) es visible para todos los miembros del equipo que trabaja en el proceso. Las primitivas de coordinación están previstas para resolver los problemas subyacentes en las actividades o tareas.
5. TeamLog: Trabajo que introduce el concepto de Minería en el contexto de los procesos ad hoc. Utilizado en conjunto con Caramba, convierte el log transaccional que ofrece Caramba, en un formato general (9).
6. Hybrid Algorithmic-Crowdsourcing Workflows (10): Ha demostrado ser una poderosa técnica para superar desafíos en procesamiento de la información donde los algoritmos existentes aun no pueden resolver. Crowdsourcing ha comenzado a ser utilizado para muchos problemas que no son fácil de resolver por métodos o algoritmos automáticos, o que requieren una cantidad significante de esfuerzo o retroalimentación humana, es decir, donde los algoritmos heurísticos no son efectivos. Actualmente, una gran variedad de plataformas como **AMAZON MECHANICAL TURK**, **MULTITUD-FLOR.COM**, o **CLICKWORKER.COM** están ofreciendo servicios con diferentes grados de sofisticación, donde cualquier tipo de tareas cognitivas (por lo general relativamente simples) puede ser publicado y tratado de forma dinámica por mano de obra disponible.
7. Social BPM (11) (12): Nicolas Pflanzl y Gottfried Vossen han dedicado dos de sus trabajos a la temática de Social BPM. En ambos trabajo se profundiza sobre el significado de Social BPM y se le entrega una definición. En ambos trabajos se menciona que Social BPM es la

práctica de la participación activa de todas las partes interesadas pertinentes en BPM a través del uso de software social y sus principios subyacentes. Se dice que esto permite mejorar la exactitud, integridad y utilidad de los modelos de proceso e instancias mediante el aprovechamiento del dominio y el método de conocimiento de toda la comunidad empresarial. Sin embargo, la gran cantidad y variedad de colaboradores y contribuciones también se traduce en una serie de desafíos, los que son abordados en ambos trabajos.

2.2. Recursos utilizados

La distribución de los recursos es como se puede ver en la imagen de la Figura 7. Aquí es posible visualizar que los distintos dispositivos se conectan a las vistas de la aplicación, o llamada Interfaz de usuario, la cual pertenece a la misma aplicación identificada en la imagen como **APP WORKFLOW AD-HOC**. Además consume los servicios de **GOOGLE MAPS V3.0** y el Servidor de Objectos Acoplados (**COUPLINGSERVER**) para proveer la comunicación entre los usuarios clientes. Todo esto se desarrolló sobre un framework **SPRING MCV**.

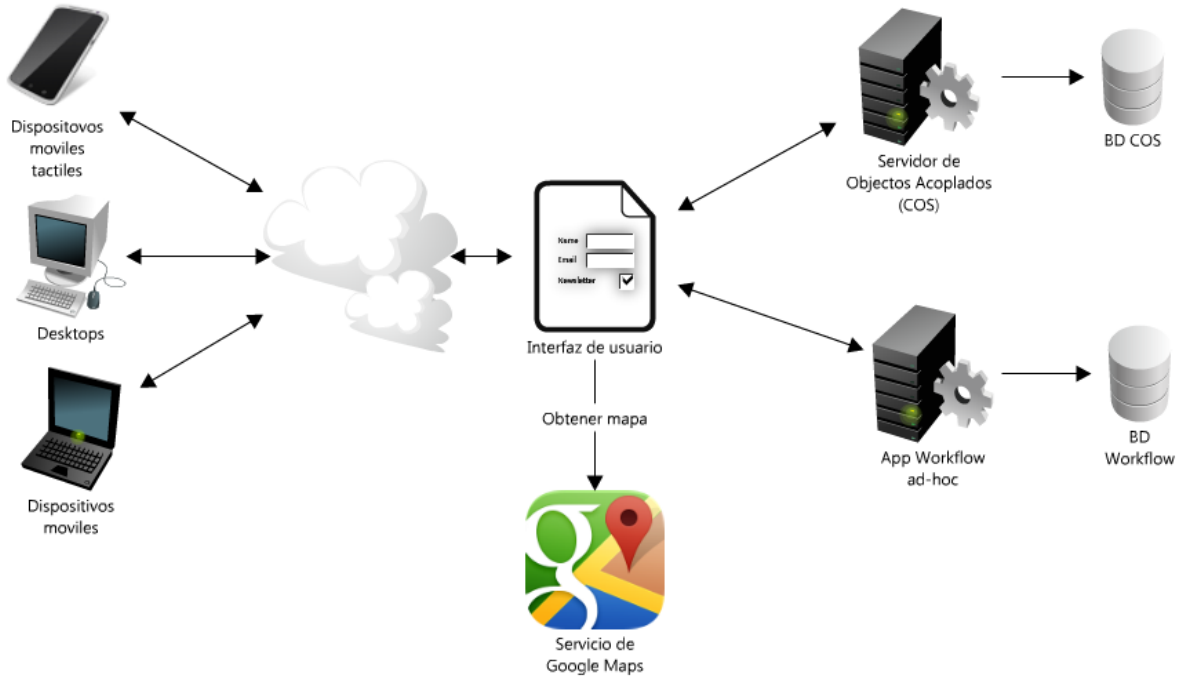


Figura 7: Estructura de los Servidores

Las tecnologías utilizadas se disgregan como muestra la Figura 8, en donde es posible ver que la mayor parte de tecnologías utilizadas se concentra en el Cliente. Esto ya que es donde se concentra la lógica de la aplicación, por tratarse de una sola página que va actualizando su contenido.

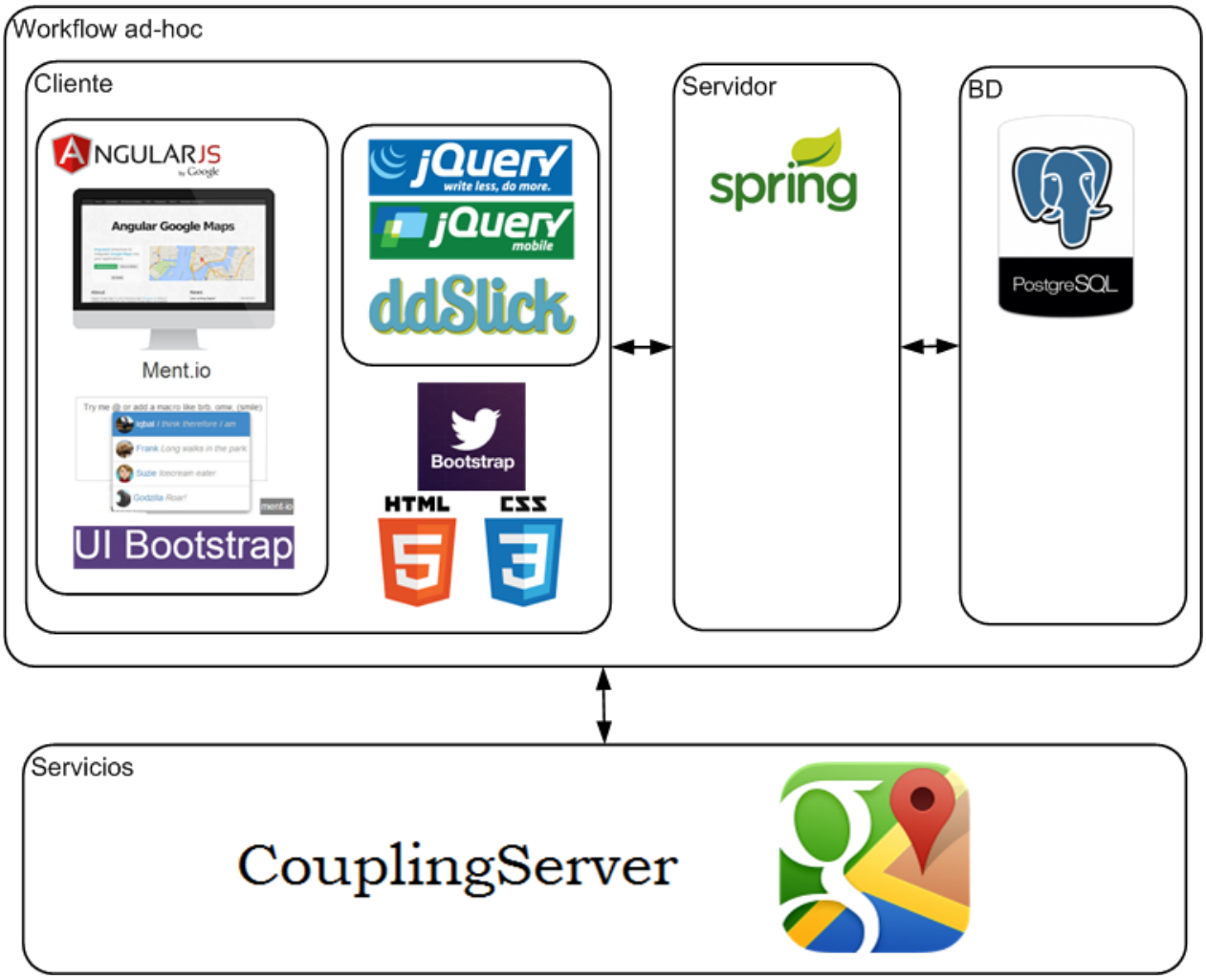


Figura 8: Recursos utilizados

A continuación se detalla cada uno de las tecnologías utilizadas:

- Spring MVC: **FRAMEWORK** de código abierto para desarrollo de aplicaciones y contenedor de inversión de control para la plataforma **JAVA**. Esto último quiere decir que permite la configuración y la administración del ciclo de vida de los objetos Java, llevándose a cabo a través de la inyección de dependencias. Implementa el patrón **MODELO VISTA CONTROLADOR** (**MVC**), basado en **HTTP** y **SERVLETS**.
- Spring Tool Suit: Ambiente de desarrollo basado en **ECLIPSE**, personalizado para desarrollar aplicaciones sobre **SPRING**. Para el presente trabajo de memoria se utilizó la versión 3.4.0 para Windows.

- JavaEE: En ingles **JAVA PLATFORM ENTERPRISE EDITION**. Es una plataforma de programación para desarrollo e implementación de software de aplicaciones en el lenguaje de programación java.
- CSS: En ingles **CASCADING STYLE SHEETS**. Es un lenguaje utilizado para definir la presentación de un documento estructurado en **HTML** o **XML**. La idea de utilizar **CSS** es separar la estructura del documento de su presentación.
- JavaScript: Es un lenguaje de programación interpretado y orientado a objetos basado en prototipos, imperativos, incluyendo algo de tipos y dinámico. **JAVASCRIPT** es utilizado principalmente por el lado del cliente implementado como parte del navegador web mejorando la interfaz de usuario y páginas web dinámicas.
    - AngularJs: Es un **FRAMEWORK JAVASCRIPT** que implementa el patrón de programación **MVC**. Adapta y amplia el **HTML** tradicional para utilizar de mejor manera el contenido dinámico, a través de **DATA-BINDING BIDIRECCIONAL** permitiendo así la sincronización automática entre modelo y la vista. Tiene un rol protagónico dentro del desarrollo de la aplicación ya que se utiliza para describir toda la lógica de los procesos, flow, geolocalización y de la interfaz de usuario.
        - Angular Google Maps: Conjunto de directivas escritas en **ANGULARJS** el cual integra **GOOGLE**. Utiliza la versión 3 de **GOOGLE MAPS**. Se utiliza para gestionar la configuración y utilización de **GOOGLE MAPS** en la aplicación.
        - UI Bootstrap: Modulo o directiva de **ANGULARJS** que implementa componentes de **BOOTSTRAP** escritos sólo con componentes de Angular. No depende de las librerías principales de **BOOTSTRAP**, necesitando como dependencia solo **ANGULARJS** y **CSS** de **BOOTSTRAP**.

- Ment-io: Directiva de **AngularJS** la cual es aplicable a un elemento del **DOM** el cual acepte ingreso de texto seleccionable, como un input text. Al escribir texto en el campo al cual se aplicó la directiva, se realizó una búsqueda sobre un conjunto de datos, de los cuales se sugiere los que contengan el texto ingresado. Además soporta trigger específicos a un determinado carácter. En la aplicación, esta directiva es utilizada para sugerir nombres de usuario al ingresar el carácter @, o para sugerir el nombre de un flow luego de ingresado el carácter ^, en el cuadro de texto en donde se ingresan los tweet.

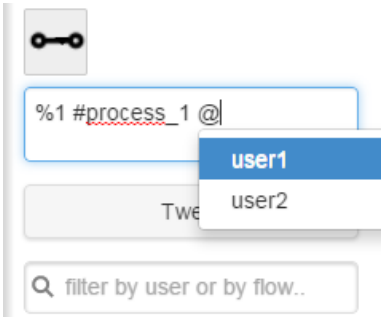


Figura 9: Ejemplo de utilización de Ment-io

- Angular-jqm: Directiva de **AngularJS** para incluir las funcionalidades que provee **jQuery Mobile**. Cumple la función de proveer los paneles laterales, propios de **jQuery Mobile**, sin causar conflicto con las otras librerías.

- Google maps v3: **API Javascript** de **Google Maps** permite insertar **Google Maps** en una página web proporcionando diversas utilidades para manipular los mapas. Es posible añadir contenido en el mapa mediante diversos servicios, permitiendo crear solidas aplicaciones de mapas.
- JQuery: Biblioteca de **Javascript** que permite simplificar la manera en que **Javascript** interactúa con los documentos **HTML** manipulando el árbol del **DOM**, manejar eventos, desarrollar animaciones y agregar

interacción a través de **AJAX**. En la presente aplicación es utilizado principalmente para la comunicación entre la vista y el servidor a través de **AJAX**.

  - ddSlick: Plugin de **JQUERY** que permite un *dropdown* personalizado con imágenes y descripción. En la aplicación se utiliza para modelar el *dropdown* de donde los usuarios escogerán el tipo de patrón de **WORKFLOW** que desean emplear.

- CouplingServer: Framework que se encarga de la gestión y administración de la propagación de los mensajes enviados entre los distintos actores participantes de un proceso. Su principal componente es **COUPLINGMANAGER**, que se encarga de administrar el flujo del mensaje, procurando que sean validados, persistidos y propagados a los clientes correspondan (6).
- PostgreSQL: Es un sistema de administración de base de datos relacional, orientado a objetos.

2.3. Dispositivos utilizados

Para realizar las pruebas se montó un servidor sobre un computador con las siguientes características:

- Procesador: Intel Core i5-3330 CPU @ 3.00GHz 3.20 GHz
- Memoria RAM: 8 GB
- Disco Duro: 464 GB
- Sistema Operativo: Windows 8.1 Pro de 64 bits

Para conectarse al servidor, un dispositivo táctil móvil con las siguientes características:

- Procesador: Chip Apple A4 a 1 GHz
- Capacidad: Memoria flash de 16, 32 o 64 GB
- Video: Vídeo H.264 de hasta 720p, 30 fotogramas por segundo
- Otras especificaciones:
    - UMTS/HSDPA (a 850, 1.900 y 2.100 MHz)
    - GSM/EDGE (a 850, 900, 1.800 y 1.900 MHz)
    - Solo datos2
    - Wi-Fi (802.11 a, b, g y n)
    - Tecnología Bluetooth 2.1 + EDR

3. ESPECIFICACION DEL PROBLEMA

3.1. Descripción del problema

Este trabajo de título tiene como objetivo plantear una solución de software para el apoyo a los procesos de negocio georreferenciados. Como ya se ha planteado en los capítulos anteriores, cada día más empresas toman la decisión de mejorar sus procesos de negocios incorporando una disciplina BPM para su modelamiento y posterior implementación con sistemas de software que apoyan la ejecución de estos procesos en el quehacer diario de la organización. Sin embargo, podemos ver que no siempre los procesos definidos pueden ser ejecutados tal cual se modelaron e implementaron. Hay actividades que necesariamente, en ocasiones especificas del negocio, deben ser vistas como una gran macro actividad y es necesario crear un subproceso ad hoc asociado a ésta para su ejecución. La mayoría de las veces este proceso ad hoc, además de poder ser diseñando con las metodologías y buenas prácticas que propone BPM, aparece una variable que no estaba contemplada en el proceso de negocio padre. Esta variable es una ruta geográfica donde las tareas se van a ejecutar de manera distinta en función de donde se realicen geográficamente. La ruta en sí misma es el proceso y las tareas se van ejecutando de acuerdo a la ubicación de donde se debe realizar dicha tarea en esta ruta.

El presente trabajo tiene como objetivo construir una solución de software que apoye la creación y ejecución de este subproceso o definitivamente procesos ad-hoc, donde hay que considera la tensión que se produce entre un proceso modelado en función del flujo de tareas versus un proceso modelado en función de un flujo de ubicaciones geográficas donde las tareas deben ser coordinadas de acuerdo al lugar geográfico donde será ejecutada. Para apoyar esta coordinación se incluirá en esta solución de software los conceptos y técnicas aplicadas en los sistemas de microbloggins, donde no sólo apoyara con información asociada a una tarea y a un punto en el

espacio geográfico definido, sino que también coordinara la comunicación entre los distintos actores del proceso.

3.2. Requisitos de la solución a construir

BPM presupone la existencia de dos núcleos constituyentes: **PROCESS AWARE INFORMATION SYSTEM** (**PAIS**) y modelación de procesos (13). El acrónimo **PAIS** se refiere a una categoría de sistemas que adopta una visión de proceso en donde los objetivos del negocio son descompuestos en un número finito de actividades. En este modelo las actividades son coordinadas a través de patrones **WORKFLOW** (5).

Por otro lado, **GIS** propone dos conceptos: transformación de la información y visualización. Transformación de la información se refleja en la representación de la geolocalización, mientras que la visualización provee de conciencia en el ámbito espacial en el que se realizan los **FLOWS**. Dado esto, es posible describir un **FLOW** con una pequeña leyenda y adjuntarle una ubicación para ser asignado a un usuario. Si este usuario se mueve luego a otra ubicación entonces esta sería un nuevo **FLOW**.

Otra forma de modelar esto, con el fin de simplificarlo es, si se tiene una tarea que se realiza en múltiples geolocalizaciones, entonces asociar múltiples geolocalizaciones a una determinada tarea. De esto nace el primer requerimiento.

1. Permitir asociar geolocalizaciones a un **FLOW**.

Por otro lado, **GIS** no impone dependencias entre tareas y se sugiere que un **BPM AD-HOC GEORREFERENCIADO** no debería imponer restricciones sobre como los usuarios interactúan sobre los elementos georreferencias. De esto nace el segundo requerimiento:

2. **WORKFLOW AD-HOC GEORREFERENCIADO** debe considerar que los usuarios determinen el orden de los **FLOWS** y georreferencias de éstas.

También se busca interacción por parte del grupo de actores participantes de un proceso.

3. **WORKFLOW AD-HOC GEORREFERENCIADO** puede ser implementado sobre una plataforma basada en **MICROBLOGGING** en donde los usuarios intercambien ideas, además de la información intrínseca del proceso.

Y, por supuesto, manejar un lenguaje en común para que sea posible la comunicación la administración de los componentes que envuelve una tarea desprendiéndose los siguientes requerimientos:

4. **WORKFLOW AD-HOC GEORREFERENCIADO** debe manejar una sintaxis basada en patrones de **WORKFLOW**, además de un conjunto de reglas para simular el motor Workflow.

Además del lenguaje de patrones Workflow se debe, también, aplicar reglas sobre estos patrones, de tal forma de llevar la administración de los procesos. En la imagen de la Figura 12 es posible ver cómo se comportará un proceso de acuerdo al estado en el que se encuentre.

5. **WORKFLOW AD-HOC GEORREFERENCIADO** debe manejar lenguaje similar a Twitter, vale decir, expresiones similares a HashTag, en donde se utiliza un carácter especial para gatillar un **TRIGGER**. Los caracteres utilizados serian @ para los nombres de usuario, ^ para las tareas y # para los procesos.

4. W-adhoc: Aplicación para gestionar Workflows ad-hoc

En esta sección se detalla cómo se llevó a cabo la aplicación, describiendo los aspectos más relevantes de **W-ADHOC**.

4.1. Estructura General de la Plataforma

Basado en las estructuras de capas encontradas en los trabajos que satisfacían similares requerimientos no funcionales, en W-adhoc se definieron las siguientes capas:

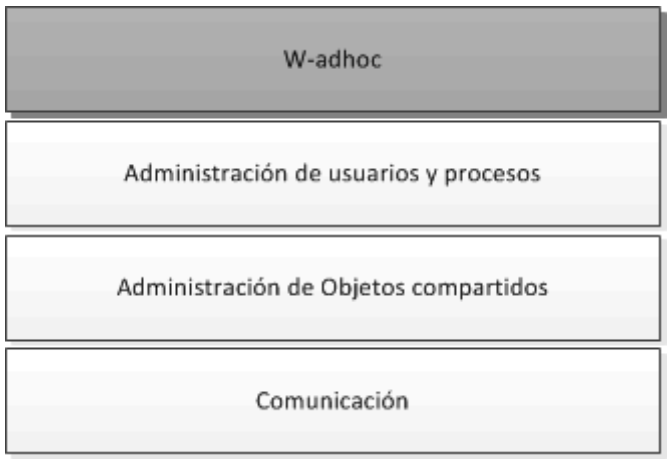


Figura 10: Capas de W-adhoc

- **CAPA DE ADMINISTRACIÓN DE USUARIOS Y PROCESOS**, se encarga de almacenar y gestionar la información de los usuarios y de los procesos en los que participa cada usuario, relacionándolos.
- **CAPA DE ADMINISTRACIÓN DE OBJETOS COMPARTIDOS**, capa que administra y controla la interacción entre los usuarios. Se encarga de organizar y dejar disponibles los objetos pertenecientes a los procesos, así como los estados de éste. Cada vez que un usuario ingresa a un proceso, esta capa cumple la función de actualizar al usuario con el último estado del proceso.
- **CAPA DE COMUNICACIÓN**, que permite el envío de mensajes a través de **COUPLINGSERVER**.

A continuación, se procederá a explicar en detalle cada una de estas capas.

4.2. Diseño de la Capa de Administración de usuarios y procesos

Esta capa es la que administra y gestiona los usuarios y los procesos, permitiendo a un usuario crear una cuenta en la aplicación, iniciar sesión con una cuenta creada previamente, recordar y cargar los procesos en los que ha participado un usuario.

La capa se divide en los siguientes módulos:

- Definición de la capa de modelo de datos. Esta capa vive en la capa de Modelo del framework utilizado para el desarrollo de la aplicación que está basado en el patrón de diseño Modelo Vista Controlador. Esto se aborda con más detalle en la sección 4.5. Arquitectura del software.
- Persistencia y acceso de la información en la BD. Descrita con mayor detalle en el apartado 4.6. Diseño de estructura de datos.
- Integración para comunicación con la Capa de Administración de Objetos compartidos. En la sección 4.5. Arquitectura del software utilizada se introduce al por qué se escogió la tecnología utilizada para llevar a cabo esta comunicación.
- Implementación de patrones Workflow para permitir la gestión y administración de los procesos.

A continuación se detallara el último punto mencionado en la lista anterior.

4.2.1. Administración de Procesos

En la sección 1.1 se quiso dar una breve reseña de lo que serían la gestión de los procesos a través de patrones workflow. En esta sección se pretende explicar de manera detallada a que corresponde cada uno de estos patrones y como se relacionan entre ellos.

Con el fin de administrar los procesos, se implementaron los patrones workflow descriptos en la tabla de la Figura 11:

Patrón	Descripción
Start	Patrón de workflow creado para iniciar un proceso.
Sequence	Patrón que se utiliza para indicar que la tarea sigue de manera secuencial a la tarea anterior, luego de que ésta fue completada.
Split	Se utiliza para separar el flujo de uno secuencial a un flujo paralelo. No se define como la creación de dos o más nuevas tareas, sino más bien como la creación de una nueva tarea que se realiza en paralelo a la última tarea secuencial creada.
Join_a	Punto de convergencia de múltiples tareas. En este caso se menciona que se ha terminado la tarea para que el coordinador del proceso haga el ***merge***. Este merge es en manera figurativa, ya que no se creó un patrón para cerrar el proceso, sino más bien el usuario coordinados, que se denoto como el creador del proceso, debe asegurarse de que la tarea que queda activa en ese momento sea completada para poder continuar con el proceso.
Join_b	Otra manera de converger tareas en paralelo. En este caso sólo se informa el término de la tarea al usuario que se encuentra realizando la otra tarea en paralelo con el fin de comunicar que el flujo se ha unificado nuevamente.
Finish	Patrón utilizado para cerrar los procesos. Estos pueden ser cerrados en cualquier momento sólo por el usuario que le dio vista a éste.

Figura 11: Patrones workflow utilizados

Con el fin de relacionar los patrones antes relatados, se definió el diagrama de estados de la Figura 12, en donde los patrones *Start* y *Finish* están representados por los puntos de comienzo y fin del diagrama.

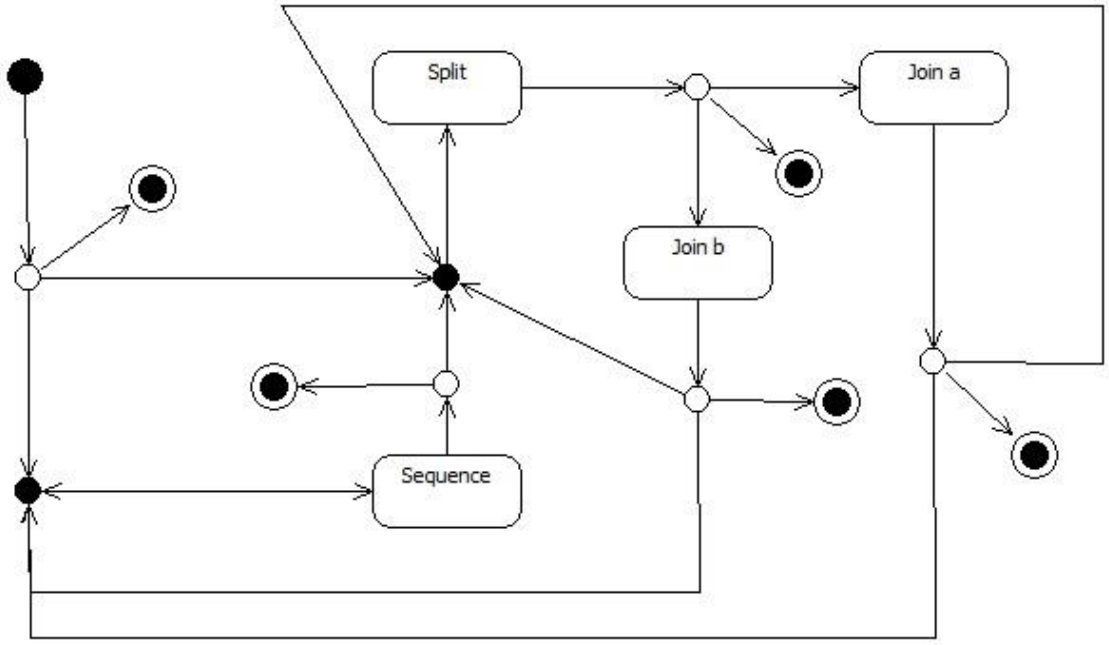


Figura 12: Diagrama de estados

En el diagrama se puede ver que al comenzar un proceso no es posible utilizar los patrones *Join_a* y *Join_b*. Esto es lógico, ya que al crear un proceso no existe una bifurcación antes comenzada con el patrón *Split*.

Por otro lado, es posible finalizar un proceso en cualquier momento con el patrón Finish. Esta acción solo es posible que la lleve a cabo el usuario propietario del proceso. Otra observación importante de detallar es que luego de un patrón *Split* solo es posible aplicar un patrón *Join_a* o *Join_b*. Esto es porque se intenta mantener un flujo secuencia, entonces, al crear uno o más flujos paralelos, es necesario volver a un flujo secuencial para comenzar o continuar otro flujo paralelo.

4.3. Administración de la Capa de Objetos Compartidos

Esta es la capa que contiene la mayor parte de la lógica de la aplicación. Administra y controla la concurrencia de la información que fluye dentro de un proceso y que ésta sea mostrada de manera congruente dentro de cada

sesión de cada usuario conectado paralelamente. Esta capa contempla las siguientes funcionalidades:

- Integración con la capa de Administración de usuario y procesos.
- Integración con la capa de comunicación y persistencia de los Objetos compartidos.
- Estructura de datos de los objetos compartidos
- Despliegue de los objetos compartidos para ser visualizados en GIS.

A continuación, se especificaran los objetos compartidos.

4.3.1. Tweet

Como se describió con antelación, un tweet es un mensaje corto emitido por un usuario de la aplicación. Este mensaje puede ser georreferenciado en una o muchas ubicaciones o zonas.

Un tweet tiene una serie de atributos y una estructura que lo completan. Entre ellas se contempla el usuario creador de éste, pero lo más importante es la estructura que este contiene. En la se vuelve a recordar las expresiones regulares que definen a cada patrón.

Patrón	**Expresión regular**	**Descripción**
Start	#.*	Creación de un proceso
Sequence	%1 #.*[*|@.* ^.*]	Creación de un Tweet secuencial
Paralel	%2 #.*[*|@.* ^.*]	Creación de flujo paralelo a la rama inicial
Join a	%3a #.*[*|^.*]	Volver a unir rama paralela a rama principal
Join b	%3b #.*[*|@.*]	Volver a unir rama paralela a rama principal
Finish	%11 #.*	Cierre de proceso

Figura 13: Definición de patrones de workflow utilizados

El software proporciona una validación de cumplimiento de estos patrones. Implementados a través de un directiva de **AngularJs**, ésta actúa sobre el objeto del DOM al cual se le desea aplicar como una validación *customizada*, en este caso el cuadro de texto que recibe la información del usuario.

Dentro de la estructura del tweet es posible encontrar palabras que comienzan con los caracteres '@' o '^' los que definen usuarios y nombres de tareas respectivamente. Cuando la aplicación recibe un tweet, ésta lo toma y lo desglosa para obtener la información de usuario a quien se le asigna la tarea y el nombre de la tarea, desde el tweet. Esta información es relacionada al tweet y estructurada en un objeto para su fácil acceso.

En el prototipo inicial de la aplicación se había pensado en georreferenciar un nombre de tarea en distintos tweets, para luego limitar esta *feature* y solo permitir georreferenciar un tweet y no un nombre de tarea, ya que si bien se puede realizar la misma actividad en diferentes ubicaciones, al pertenecer a otro tweet lo hace una nueva tarea, por lo que lo primero no tenía sentido.

4.3.2. Georreferencias y GIS

Para implementar las georreferencias, se utilizó la API de Google maps. Las georreferencias se asocian a un tweet, pudiendo ser ninguna, una o muchas georreferencias asociadas a un tweet. Una georreferencia se almacena como un conjunto de puntos los cuales se definen por un par (latitud, longitud).

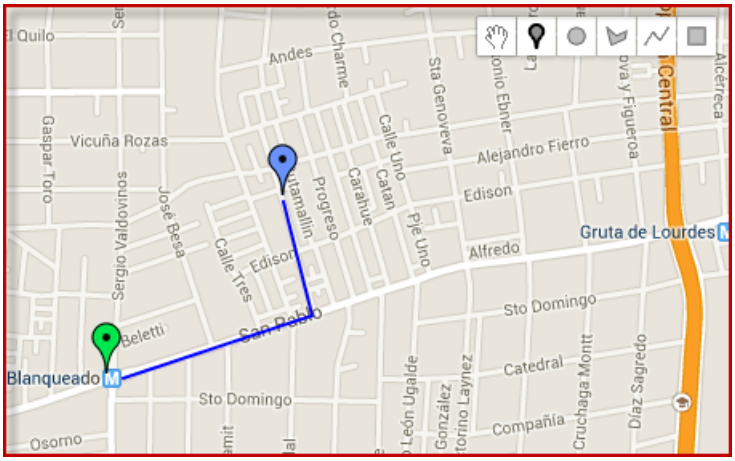
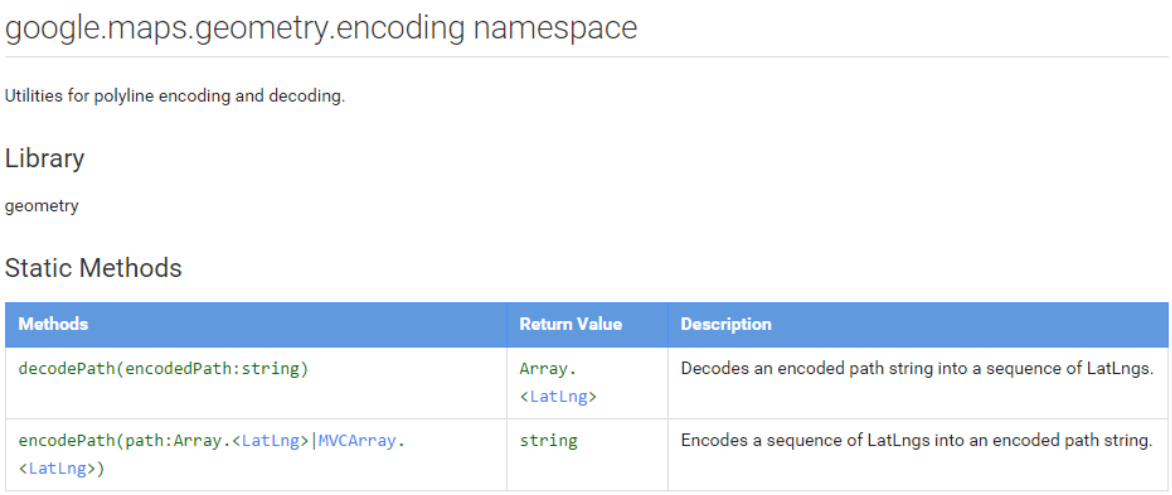


Figura 14: Georreferencias en Google maps.

Las figuras utilizadas fueron **MARKER**, **CIRCLE**, **POLYGONE**, **POLYLINE** y **RECTANGLE**. Uno de los problemas a enfrentar fue que, como es posible asociar muchas geo-localizaciones a un tweet, se hace pesada la cantidad de información a compartir entre los usuarios. Para agilizar este envío de información se

utilizó la función **encodePath** y **decodePath** provista por la API de Google maps.



google.maps.geometry.encoding namespace

Utilities for polyline encoding and decoding.

Library

geometry

Static Methods

Methods	Return Value	Description
decodePath(encodedPath:string)	Array.<LatLng>	Decodes an encoded path string into a sequence of LatLngs.
encodePath(path:Array.<LatLng>|MVCArray.<LatLng>)	string	Encodes a sequence of LatLngs into an encoded path string.

Figura 15: encodePath y decodePath

La función **encodePath**, toma un Array de objetos LatLng y los convierte en un string, por lo que el objeto resultante a enviar quedaba compactado a un objeto más reducido en tamaño, además de poderse enviar un conjunto de estos con mayor eficiencia a través de un objeto Json.

Luego al recibir el objeto, se procede a decodificar con la función **decodePath** y volver a crear el objeto geométrico que representa la geo-localización.

4.4. Capa de Comunicación

En esta sección se introducirá información sobre el funcionamiento de la capa de Comunicación. Cabe señalar que esta capa no es parte del proyecto **W-adhoc**. Para la comunicación de W-adhoc se utilizó **CouplingServer**.

CouplingServer es un framework de comunicación desarrollado para la implementación de **Mobiz**, desarrollado sobre un paradigma llamado *Objetos Acoplados*, que está específicamente pensado para ser usado en la adaptación de aplicaciones ya existentes para transformarlas en aplicaciones concurrentes

(6). Además, para el funcionamiento de **CouplingServer**, éste debe ser acompañado por:

- Un servidor, implementado con el framework antes mencionado.
- Una API JavaScript para HTML5 para la implementación de aplicaciones concurrentes y que utiliza el servidor antes mencionado para gestionar la persistencia de los modelos y la comunicación entre los participantes de la aplicación.

De esta manera se implementó la misma estrategia utilizada para el desarrollo de Mobiz: Primero crear una aplicación monousuario, para luego extenderla a múltiples usuarios.

4.5. Arquitectura del software utilizada

Dado que **W-adhoc** es una aplicación concurrente, se necesitó de una arquitectura robusta en donde no sea problema la interoperabilidad entre una cantidad indefinida de usuarios conectados paralelamente. Por ello el lenguaje y la plataforma escogidos fueron Java sobre el framework **Spring MVC**, basado en el patrón de diseño **Modelo Vista Controlador (MVC)**, se tienen tres aspectos relevantes:

- Modelo: Representando los datos o reglas de negocio, así como el estado de la aplicación.
- Vista: Representando la visualización de los datos al usuario, en un formato específico.
- Controlador: Procesando las acciones que el usuario realiza en la vista, para obtener los recursos necesarios desde el Modelo con el fin de poder responder los requerimientos hechos por el usuario.

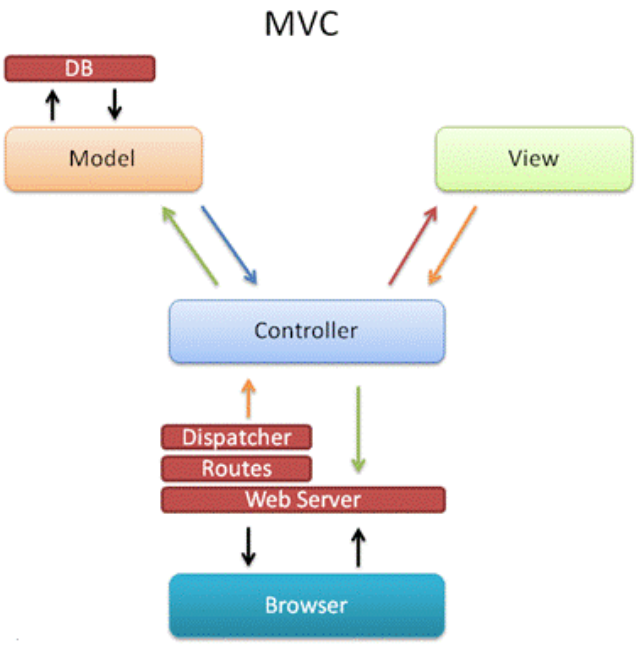


Figura 16: Arquitectura que propone patrón Modelo Vista Controlador

Además, **SPRING MVC** mejora la aplicación del patrón **MVC** para proveer una mejor experiencia a los usuarios que utilizan **JAVASCRIPT**, **AJAX** y el uso de los formatos especiales de información como son **JSON** o **XML**. En la Figura 17 se muestra el concepto de la mejoras en el patrón MVC.

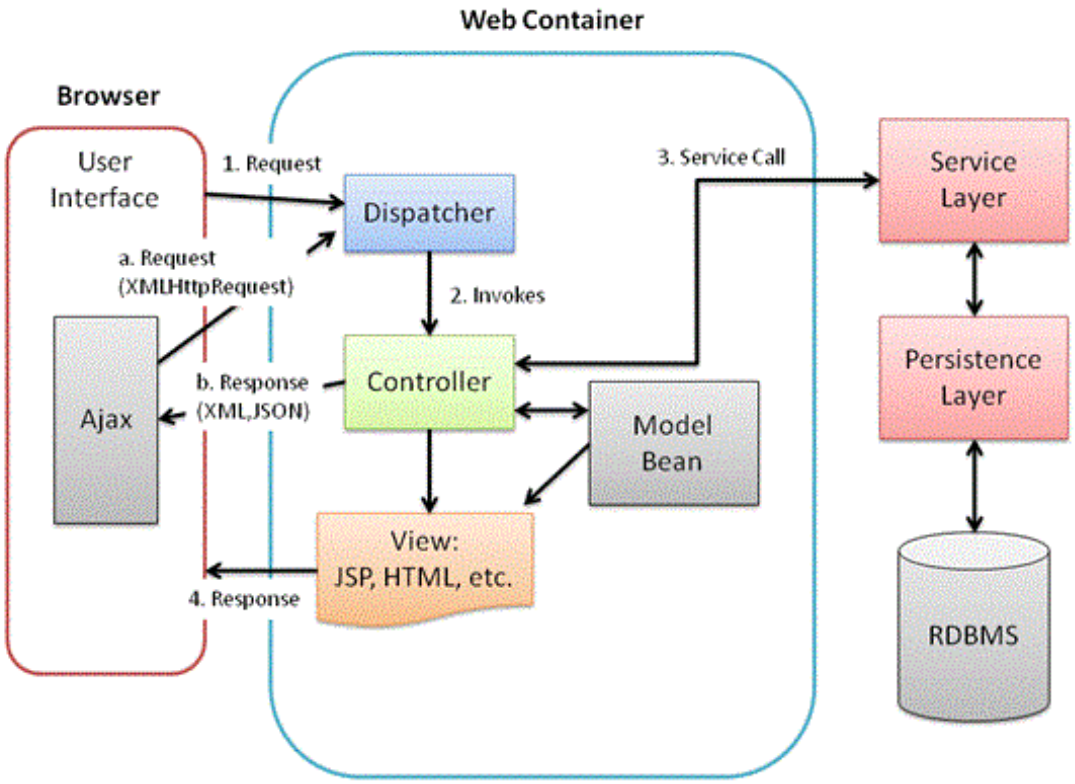


Figura 17: Concepto de mejora de utilización de JavaScript en Spring MVC

Aprovechando las bondades antes mencionadas, la mayor parte de la lógica de la aplicación se desarrolló en el lado del Cliente, es decir, en la capa de la Vista. Dentro de éste se utilizó un **FRAMEWORK JAVASCRIPT** con el fin de aumentar la mantenibilidad y modularidad del código. El Framework utilizado, llamado **ANGULARJS**, a su vez, también implementa un patrón **MVC** permitiendo controlar de mejor manera los elementos en el DOM.

La Figura 18 ilustra la estructura MVC de **ANGULARJS**, en donde las variables del *scope* vendría siendo el modelo, las que son declaradas dentro de cada controlador. Luego, en los elementos del **DOM**, se encapsula un trozo de éste asignándole un controlador por medio de *ng-controller*, en donde se podrá acceder a todas las variables del *scope* de dicho controlador.

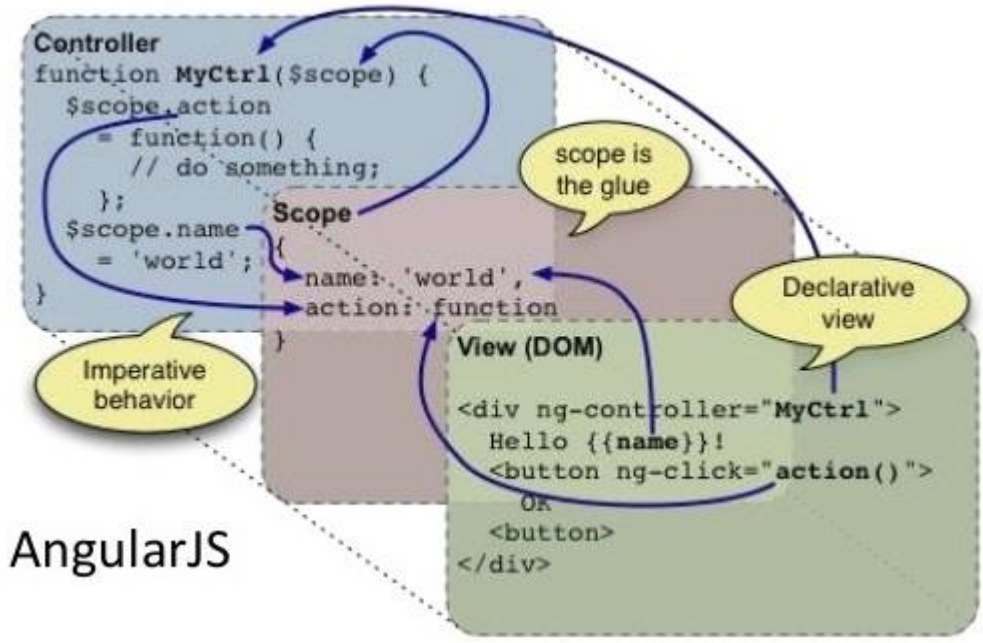


Figura 18: Arquitectura MVC del Framework AngularJs

4.5.1. Arquitectura de W-adhoc

En la imagen de la Figura 19 es posible apreciar la arquitectura de software con la que se desarrolló **W-ADHOC**. Desde la vista se consumen los servicios de **GOOGLE MAPS V3.0**, para ser manejados a través de **JAVASCRIPT**, al igual que la vista, quien es manejada en su mayoría por éste. Al ser una vista estática, es decir, una única página que cambia su contenido, la interacción directa entre la vista y el controlador es mínima. El grueso de la interacción con el controlador es manejado a través de **JAVASCRIPT** por medio de peticiones **AJAX**. Luego, el controlador, de acuerdo a las solicitudes tanto directas de la vista como de las peticiones **AJAX**, se comunica a la base de datos a través un **DATA ACCESS OBJECT** (**DAO**) con ayuda del **ENTITY MANAGER**, lo que obtiene el objeto de la BD mapeado a un **DATA TRANSFER OBJECT** (DTO), es decir, a un **OBJETO JAVA** puro.

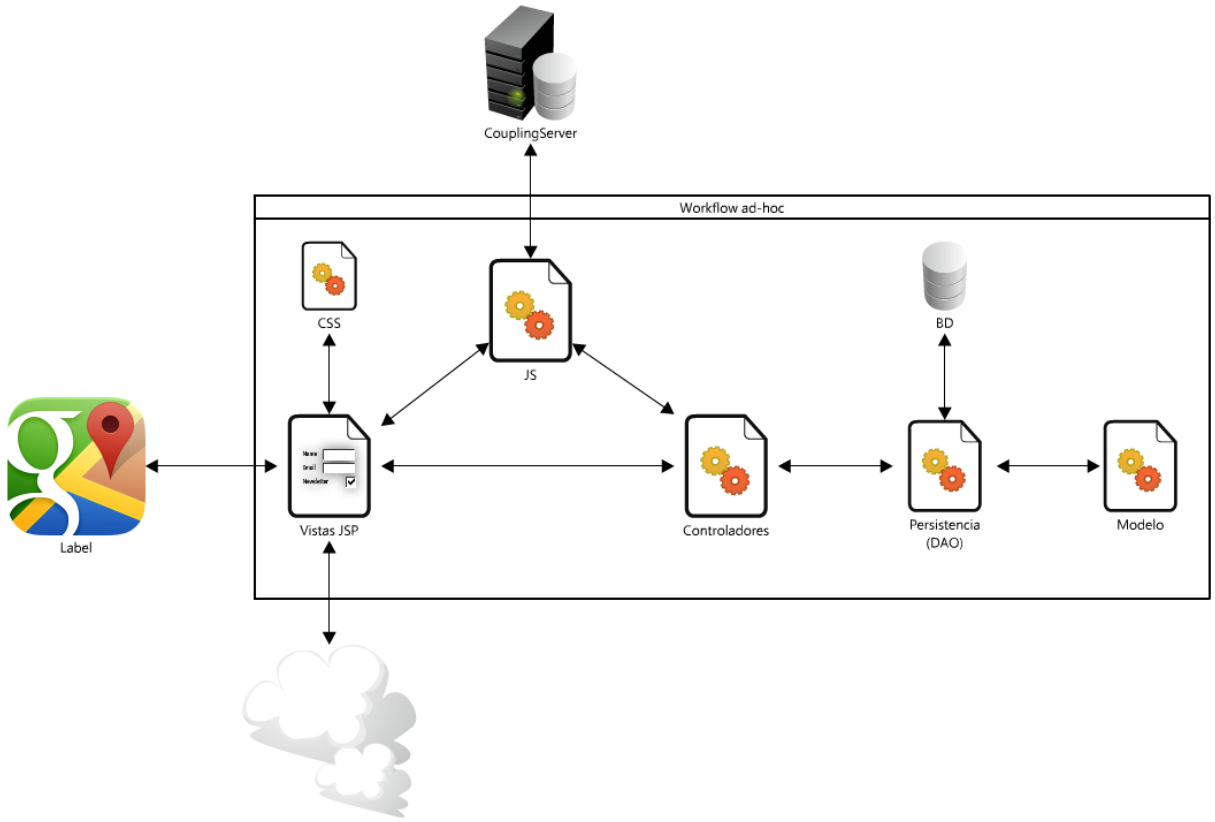


Figura 19: Arquitectura de Software de Workflow ad-hoc

Adicionalmente, dado que se utilizó un framework para ayudar a estructurar el código **JAVASCRIPT**, se describirá la estructura de los archivos que contienen la lógica y la utilización de éste. En la Figura 20 se muestran los archivos en donde se describe el modelo contenido dentro del **CLIENTE**. A continuación, se describirá cada uno de éstos y su importancia.

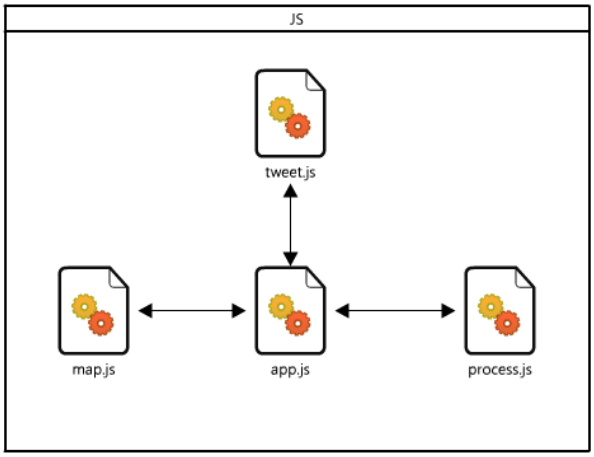


Figura 20: Estructura modelo en archivos Js

- process.js: Contiene la comunicación con **CouplingServer**. Se encuentra estructurado como un **Objeto Javascript** por medio *prototype*. Su principal funcionalidad es enviar y recibir la información hacia y desde **CouplingServer**, información que luego comparte con el controlador principal: **AppController**.

  La manera de comunicar es a través de la obtención del *scope* del controlador con el que se desea realizar la comunicación. El *scope* se obtiene obteniendo el elemento invocando la función `angular.element(idElemento).scope()`. Ya teniendo el *scope* del controlador es posible acceder a todas las funciones o métodos que éste contenga (Figura 21).

```
function Process(session){
    this.adapter = new ClientAdapter("/CoupledObjectWebServer/");
    this.session = session;
    this.adapter.joinSession(session);
    this.adapter.start(500);
    this.couple();
}

Process.prototype.preAddTweet = function(session, body, createBy, overlays){
    this.addTweet(this.session, body, this.getTime(), createBy, overlays);
};

Process.prototype.addTweet = function(session, body, createAt, createBy, overlays){
    //console.log("AcoPLANDO!!!" + session);
    var tweetId = null;
    var scope = angular.element($("#txt")).scope();
    scope.$apply(function(){
        tweetId = scope.addTweetToModel(session, body, createAt, createBy);
    });

    console.log("addTweetToModel: overlays>  ");
    console.log(overlays);

    var scopeMap = angular.element($("#map")).scope();
    scopeMap.$apply(function(){
        scopeMap.addTweetToModel(tweetId, overlays, null);
    });
};
```

Figura 21: Extracto de process.js

Adicionalmente, es necesario un método llamado *couple*, el que contiene la configuración para que **CouplingServer** acople los métodos que se le indique. En la Figura 22 se muestran los métodos acoplados: addTweet, closeSplit y endProcess.

```
Process.prototype.couple= function(){
    this.adapter.coupleObject("session",this,{
        messageType:"EVENT",
        explicitMapping:["addTweet", "closeSplit", "endProcess"]
    });
};
```

Figura 22: Configuración para CouplingServer

- app.js: Contiene la definición del controlador principal: **AppController**. Aquí se encuentra la mayor parte de la lógica de interacción con la vista como los filtros de tweets, la carga de los procesos en los que participa un usuario, agregar amigos a un proceso, entre otros.
- map.js: Incluye toda la configuración del mapa de **Google Maps**. Utiliza para ello, una directiva desarrollada en **AngularJs**. Interactúa directamente con la vista. En la imagen de la Figura 23 se muestra la interacción del controlador con el elemento del **DOM** que contiene el mapa.

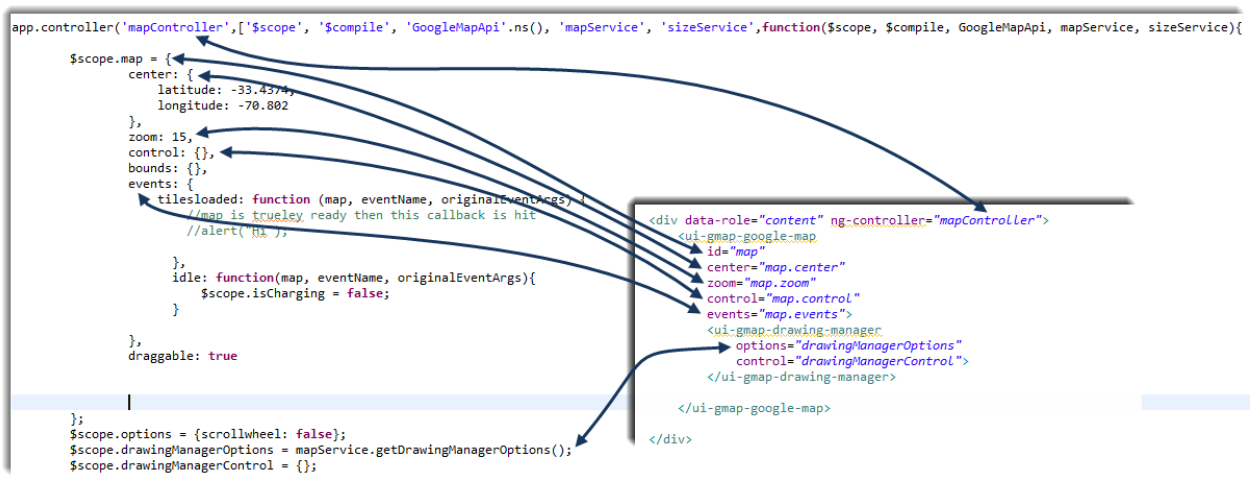


Figura 23: Extracto de interacción mapController en map.js con elementos del DOM

- tweet.js: Incluye el modelo y las funciones para que la vista interactúe con los **Tweets**.

En la Figura 24 se adjunta una imagen completa de la estructura del proyecto en el **IDE Spring Tool Suite**.

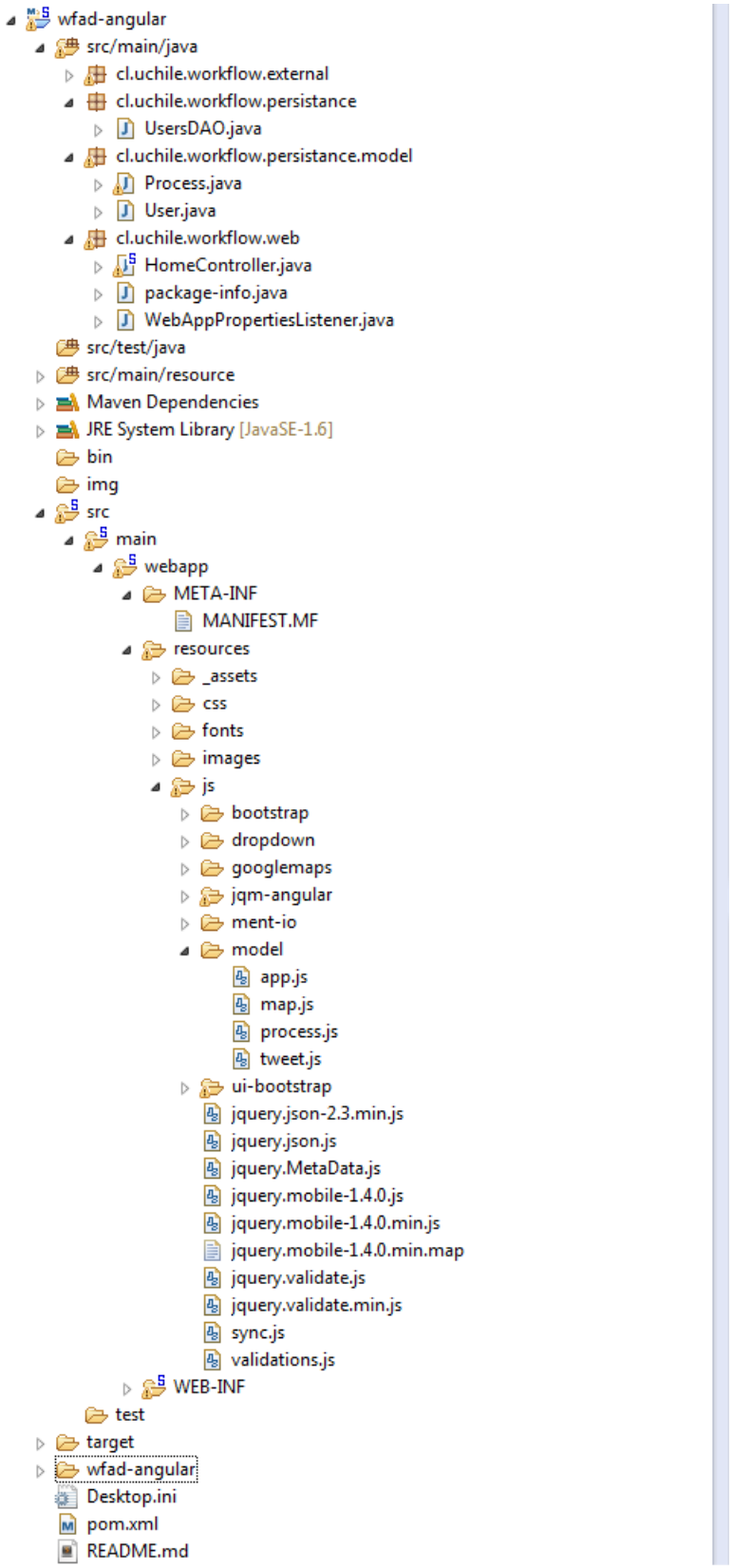

wfad-angular
src/main/java
cl.uchile.workflow.external
cl.uchile.workflow.persistance
UsersDAO.java
cl.uchile.workflow.persistance.model
Process.java
User.java
cl.uchile.workflow.web
HomeController.java
package-info.java
WebAppPropertiesListener.java
src/test/java
src/main/resource
Maven Dependencies
JRE System Library [JavaSE-1.6]
bin
img
src
main
webapp
META-INF
MANIFEST.MF
resources
_assets
css
fonts
images
js
bootstrap
dropdown
googlemaps
jqm-angular
ment-io
model
app.js
map.js
process.js
tweet.js
ui-bootstrap
jquery.json-2.3.min.js
jquery.json.js
jquery.MetaData.js
jquery.mobile-1.4.0.js
jquery.mobile-1.4.0.min.js
jquery.mobile-1.4.0.min.map
jquery.validate.js
jquery.validate.min.js
sync.js
validations.js
WEB-INF
test
target
wfad-angular
Desktop.ini
pom.xml
README.md

Figura 24: Estructura del proyecto Workflow ad-hc

4.6. Diseño de estructura de datos

En este apartado se procede a especificar el diseño de datos en la aplicación (Capa de Modelo)

Dado la presencia de Spring Framework en la aplicación y que éste soporta integración con **HIBERNATE**, **JAVA PERSISTANCE API** (**JPA**) y **JAVA DATA OBJECT** (**JDO**) para administrar los recursos, además de implementar **DATA ACCESS OBJECT** (**DAO**) es posible configurarlos para soportar **OBJECT RELATIONAL MAPPING TOOLS** (**ORM**) a través de la inyección. Vale decir, **HIBERNATE** permite mapear los objetos, desde la base de datos a un objeto **JAVA**.

Por ello, la estructura de datos de **W-ADHOC** es en base a orientación a objetos. En la sección 7, apartado 0 se ha dejado de estructura de las clases a la cual se mapean los objetos de la base de datos.

4.7. Diseño de la Base de Datos

En la presente sección se muestra el diseño de la base de datos para **W-ADHOC**. En la imagen de la Figura 25, se puede ver las tablas que componen la base de datos. Estas tablas son generadas automáticamente por **HIBERNATE** en **SPRING**.

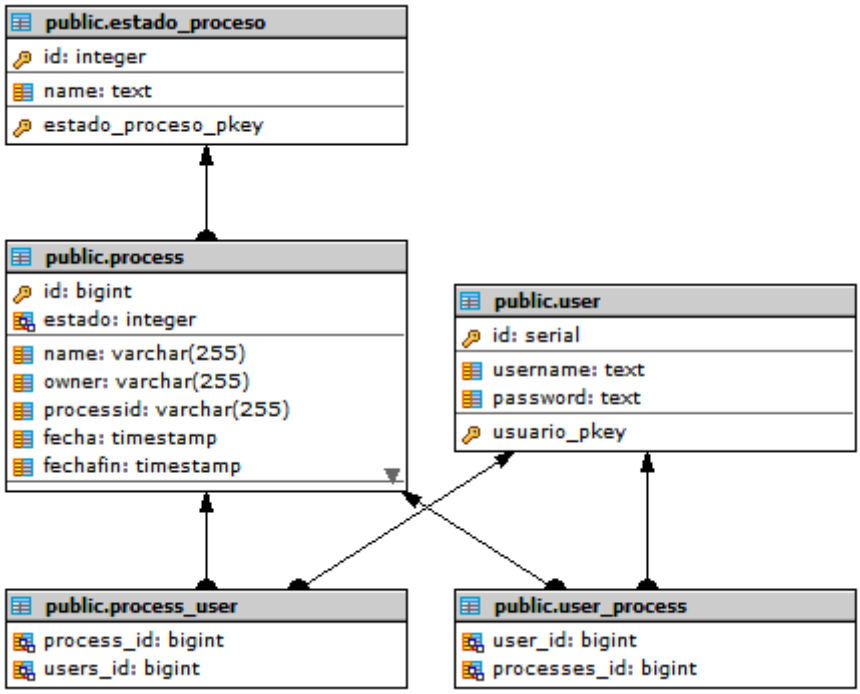


Figura 25: Modelo de Datos en BD

A continuación se describirá cada una de las tablas y su rol.

4.7.1. process

Incluye la información de los procesos. Entre sus campos incluye:

- nombre: Nombre del proceso
- owner: Propietario y creador del proceso.
- processId: Es el nombre pero como un *string* sin el carácter espacio.
- fecha: Fecha de creación del proceso
- fecha_fin: Fecha de finalización del proceso
- estado: Estado en el que el proceso se encuentra. Se describirá con mayor detalle en la siguiente sub-sección.

4.7.2. estado_proceso

Tabla con datos en los que se guarda el estado en el que puede estar un proceso. Sus campos son *id* y una breve descripción del estado. En la tabla de la Figura 26, se puede ver la información que contiene *estado_proceso*.

Id	Nombre
1	Start
2	Sequence
3	Split
4	Merge a
5	Merge b
6	Close

Figura 26: Tabla estado_proceso

4.7.3. user

Esta tabla contiene la información de **LOGIN** de los usuarios. Entre sus campos están:

- id: Identificador de la tabla
- username: Nombre de usuario. Necesario para el login de usuario.
- password: contraseña de usuario. Necesaria para login de usuario.

4.7.4. process_user y user_process

Ambas tablas son creadas por **HIBERNATE** de **SPRING**, por lo que, en estricto rigor, solo se necesitaría una de ellas, es por ello que se describirán dentro del mismo apartado. Tabla que contiene la relación de muchos es a muchos entre las tablas *user* y *process*.

4.8. Diseño de la interfaz de usuario

En el siguiente apartado se incluirán las interfaces de usuario pertenecientes a **W-ADHOC**.

4.8.1. Login de Usuario

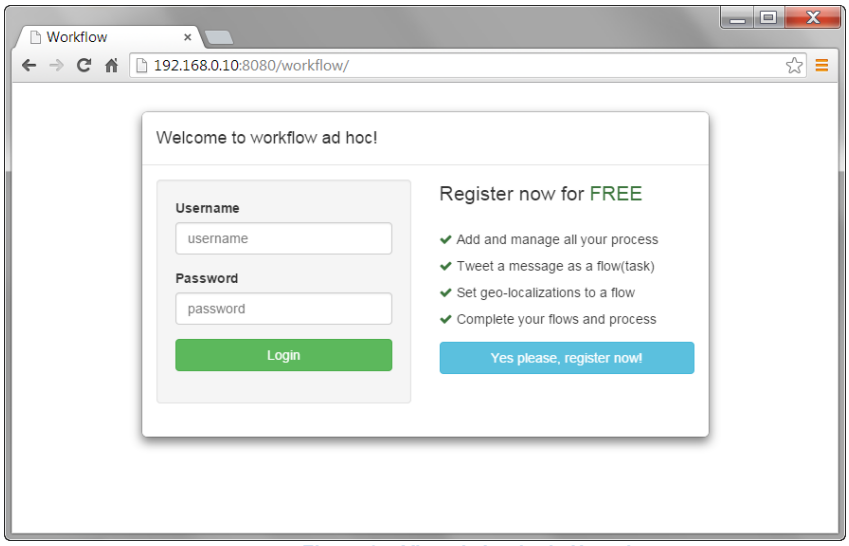


Figura 27: Vista de Login de Usuario

Al ingresar a la aplicación, la primera vista que se muestra es la de Login de usuario. Como su nombre lo indica, está página permite iniciar sesión a los usuarios ya registrados. Como input, pide el nombre de usuario y la contraseña manejando validaciones de éstos en los casos que sean incorrectos (ver Figura 28). Si un usuario no se encuentra registrado entonces puede hacer click sobre el botón **YES PLEASE, REGISTER NOW!** para ingresar al formulario de Registro.

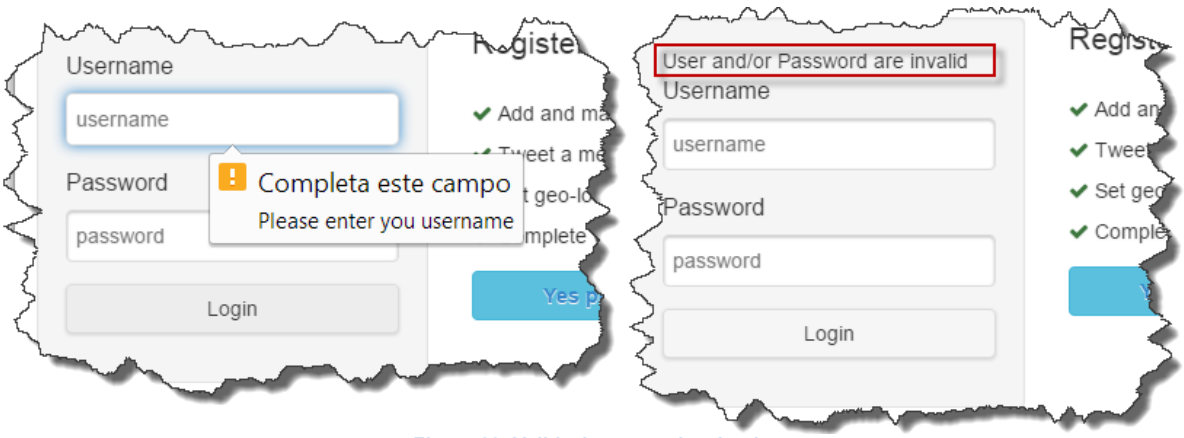


Figura 28: Validaciones en vista Login

4.8.2. Registro de Usuario

Cuando un usuario quiere registrarse, para tal efecto, debe ingresar a la vista de registro de usuario. En ella se le pedirá de manera obligatoria un usuario y una *password*. Ésta última deberá ingresarla dos veces para disminuir la probabilidad de ingresar una *password* y luego no recordarla.

En la imagen de la Figura 29 se puede ver el formulario de registro de usuario.

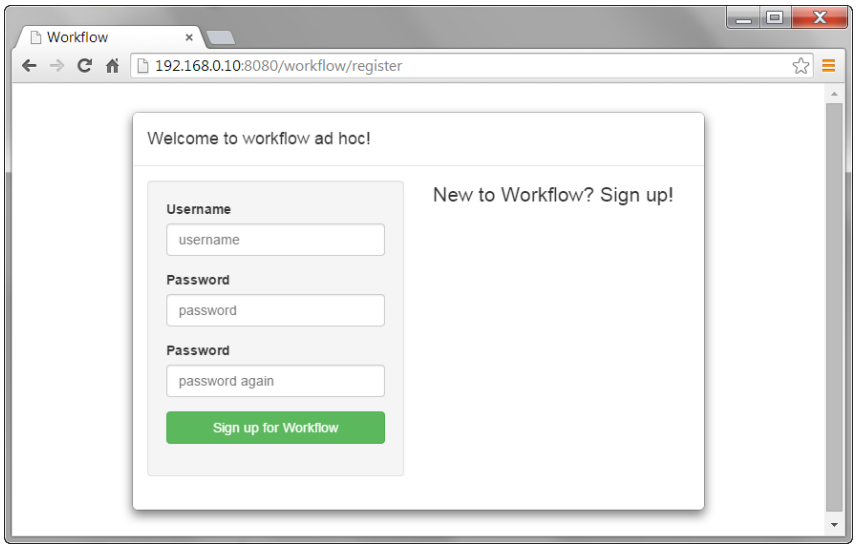


Figura 29: Vista de Registro de Usuario

4.8.3. Mapa

Concluyendo el registro e iniciado sesión, el usuario vera la vista de la imagen de la Figura 30, en donde se le da la bienvenida. Inicialmente el usuario no se encuentra dentro de un proceso, por lo que las acciones que puede realizar son:

1. Ingresar a un proceso ya creado o
2. Crear un nuevo proceso.

Recién ingresando a un proceso se le permitirá invitar otros usuarios a éste.

En esta vista el usuario tiene dos paneles laterales, en donde se le muestra una lista de los procesos en los que está participando y puede ingresar a ellos. Desde uno de los paneles también está la funcionalidad invitar a un

amigo, que sólo se muestra visible si y sólo si el usuario se encuentra dentro de un proceso.

En la imagen de la Figura 31 se muestra el panel Opciones, en donde es posible realizar estas acciones.

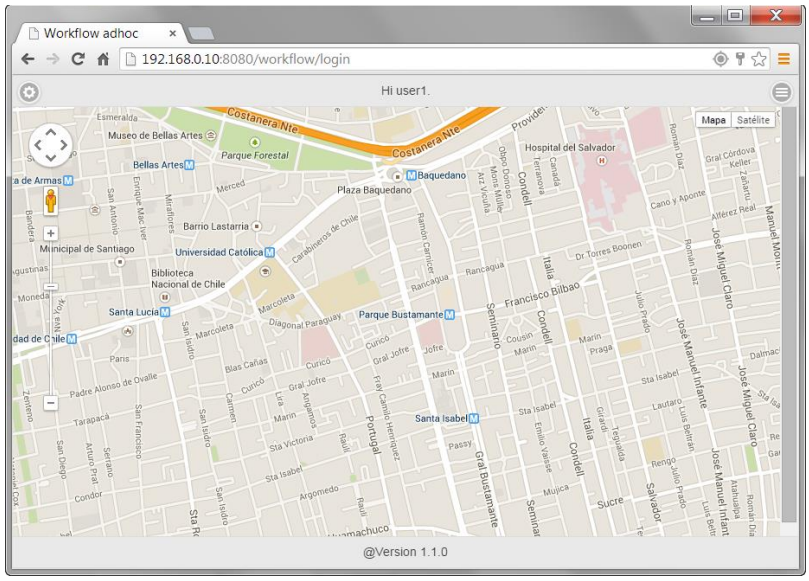


Figura 30: Vista principal: Mapa

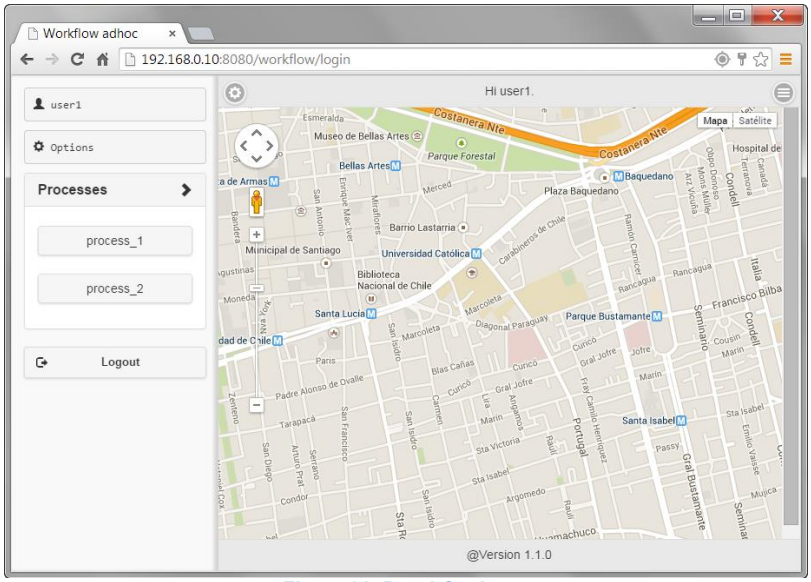


Figura 31: Panel Opciones

Luego, en la imagen de la Figura 32, es posible ver el otro panel que incluye el chat para escribir los mensajes con la estructura **WORKFLOW**.

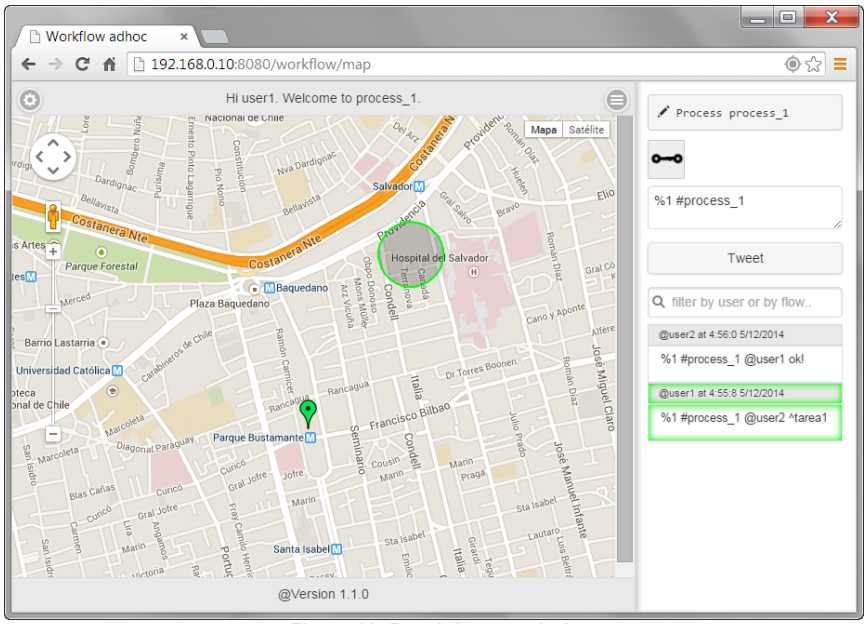


Figura 32: Panel de mensajería

Si un usuario envío un Tweet georreferenciado, luego, al recibirlo, podrá ver la georreferencia haciendo *click* sobre el tweet, el cual se pintará de un color característico para diferenciarse de las georreferencias de los demás **TWEETS**. En la imagen de la Figura 33 es posible ver un ejemplo de esto.

Figura 33: Diferenciación de georreferencias

Finalmente, se deja una imagen resumiendo los apartados de la vista:

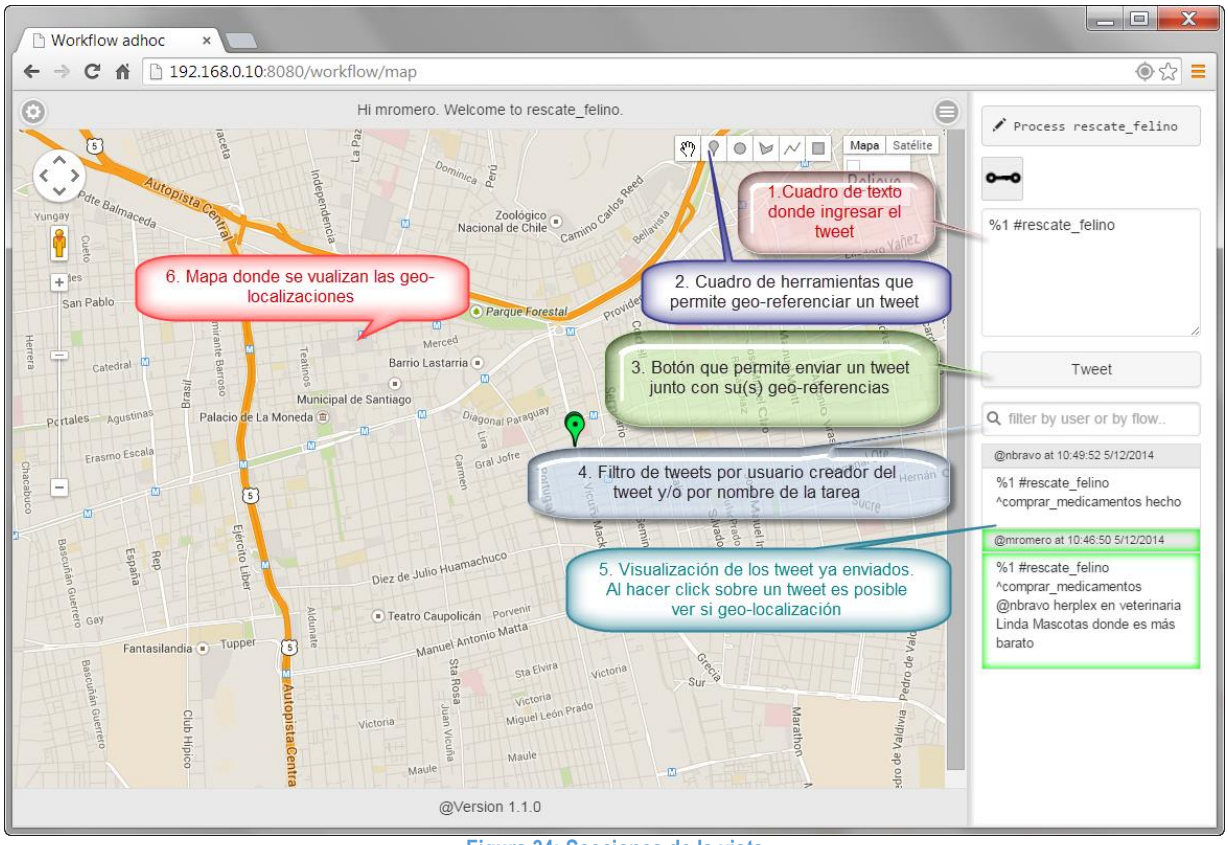


Figura 34: Secciones de la vista

4.8.4. Justificación del diseño de la interfaz de usuario

El sistema desarrollado en este trabajo de título es un sistema de apoyo a la ejecución de flujos de trabajo (workflows), donde el usuario es un usuario de operaciones, que está concentrado en la ejecución de la tarea y en sus resultados. Por lo tanto, el sistema de software que apoya este objetivo principal, que es la eficacia y eficiente ejecución del flujo de tareas que conduce a concretar un proceso de negocio donde el resultado de este proceso afectará directamente a un cliente de la empresa o usuario de la organización. Dicho esto, las interfaces usuarios del sistema desarrollado está pensado para un usuario operativo, donde la parafernalia gráfica que habitualmente es usada en sitios de marketing generalmente no aplica en una interfaz para un usuario operativo, donde debe tomar decisiones con el menos rango a error posible, todo lo que sea innecesario que no apunte a la interacción eficiente con el sistema de software se elimina en el diseño de las

interfaces usuarios de la aplicación desarrollada en este trabajo de título. Entonces, la **CLARIDAD** (evitando las ambigüedades y dando prioridad a la simplicidad) , la **EFICIENCIA** (realizar la tarea en el menor tiempo posible), la **CONSISTENCIA** (permitir que el usuario reconozca patrones y lo apoye al aprendizaje), la **SIMPLICIDAD** (concisa, evitar la sobre carga de elementos que distraen), la **GESTIÓN DE ERRORES** (el usuario puede saber en forma fácil que algo está haciendo mal y puede volver atrás), la **FAMILIARIDAD** (que con el tiempo la interacción con el sistema sea intuitiva) son características que se privilegiaron en el momento del diseño de las interfaces para que el usuario ponga toda la atención en la tarea del proceso que está ejecutando y no en la herramienta.

5. Validación de la Solución

La ONEMI, Oficina Nacional de Emergencias, frente a un terremoto debe coordinar a los diferentes actores sociales tales como Bomberos, Autoridades, Servicios de Urgencias, Organizaciones Sociales o Vecinales, entre otros. Esto para hacer frente a la Crisis o Catástrofes que se deben enfrentar frente a un terremoto, donde muchos de los servicios básicos dejan de funcionar, como agua potable, electricidad, carreteras, etc.
La Catástrofe obliga a tomar decisiones en función de eventos o ubicaciones geográficas, por ejemplo, un puente caído, un hospital siniestrado, zonas inundadas, poblaciones sin agua potable, coordinación de albergues y muchos otros. En cada uno de estos puntos geográficos se deben ejecutar tareas y coordinar personas que deben asumir diferentes roles, estas tareas dependen de uno más puntos geográficos donde deben ser ejecutadas en contexto de su ubicación. Por ejemplo abastecer de agua potable es una tarea que no se realiza de la misma forma en un cerro de Valparaíso que en el plano de Viña del Mar. Para cada una de estas macro-tareas de abastecer agua potable a la población afectada por la Catástrofe se debe generar un

proceso ad hoc en función de los puntos geográficos donde se va a realizar la acción de abastecimiento.

Estos procesos ad hoc deben crearse para la mayoría de las Actividades Globales del Proceso de emergencia de la Onemi. Entre ellos están: procesos ad hoc para asignar los cupos en los albergues, procesos ad hoc para armar hospitales de campaña y para la atención de la población afectada, etc. Todas estas tareas que debe enfrentar la Onemi, exigen procesos ad hoc georreferenciados, donde la ubicación geográfica de donde se ejecutará la tarea entrega por si misma mucha información que debe ser considerada para la realización de la tarea propiamente tal. La Coordinación de estos procesos georreferenciados es de gran importancia, dado que no podemos entregar información errada a las personas que necesitan urgente atención. Para esto, aplicaciones como Whatsapp o Twitter son una necesidad, considerando que las redes de celulares generalmente siguen operando, no así las redes cableadas o eléctricas, estas plataformas de microbloggins juegan un gran papel en complementar la eficacia de estos procesos ad hoc georreferenciados para este tipo de situaciones de catástrofe natural, donde saber dónde actuar geográficamente puede llegar a ser vital.
La aplicación, objeto de este trabajo de título es el punto de partida para motivar y mostrar que los procesos ad hoc georreferenciados y coordinados con plataformas de microblogging pueden ser de mucha utilidad para este tipo de oficinas gubernamentales.

Para validar la solución, se mostrará cómo W-adhoc apoyaría un proceso desarrollado en un escenario como en el que se describe anteriormente. El escenario escogido fue el terremoto del 27 de Febrero del 2010 (27/F) en Talcahuano y Concepción.

5.1. Caso de uso: Entrega de agua potable

Un ejemplo emblemático se vivió entre la sexta y la octava región de nuestro pais, luego del terremoto del 27/F donde además del terremoto hubo que enfrentar un maremoto. Esta situación no era esperada por el servicio de agua potable de la zona, Essbio, quienes si bien contaban con planes de emergencia exigidos por sus reguladores, no contaban con uno para enfrentar la situación que provoco la catástrofe causada por el terremoto y el maremoto del 27/F. Especialmente, en la cuidad de Talcahuano el maremoto devasto las redes de distribución de agua potable y alcantarillado, pero solo cuando se dio el agua se puso conocer los daños en esas redes. La empresa reparaba las roturas en la medida que las iba encontrando, por lo que no sabían con certeza el tiempo que demoraría la reparación total para volver a retomar el suministro de agua potable a la normalidad. Para ello se inició una labor de mitigación, entregando estanques en las poblaciones y repartiendo agua en los camiones aljibe, donde la labor y apoyo del municipio y de bomberos fue crucial. Para conocer las necesidades de la gente, Essbio formo alianza con la Unidades vecinales de la zona. Para comunicarse en el proceso de abastecimiento Essbio declaró haber utilizado Twitter, sitio web y call center externos. Así, a través de este medio, la gente daba a conocer los sectores sin servicio de agua potable a la compañía.

Essbio reconoce haber tenido dificultades en el traslado del agua al Cerro Chepe y también en superar un problema en la bajada del Estanque de las Higueras. Se habíamos llegado con agua a buena parte de los cerros de Talcahuano, pero no a Las Higueras, Denavi Sur y Las Salinas, donde se demoró bastante más en abastecer, porque tanto la gran red como la secundaria estaban fragmentadas.

Dado el escenario descrito, se puede identificar funcionalidades de la solución propuesta que permiten y podrían haber apoyado la labor que se ejecutó por Essbio, junto con Bomberos, Municipios y Organizaciones vecinales.

Se sabe que el agua distribuida se obtenía desde La Mochita, para ser trasladada a los demás centros de Essbio. Dado esto es posible definir un flujo en donde los actores Essbio, Bomberos, Municipalidades y Junta vecinal y personas puedan crear un proceso ad-hoc utilizando valiosamente la geolocalización.

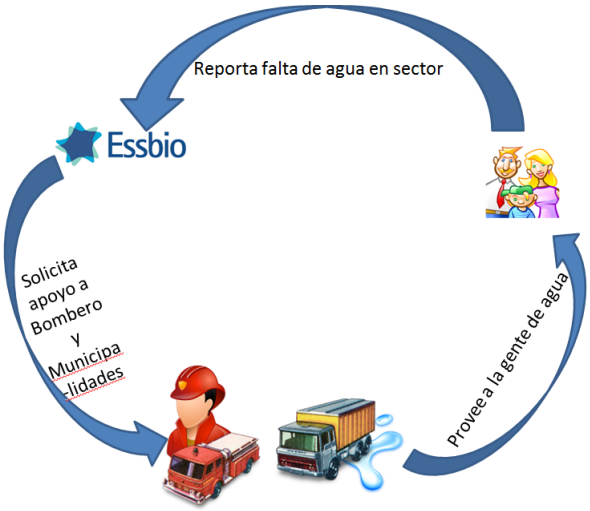


Figura 35: Escenario propuesto

Inicialmente los vecinos indican a Essbio donde se encuentran las zonas sin el recurso disponible.

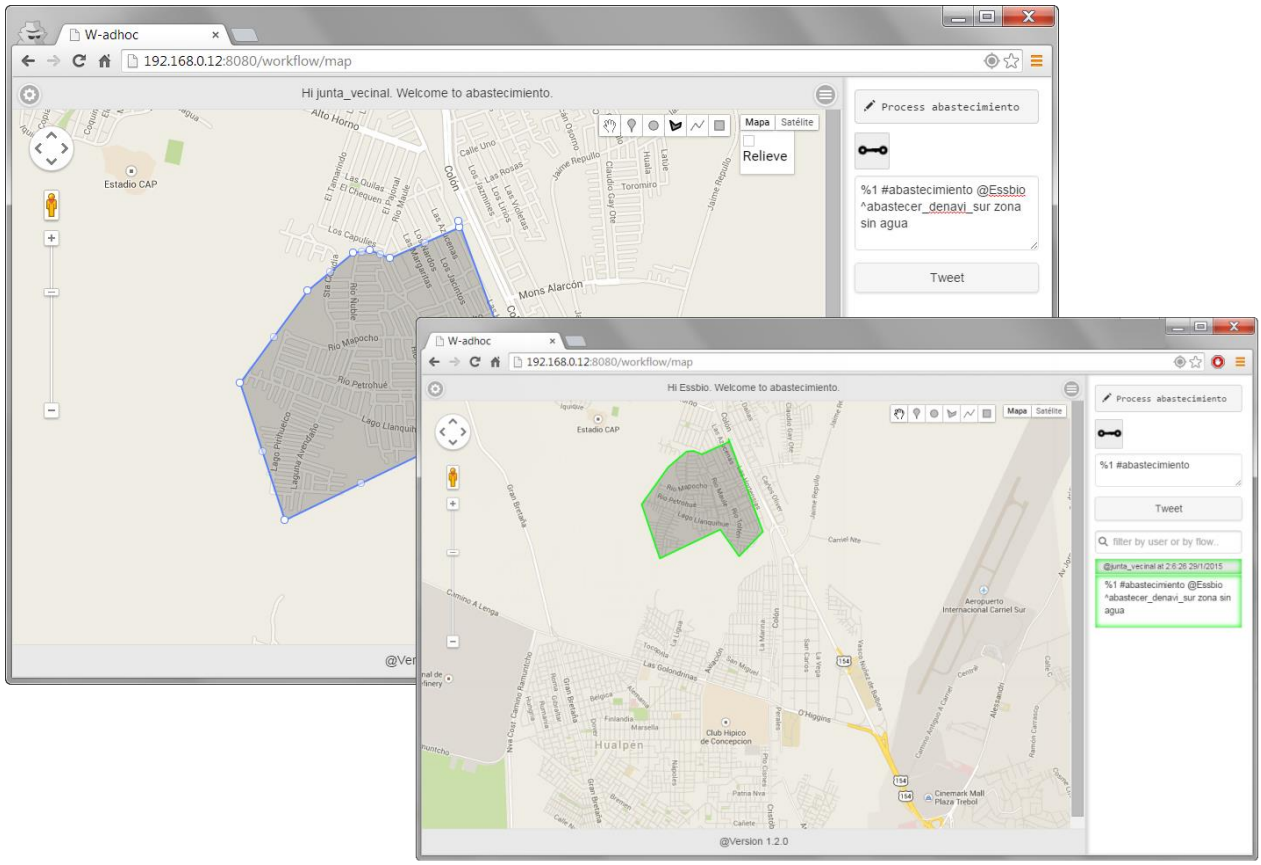


Figura 36: Junta vecinal informa falta de agua a en sector Denavir Sur a Essbio

Seguidamente Essbio pide apoyo a Bomberos, indicándole la tarea a cumplir. La tarea es: Que bombero se dirija a la Planta La Mochita para cargar sus camiones con agua.

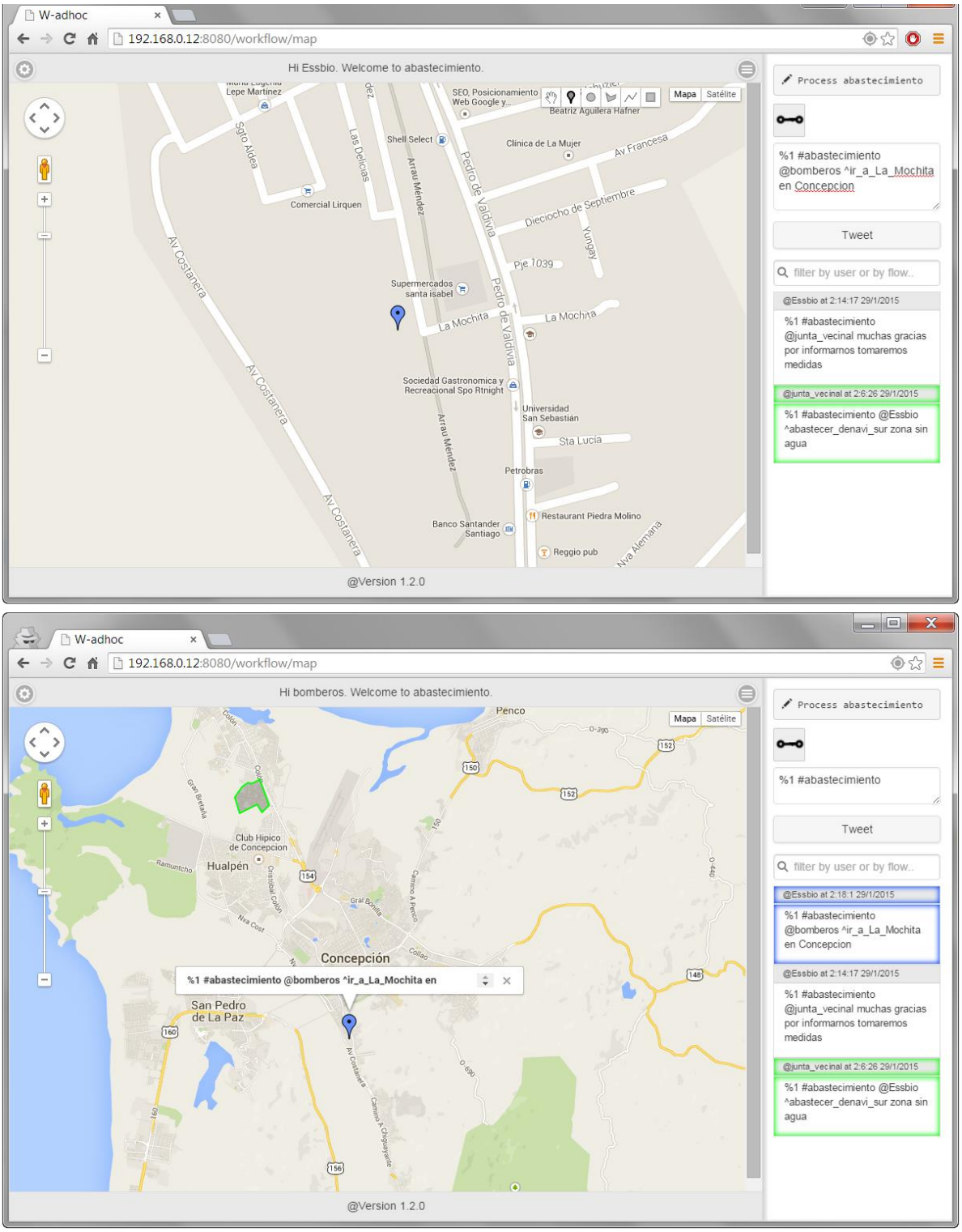


Figura 37: Essbio pide a bomberos dirigirse a La Mochita

Paralelamente Essbio pide apoyo a la Municipalidad: Que se dirija con sus camiones ya llenos al sector donde se encuentran los damnificados.

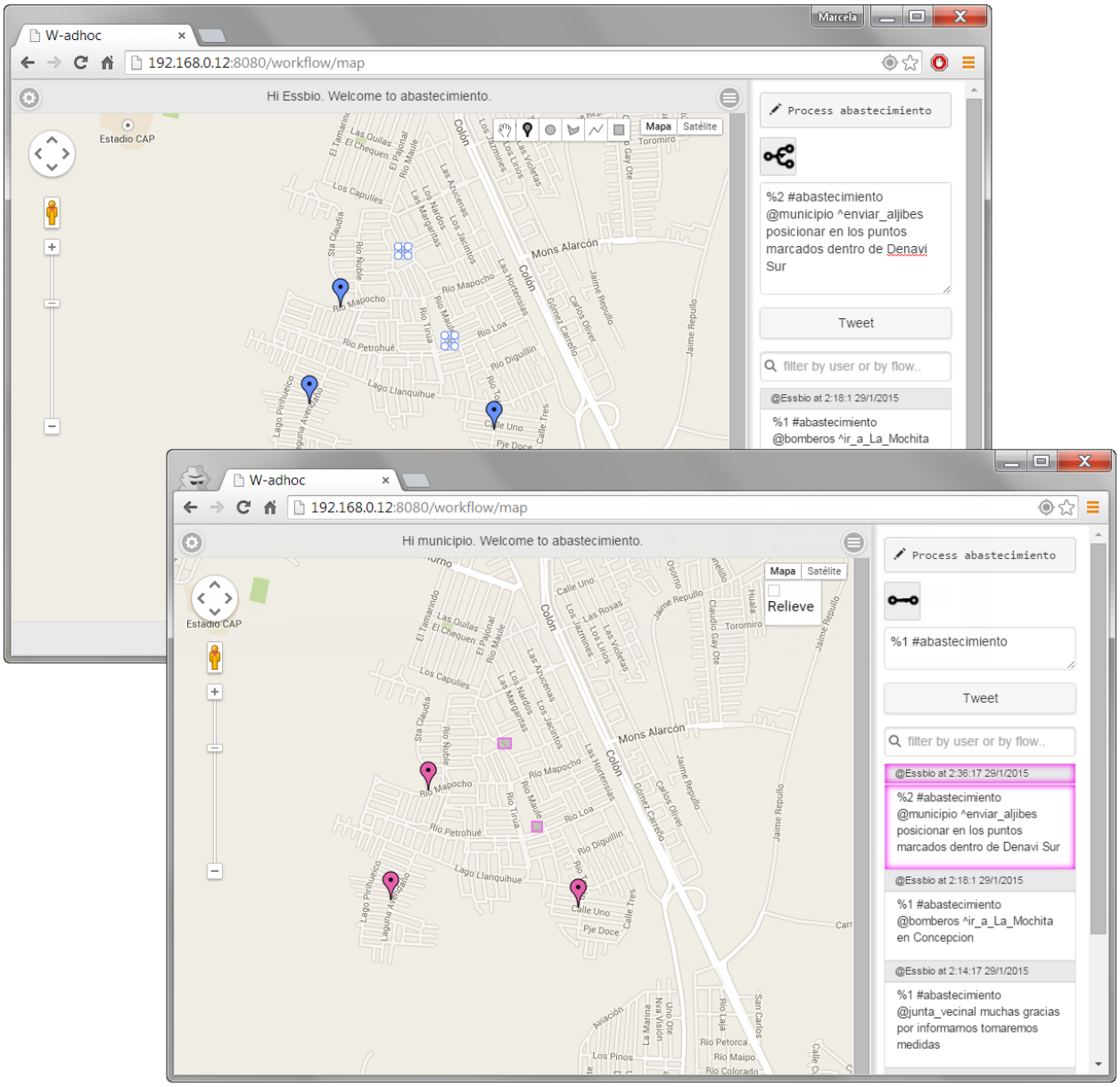


Figura 38: Creación de tarea en paralelo, Essbio pide a Municipio Trasladar Camiones ya cargados

Estratégicamente, Essbio les indica donde deben posiciones los camiones.

Así, se puede llevar a cabo un flujo, en donde las juntas vecinales sean las encargadas de identificar lugares sin acceso al recurso para luego informarle a Essbio. Se puede decir que el proceso de abastecer de agua, está compuesto por varios sub procesos en donde la tarea es la misma: proveer agua, pero que se realiza en distintos lugares cada vez de distinta manera, ya que las rutas varían y las dificultades también.

6. CONCLUSIONES

Más allá de la aplicación misma, el presente trabajo de memoria ha permitido comprender mejor el proceso de ingeniería, donde la complejidad generalmente está en la comprensión del problema a resolver, sin esta etapa superada, cualquier solución que demos es inadecuada o mala. La ingeniería nos enseña a diseñar algo para resolver un problema a alguien (persona u organización), este trabajo ha estado centrado en resolver el problema a las organizaciones que necesitan enfrentar situaciones que exigen salirse de los procesos de negocios ya definidos y consolidados, son situaciones que en cualquier parte de sus procesos de negocios normales necesitan crear un proceso ad hoc para resolver esa situación específica y con una variable espacial o geográfica no presente en el proceso de negocio formal de la organización, como ya se ha visto durante el presente trabajo. Comprendido este problema la solución planteada, este trabajo fue desarrollar una herramienta de software que permita apoyar estos procesos o flujos de trabajo ad hoc para resolver esa situación particular muchas veces única de la empresa, donde la variable de ubicación geográfica del lugar en el cual será ejecutada la tarea es decisiva para el éxito de ésta. El motivo es que esta variable entrega información que no es fácil que esté en los sistemas de procesamiento de información del negocio, por ello se incorpora un nuevo componente que es los sistemas de información geográfica. Con este nuevo componente, aparece un tema a resolver que está ya resuelto en forma fácil en los sistemas BPM tradicionales. Ésta es la coordinación de los diferentes actores y roles, que son parte del modelo. Sin embargo, en estos procesos ad hoc, la coordinación de los actores que ejecutan las tareas, dado que éstas son ejecutadas tomando variables de la ubicación geográfica donde será realizada, hace necesario un sistema que entregue información de coordinación a estos actores. Para lograr esto se tomó la decisión de aplicar lo ya existente en el mercado, el concepto de microblogging. A partir de

estos componentes se diseñó y se construyó a modo de prueba de concepto la aplicación que se describe en este trabajo, la cual es una semilla que, se espera, motive a otros a desarrollar una herramienta que permita dar solución completa a este problema de procesos ad hoc con necesidad que sus workflows sean ejecutado en una ruta geográfica cambiante o volátil. La gran conclusión es que no es fácil construir una solución de software para un problema que no esté totalmente comprendido y más difícil aún si este problema lo tiene un cliente que necesita que la solución sea lo antes posible.

6.1. Posibles trabajos futuros

El trabajo que quiere mostrar el presente informe de memoria, como bien se menciona en el capítulo anterior, solo describe la primera etapa de lo que podría ser algo mucho más grande y explotable para incrementar en funcionalidades.

Una de ellas podría ser implementar alertas cuando se agrega un usuario a un proceso y/o cuando se ha asignado una tarea un usuario, por ejemplo. Tal vez explotar la información de los procesos, viéndolos por tema y extrayendo datos a una base de datos ETL[2] con el fin de encontrar información relevante de un proceso, como bien se decía en la sección 3.1, para luego visualizar e identificar el cambio del proceso con el fin de economizar recursos en las empresas.

Podría incluso poder subirse información audiovisual a los procesos, tratando de utilizar más la base de datos interna de **WORKFLOW AD-HOC** y liberar un poco la carga de la base de datos de **COUPLINGSERVER**, entre muchas cosas más.

[2] Extract, Transform and Load: Es el proceso que permite a las organizaciones mover datos desde múltiples fuentes, reformatearlos y limpiarlos, y cargarlos en otra base de datos, data mart, o data warehouse para analizar, o en otro sistema operacional para apoyar un proceso de negocio.

7. ANEXO A

7.1. Apéndice 1: Estructura de Datos

```java
package cl.uchile.workflow.persistance.model;

import java.io.Serializable;

@Entity
@Table(name="process", schema="public")
public class Process implements Serializable {
    /**
     *
     */
    private static final long serialVersionUID = -7788619177798333712L;
    @Id
    @GeneratedValue(strategy = GenerationType.TABLE)
    private Long id;
    private String processId;
    private String name;
    private String owner;

    private Integer estado;

    private Date fecha;

    private Date fechaFin;
    @ManyToMany(fetch = FetchType.EAGER)
    private List<User> users=new ArrayList<User>();

    public Process(String name, String owner) {
        super();
        this.name=name;
        this.owner = owner;
        this.processId = name.trim().replaceAll(" ", "");
        this.fecha = new Date();
    }

    public Process() {
        super();
    }

    public String getProcessId() {
        return processId;
    }

    public void setProcessId(String processId) {
        this.processId = processId;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public List<User> getUsers() {
        return users;
    }

    public void setUsers(List<User> users) {
        this.users = users;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    @OneToOne(fetch=FetchType.LAZY, mappedBy = "process")
    public String getOwner() {
        return owner;
    }

    public void setOwner(String owner) {
        this.owner = owner;
    }

    @Temporal(TemporalType.TIMESTAMP)
    @Column(name="fecha", nullable= false)
    public Date getFecha() {
        return fecha;
    }

    public void setFecha(Date fecha) {
        this.fecha = fecha;
    }

    @Temporal(TemporalType.TIMESTAMP)
    @Column(name="fecha_fin")
    public Date getFechaFin() {
        return fechaFin;
    }

    public void setFechaFin(Date fechaFin) {
        this.fechaFin = fechaFin;
    }

    public Integer getEstado() {
        return estado;
    }

    public void setEstado(Integer estado) {
        this.estado = estado;
    }

}
```

```java
package cl.uchile.workflow.persistance.model;

import java.io.Serializable;

@Entity
@Table(name="user", schema="public")
public class User implements Serializable {

    private static final long serialVersionUID = -2170165293671855171L;
    @Id
    @GeneratedValue(strategy = GenerationType.TABLE)
    private Long id;
    @Column(name="username", unique=true)
    private String user;
    private String password;
    @ManyToMany(cascade=CascadeType.ALL, fetch= FetchType.LAZY)
    private List<Process> processes=new ArrayList<Process>();

    public User(String user, String password) {
        super();
        this.user = user;
        this.password = password;
    }

    public User() {
        super();
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getUser() {
        return user;
    }

    public void setUser(String user) {
        this.user = user;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public List<Process> getProcesses() {
        return processes;
    }

    public void setProcesses(List<Process> processes) {
        this.processes = processes;
    }
}
```

Figura 39: Estructura de Dato de Workflow ad-hoc

8. BIBLIOGRAFÍA

1. *An Ad-Hoc Workflow System Architecture Based on Mobile Agents and Rule-Based Processing.* **Meng, J., Helal, S. y SU, S.** Gainesville : s.n., 2000. Computer and Information Science and Engineering, University of Florida.

2. *Creating Mobile ad hoc Workflow with Twitter.* **Treiber, M., y otros, y otros.** TU Vienna, Wien : s.n., 2012. Distributed Systems Group.

3. *Tweetflows: flexible workflows woth twitter.* **Trieber, M., y otros, y otros.** 2011. Proceedings of the 3rd International Workshop on Principles of Engineering Service-Oriented Systems, ACM.

4. *Managing Escalation of Collaboration Processes in Crisis Mitigation Situations.* **Georgakopoulos, D.** 2000. 16th International Conference on Data Engineering.

5. *Workflow Patterns.* **Van Der Aals, W., Hofstede, A. y Kiepuszewski, B.** 2003. Distributed and Parallel Databases.

6. *Desarrollo de herramienta colaborativa para el levantamiento de procesos BPMN en dispositivos móviles.* **Aguirre, D.** Santiago : s.n., 2012. Departamento de Ciencias de la Computación, Universidad de Chile.

7. *Ad hoc Iteration and Re-execution of Activities in Workflows.* **Sonntag, M. y Karastoyanova, D.** 2012. Institute of Architecture of Application Systems, University of Stuttgard.

8. *Caramba—A Process-Aware Collaboration System.* **Dustdar, S.** The Netherlands : s.n., 2004. Kluwer Academic Publishers.

9. *Mining of ad-hoc business processes with TeamLog.* **Dustdar, S., Hoffman, T. y Van der Aalst, W.** 2005.

10. **Lofi, C. y Maarry, K. El.** *Design Patterns for Hybrid Algorithmic-Crowdsourcing Workflows.* 2014.

11. **Pflanzl, N. y Vossen, G.** *Human-Oriented Challenges of Social BPM: An Overview.* 2013.

12. —. *Challenges of Social Business Process Management.* Waikoloa, HI : s.n., 2014.

13. *Change patterns and change support features – Enhancing flexibility in process-aware information systems. .* **Weber, B., Reinchert, M. y Rinderle, S.** 2008. Data & Knowledge Engineering.