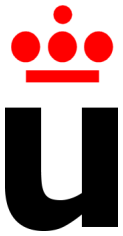

UNIVERSIDAD
REY JUAN CARLOS

Máster Universitario en Software Libre

Curso Académico 2012/2013

Proyecto Fin de Máster

Participación en el desarrollo del nuevo gadget de Troco

Autor: Esteban Carreras Genis

Tutor: Dr. Gregorio Robles

Agradecimientos

A mis profesores por todo lo que me han enseñado sobre este mundo del Software Libre.

A mi tutor por su constante e inestimable ayuda.

A mis compañeros por su ayuda durante todo el Máster y por sus ánimos para finalizarlo.

A Samer Hassan y a Vicente J. Ruiz por toda la ayuda que me han brindando y por tener una paciencia infinita.

A mis padres por su eterno apoyo y por creer en mí siempre.

A mis amigos por estar siempre ahí.

Y por supuesto a mi mujer que siempre me ayuda y apoya, incluso cuando bajo los brazos.

Índice general

Resumen

Troco es un gadget FLOSS[1] para la plataforma Kune [1] creado, de forma experimental y con un funcionamiento básico, en 2009 por miembros de la asociación Comunes [2]. Actualmente desde esta asociación surge la necesidad de evolucionar este proyecto, dotándolo de mayor funcionalidad y mejores interfaces con el objetivo de proporcionar un sistema completo de intercambio de bienes y servicios completamente descentralizado para los distintos usuarios y proyectos de la plataforma Kune, permitiendo incluso personalizar en cierta medida el gadget para su utilización a nivel particular.

La funcionalidad de la primera versión de la herramienta de Troco [3] es permitir el intercambio básico de bienes y servicios entre dos usuarios de Kune, permitiendo establecer el valor que se desea intercambiar y la fecha tope para realizar dicho intercambio. Posibilitando añadir a un receptor de dicho intercambio o bien permitiendo dejar el Troco abierto, o lo que es lo mismo, pendiente para que se agrege algún receptor del mismo y especifique el valor de retorno, que puede ser el mismo que se ofrece o algo completamente distinto. Una vez establecidos los valores por ambas partes, y si los usuarios aceptan ese intercambio, éstos deben acordar, mediante algún tipo de comunicación, la forma de realizar el intercambio real y que así se complete el proceso.

A continuación en la figura 0.1, se puede apreciar el funcionamiento básico de la primera versión del gadget de Troco:

[1] Free/Libre and Open Source Software

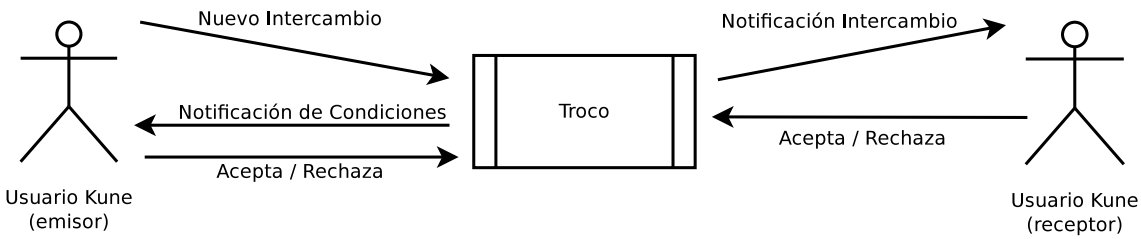

Figura 0.1.: Funcionamiento de la primera versión del gadget de Troco

La evolución de Troco pretende aumentar el funcionamiento de la primera versión del gadget, ampliando las funcionalidades y renovando por completo esta herramienta, para permitir mejorar la experiencia de usuario y le proporcionen las características necesarias para poder realizar el tipo de intercambio más adecuado a sus necesidades, con mayor nivel de detalle y de una manera más flexible y sencillas, entre ellas se incluyen:

- Un asistente (wizard) que orientará a los usuarios nuevos a la hora de seleccionar el tipo de intercambio más adecuado para sus necesidades.

- Un tutorial que proporcionará la ayuda en línea necesaria para el usuario.

- El intercambio directo que permitirá a los usuarios experimentados utilizar el sistema de una forma rápida, sencilla y eficiente.

Troco, como se ha comentado anteriormente, es un proyecto de software libre, y por tanto respeta las cuatro libertades básicas establecidas por la FSF[2] [4], como son la libertad de uso, la libertad de copia, la libertad de modificación y la libertad de distribución. Además se ha desarrollado en el marco de una comunidad de software libre llamada Comunes, lo que proporciona el entorno perfecto para poder trabajar de forma colaborativa, en diferentes proyectos de distintas índoles, ayudar en distintas áreas y por supuesto proponer y mejorar o cambiar cualquier programa que actualmente existe en dicha comunidad.

Para la defensa de este proyecto vamos a hablar de tres aspectos:

[2]Free Software Foundation

- **Redefinición de la funcionalidad del gadget actual de Troco**, definiendo de nuevo todas las características que debe proporcionar a los usuarios, así como dotarlo de una nueva interfaz gráfica que ayude a mejorar la experiencia de usabilidad y proporcionar la ayuda necesaria para orientar a todos los usuarios a los diferentes tipos de intercambios que se pueden realizar desde esta herramienta.

- **Participación en el desarrollo del nuevo gadget**, ayudando en el desarrollo del código fuente del nuevo gadget, donde he tenido la oportunidad de ampliar conocimientos sobre el framework de desarrollo GWT[3] [5] y de trabajar de forma colaborativa con otros desarrolladores de Comunes en un proyecto FLOSS real.

- **Impresiones y seguimiento de mi colaboración en un proyecto FLOSS en conjunto con la comunidad**, donde se explicarán cada una de las tareas realizadas desde el inicio del proyecto, los problemas surgidos durante el desarrollo, el grado de avance actual y los futuros pasos que se establecerán para finalizar el proyecto completamente.

[3]Google Web Toolkit

1. Introducción

En este capítulo se explicarán los orígenes de la primera versión del gadget de Troco, exponiendo el sistema en el cual se basó su idea, así como las motivaciones que llevaron a su desarrollo. También se describirán brevemente las principales razones por las cuales se realizará una evolución y en qué consistirán estos cambios, mencionando herramientas que tienen propósitos similares y las herramientas técnicas utilizadas a lo largo del proyecto.

1.1. Historia

La primera versión de Troco surgió en 2009 como un proyecto para proporcionar un sistema alternativo y descentralizado de intercambio de bienes para la plataforma Google Wave (actualmente llamada Apache Wave [6]).

Aunque el concepto de Troco en definitiva, como es lógico, se basa principalmente en el sistema tradicional de trueque, que fue uno de los primeros sistemas de comercio utilizados por la humanidad (aproximadamente comenzó a utilizarse hace 10.000 años), realmente está fuertemente influenciado por el sistema japonés WAT [7] y su plataforma digital i-WAT [8].

El sistema WAT surgió en el año 2000 y fue creado por Eiichi Morino [10], que era el presidente de la Sociedad de Investigación Gesell Japón [11], para solucionar las quejas y los problemas surgidos en la gestión de las redes de intercambio de energía limpia producida por turbinas

de viento y paneles solares que estaban establecidas en esa época para comerciar entre diferentes regiones, separadas por grandes distancias, y donde se establecía un valor de venta distinto dependiendo de la zona.

El principal objetivo de este sistema era proporcionar un método justo para gestionar y organizar este intercambio de una manera eficiente, así como minimizar los costes asociados, estableciendo una alternativa a los distintos y desiguales sistemas de intercambio económico que existían en las diversas regiones de Japón.

La manera de solucionar estas dificultades, fue crear una red de intercambio descentralizada basada en la confianza y la solidaridad para generar valor, donde se establece una moneda ficticia, llamada WAT-ticket [9], que se crea sobre una base de igual a igual. Evitando así la necesidad de establecer una administración centralizada para gestionar estos intercambios y reduciendo los costes asociados a dicha gestión, ya que cualquiera que acepte WAT-tickets se convierte en miembro del grupo y puede comerciar con el resto del círculo de confianza.

La forma de calcular el valor de los WAT-tickets es muy peculiar, ya que se basa en comparar 1 WAT con 1 kWh de corriente eléctrica generada de forma cooperativa por un ciudadano en una unidad de tiempo de trabajo básico y una cantidad aproximada en Yenes. De esta forma, actualmente 1 kWh se corresponde aproximadamente con:

1 kWh = 6 minutos de trabajo simple + de 75 a 100 ¥

En verano del 2000 se comenzó a utilizar este sistema distribuyendo WAT-tickets y formando la fundación WAT-friends [9], que contaba inicialmente con participantes que provenían de diferentes regiones y que ya realizaban intercambios. Al cabo de un mes el sistema WAT comenzó a extenderse entre muchas otras regiones que no habían comerciado entre sí anteriormente debido a los problemas que tenía el sistema anterior.

Posteriormente surgió el software i-WAT, que es una versión digitalizada de la base del sistema de intercambio WAT, que se ha mencionado anteriormente. Además es una herramienta

FLOSS y se distribuye bajo la licencia GNU GPL V1[1] [12].

Esta herramienta es un sistema económico complementario, cooperativo, distribuido y autónomo que permite a las comunidades que lo utilizan establecer de una forma sencilla relaciones de intercambio mediante la construcción de una red de confianza y la creación de monedas que se ajusten a los objetivos y necesidades específicas de cada comunidad.

1.2. Motivación

La motivación inicial de Troco era proveer un sistema de intercambio de bienes entre los usuarios de la plataforma Kune, de forma que pudieran cambiar servicios y/o bienes materiales de cualquier índole, sin depender de un intercambio económico.

La nueva versión del gadget de Troco viene motivada por la necesidad de ampliar el funcionamiento de esta primera versión del gadget, debido a que hasta ahora sólo se permitía un intercambio básico de bienes entre dos personas, proveyendo a los usuarios de Kune de una plataforma para poder realizar distintos tipos de intercambios de una manera sencilla, segura y práctica, y proporcionando de esta forma una red de colaboradores amplia donde poder intercambiar diferentes tipos de bienes y servicios basados en la confianza y la reputación de los colaboradores. Para conseguir este objetivo además es necesario redefinir y mejorar la interfaz de usuario de forma que proporcione a los usuario una herramienta sencilla, actualizada y más atractiva.

Por tanto esta evolución pretende dar cabida a un sistema descentralizado de intercambio de bienes con mayor funcionalidad, más características y escalable, que proporcionará a los usuarios una red de intercambio distribuida a través de la plataforma Kune, creando un tejido comercial distribuido e independiente del comercio habitual. Así como proporcionar un *look and feel* mejorado y más intuitivo para los usuarios.

[1] GNU General Public License

1.3. Precedentes

En la actualidad ya existe y se encuentra activo un gadget experimental de Troco para la plataforma Kune que proporciona la funcionalidad básica de intercambio entre dos participantes, permitiendo definir el valor que desea intercambiar el emisor del Troco, el valor que se compromete a devolver el receptor del mismo y la fecha cuando termina dicho intercambio, así como el estado actual en el cual se encuentra el intercambio, como se puede apreciar en la figura 1.1:

Figura 1.1.: Gadget actual de Troco

En los últimos años han proliferado los sistemas de intercambio de bienes online y de trueque de gran cantidad de productos y servicios, motivados muy posiblemente por la crisis económica en la que estamos inmersos y que son posibles debido a las facilidades que proporciona Internet

para comunicarse con gran cantidad de personas de diferentes lugares del mundo.

Como veremos en detalle en el capítulo 3, existen diversas plataformas que posibilitan la realización de intercambios de diferentes bienes y servicios, de una forma libre entre diferentes personas. A continuación se expone una lista con algunas de ellas:

- Trocobuy [13]: para el intercambio de productos y servicios entre empresarios.
- Truequeweb [14]: permite el intercambio de todo tipo de bienes y servicios entre particulares.
- Bookmooch [15]: para el intercambio entre particulares de libros.
- Titletrader [16]: para el intercambio de libros, DVDs[2] y CDs[3] entre particulares.
- Etruekko [17]: todo tipo de intercambios entre particulares.
- Cadenadecambios [18]: intercambio entre al menos tres usuarios de todo tipo de bienes.
- Polyglot [19]: comunidad para el intercambio de habilidades lingüísticas entre los usuarios.
- Home for Home [20]: para el intercambio de casas de forma temporal.

1.4. Elementos Funcionales

La nueva versión del gadget de Troco está implementada en el lenguaje de programación Java [25], en conjunto con otras plataformas y herramientas como son:

[2]Digital Versatile Disc
[3]Compact Disc

- Plataforma Kune, donde se desplegará y utilizará el nuevo gadget de Troco.

- El framework de desarrollo Google Web Toolkit (GWT), que define la arquitectura del nuevo gadget de Troco y nos proporciona las herramientas para su desarrollo y testing.

- La base de datos MySQL [21], para almacenar los datos de todos los intercambios realizados desde la herramienta.

- El repositorio de control de versiones Git [22], para el seguimiento de cada cambio realizado por los desarrolladores.

- El modelo de MVP[4] [23] de gwt-pectin [24], utilizado para la validación y modelado de formularios GWT y utilizado para la definición y validación de las interfaces del nuevo gadget.

Estos elementos son totalmente independientes entre sí, tienen funcionalidades diferentes y pueden ser utilizados conjuntamente o no. Hemos hecho uso de estos proyectos software basándonos en distintos aspectos:

- La integración con la plataforma Kune.

- La elección de un modo de persistencia de datos.

- La migración a un repositorio de control de fuentes distribuido.

- La elección de un sistema para la validación de formulario utilizado en todos los interfaces del nuevo gadget.

En el capítulo 2 veremos en profundidad cada uno de estos componentes que hacen posible la implementación y el funcionamiento de gadget de Troco.

[4]Model-View-Presenter

2. Aspectos Técnicos

En este apartado se expondrán en detalle las diferentes tecnologías utilizadas para el desarrollo de la nueva versión del gadget Troco, desde el lenguaje de programación en el que se está implementada la aplicación como los elementos funcionales utilizados en el transcurso del desarrollo, mencionados anteriormente en la sección 1.4.

2.1. Java

El lenguaje de programación que se está utilizando en el desarrollo de Troco es Java, ya que es el lenguaje de programación utilizado en la plataforma Kune, para la cual se está desarrollando la evolución de este gadget.

Java es un lenguaje de programación extremadamente popular y ampliamente utilizado, segundo lenguaje de programación más utilizado por profesionales en en el ranking TIBOE [26]. Como es ampliamente conocido, Java es un lenguaje de programación orientado a objetos creado por Sun Microsystems [27] (adquirida por Oracle [28] en el año 2009) a principio de los años 90.

Uno de los propósitos principales de Java era mejorar y simplificar la forma de desarrollar que se venía utilizando en los lenguajes de programación de esa época. Dispone de una gran similitud con los lenguajes C y C++ [29], ya que fueron los lenguajes en los que en gran medida

se basaron, sin embargo proporciona una manera más sencilla a la hora de implementar, como por ejemplo abstraer al desarrollador de manejar punteros de acceso a memoria, que resultan especialmente complejos y son uno de los focos más habituales de errores en estos lenguajes, o bien la acciones asociadas a la liberación de memoria de objetos o estructuras de datos que ya no se están utilizando, para lo cual Java proporciona su recolector de basura que realiza esta acción de forma automática, asegurando así el rendimiento de la aplicación.

Otro de sus principales objetivos es ser un lenguaje orientado a la multiplataforma, es decir, su propósito surgió para que una aplicación creada sobre un sistema operativo no tuviera llamadas nativas al mismo y fuera capaz de funcionar para una plataforma distinta sobre la que se había desarrollado. Esto se consigue gracias a que Java es compilado en bytecode, que es un código intermedio más abstracto que el código máquina y el cuál es interpretado en una JVM[1]. Este hecho es el que hace a Java multiplataforma, ya que existen máquinas virtuales para casi cualquier sistema operativo, con lo que el código generado en bytecode puede ser ejecutado en cualquier máquina virtual de java.

Como se ha mencionado anteriormente Java es un lenguaje de programación orientado a objetos (OOP[2]), lo que hace que disponga de todas las características asociadas a los lenguajes que utilizan este paradigma de programación, que son:

- Abstracción.
- Encapsulamiento.
- Herencia.
- Polimorfismo.

[1] Java Virtual Machine
[2] Oriented-Object Programing

2.2. Google Web Toolkit(GWT)

Para el desarrollo de Troco hemos utilizado el framework de desarrollo FLOSS GWT. Esta librería fue creada por la empresa Google [30] en el año 2006 y se encuentra licenciada bajo la licencia Apache 2.0 [31].

Este framework proporciona las librerías y herramientas necesarias para poder implementar complejas aplicaciones Web de una forma sencilla y rápida, abstrayendo a los desarrolladores de la complejidad de algunos de los aspectos de la tecnología AJAX[3]. En resumidas cuentas, el propósito general de GWT es traducir el código Java creado desde un IDE[4] a código HTML[5] + JavaScript, permitiendo al desarrollador utilizar todas las ventajas y características de Java así como la depuración de código para crear aplicaciones de alto rendimiento de tipo cliente servidor.

Las aplicaciones GWT se pueden ejecutar de dos formas:

- **Developer mode**, donde la aplicación se ejecuta como bytecode de Java en la JVM, permitiendo la depuración y realizar cambios en tiempo de ejecución.

- **Web mode**, donde la aplicación se ejecuta como código Javascript y HTML puro desplegado en un servidor Web.

2.2.1. Versiones

Desde su primera versión en el año 2006, Google ha liberado actualizaciones del framework, siguiendo exclusivamente una nomenclatura numérica, que han aumentado la funcionalidad de esta herramienta, hasta llegar en la actualidad a la versión 2.5.1.

[3] Asynchronous JavaScript And XML
[4] Integrated Development Environment
[5] HyperText Markup Language

A continuación, se expondrán brevemente las diferentes versiones [32] que existen de esta librería, así como los cambios más relevantes que han incluido:

- **1.0**

  Primera versión RC[6] del SDK[7] se lanzó en mayo del 2006 y fue anunciada por Google en la conferencia JavaOne [33], que es una conferencia anual para discutir las tecnologías Java. Esta versión contenía las funcionalidades básicas del framework y resolvía una serie de incidencias con el navegador IE7[8] y con los componentes principales.

- **1.1**

  Esta versión del SDK fue lanzada tres meses después, en agosto de 2006, e incluía bastantes características nuevas como Internacionalización, nuevos widgets y clases para JSON[9] y XMLExtensible Markup Language, cambios importantes en las características de GWT, como los módulos de JUnit [34] y en el módulo de código. Además resolvía algunos errores menores.

- **1.2**

  Esta versión de la librería se lanzó en noviembre de 2006 e introducía nuevas funcionalidades para el soporte de OS X[10], mejoras en la velocidad modo host, un nuevo módulo de peticiones HTTP[11] y nuevos widgets.

- **1.3**

  Esta versión fue liberada en febrero de 2007 y fue la primera versión realmente FLOSS de esta librería. No añadía nuevas funcionalidades, sino que todos los cambios se centraron en disponer de la totalidad del código FLOSS.

  Además en esta versión se publicaron el sistema de bug tracking [35] y el repositorio de código en Subversion [36].

[6] Release Candidate
[7] Software Development Kit
[8] Internet Explorer versión 7
[9] JavaScript Object Notation
[10] Macintosh Operative System X
[11] Hypertext Transfer Protocol

- **1.4**

  En esta versión, que fue lanzada en agosto de 2007, se incluyeron por primera vez los cambios aportados por desarrolladores libres externos. Se corrigieron gran cantidad de incidencias y se realizaron cambios importantes en el API[12].

- **1.5**

  Esta versión fue liberada en agosto de 2008 y contenía gran cantidad de cambios, entre ellos, cambios en el compilador, en la librería de emulación JRE[13], en la interfaz de usuario, en los componentes RPC[14] y en la internacionalización.

- **1.6**

  Fue lanzada en abril de 2009 con soporte para war[15], compilación más rápida, nuevos widgets, mejoras en el manejo de eventos y resolución de gran cantidad de incidencias.

- **1.7**

  Esta versión se liberó en julio de 2009, donde se añadió explícitamente soporte para IE8[16], Firefox 3.5[17] y Safari 4[18], y se dejan resueltos algunos bugs críticos.

- **2.0**

  Esta nueva versión fue liberada en diciembre de 2009 y cuenta con grandes cambios para mejorar la productividad de los desarrollador, simplificación del desarrollo cross-browser y producción de aplicaciones Web más rápido.

- **2.1.0**

  Tras varias versiones menores que resolvían incidencias de la versión 2.0, en octubre de 2010 se liberó esta versión del SDK de GWT, que además de incluir la resolución de gran cantidad de bugs, disponía de nuevos widgets, inclusión del framework MVC[19],

[12] Application Programming Interface
[13] Java Runtime Environment
[14] Remote Procedure Call
[15] Web Application Archive
[16] Internet Explorer versión 8
[17] http://www.mozilla.org/es-ES/firefox/new/
[18] http://www.apple.com/es/safari/
[19] Model-View-Controller

trazas a nivel de servidor, logs e integración con otras librerías como SafeHTML [37] y SpringSource [38].

- **2.2.0**

  En esta versión, lanzada en febrero de 2011, se incluyó como parte del plugin para el IDE Eclipse [39] un editor de interfaz gráfica integrado (UI Designer [40]), además de soporte para HTML5 y mejoras en varios widgets.

- **2.3.0**

  Esta versión del SDK fue lanzada en mayo de 2011, donde se añade soporte para IE9[20], funcionalidad al plugin de Eclipse y soporte para el almacenamiento local de HTML5, además de la corrección de diferentes incidencias.

- **2.4.0**

  En septiembre de 2011 se liberó esta versión que contiene nuevas herramientas para Android [41], mejoras en RPC, soporte para el Apps Marketplace [42] y mejoras en el UI Designer.

- **2.5.0**

  Esta versión fue liberada en junio de 2011. Añade nuevas funcionalidades a la librería como mejoras en la optimización del compilador, mejoras en UIBinder [43] y en la validación, así como una nueva librería experimental, llamada Elemental, para el desarrollo Web.

- **2.5.1**

  Esta versión fue liberada en enero de 2013 y es la versión actual de GWT. Incluye gran cantidad de mejoras, como por ejemplo, mejoras de seguridad, mayor optimización del compilador, nuevas características del Developer Mode, mejoras en la emulación de JDK, nuevos widgets, mejoras en el UI Editor y ampliación de las opciones de internacionalización.

[20] Internet Explorer versión 9

2.2.2. Características

Las características principales del framework GWT son las siguientes:

- GWT dispone de una serie de componentes gráficos que son dinámicos y reutilizables, de forma que los desarrolladores pueden utilizar clases predefinidas para crear y usar gran cantidad de servicios y comportamientos de una forma óptima.
- RPC realmente sencillo.
- Administración del historial del navegador Web.
- Soporte a la depuración de Java.
- Control de diferentes características del navegador Web.
- Integración con la librería de testing JUnit.
- Internacionalización.
- Posibilidad de utilizar la interfaz nativa de Javascript (JSNI) para utilizar código Javascript dentro del código Java.
- Soporte para las APIs de Google.
- Es una librería FLOSS, licenciada bajo la licencia Apache 2.0.
- Orientación a Objetos, con todas las ventajas que supone.
- El código Javascript generado puede ser ofuscado para optimizar el rendimiento.

- Gran cantidad de bibliotecas y aplicaciones, creadas por Google y terceros que permiten ampliar las funcionalidades de GWT.

2.2.3. Arquitectura

GWT se compone de cuatro componentes principales que proporcionan a los desarrolladores las herramientas y librerías necesarias para poder crear aplicaciones Web de una forma sencilla y rápida aprovechando todas las ventajas del desarrollo orientado a objetos de Java, como se puede apreciar en la figura 2.1:

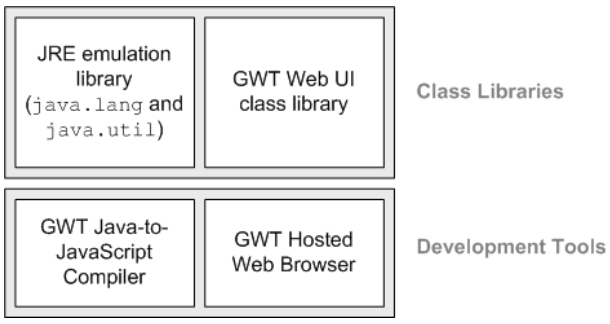

Figura 2.1.: Arquitectura de GWT

- **Compilador GWT Java-to-JavaScript:** la función principal de este componente es traducir el código Java al lenguaje JavaScript. Se emplea cuando se utiliza GWT en Web mode.

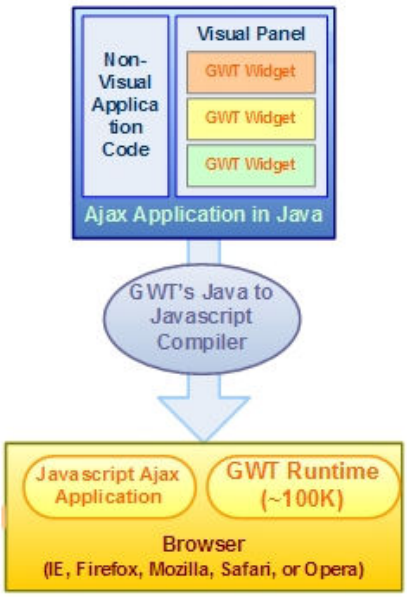

Figura 2.2.: Compilador Java-to-JavaScript de GWT

- **Navegador en modo hosted:** este componente permite la ejecución de la aplicación Java sin traducirla a JavaScript en modo hosted, donde lo que se ejecuta son bytecodes de Java sobre la JVM.

- **Emulación de librerías JRE:** contiene implementaciones en JavaScript de las librerías de clases más utilizadas en Java. Incluye la mayor parte de las clases del paquete *java.lang*, donde se encuentran las clases fundamentales para poder desarrollar en Java, y un subconjunto de las clases del paquete *java.util*. El resto de paquetes no están soportados de forma nativa por GWT.

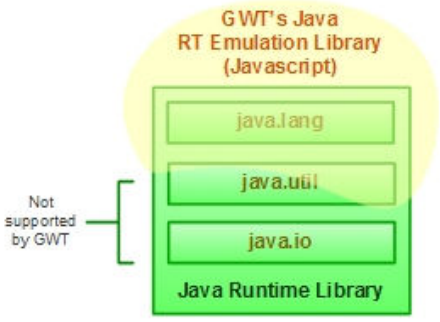

Figura 2.3.: Librerías JRE soportadas por GWT

- **Librería de clases de interfaz de usuario:** este componente contiene un conjunto de elementos de UI y clases personalizadas que permiten la creación de widgets para el navegador, como por ejemplo cajas de texto, imágenes, botones, etc.

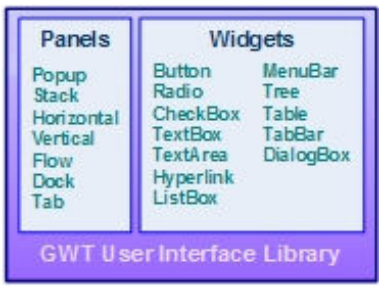

Figura 2.4.: Librería de clases de interfaz de usuario de GWT

En la figura 2.5 se puede apreciar el funcionamiento completo de los componentes anteriormente explicados:

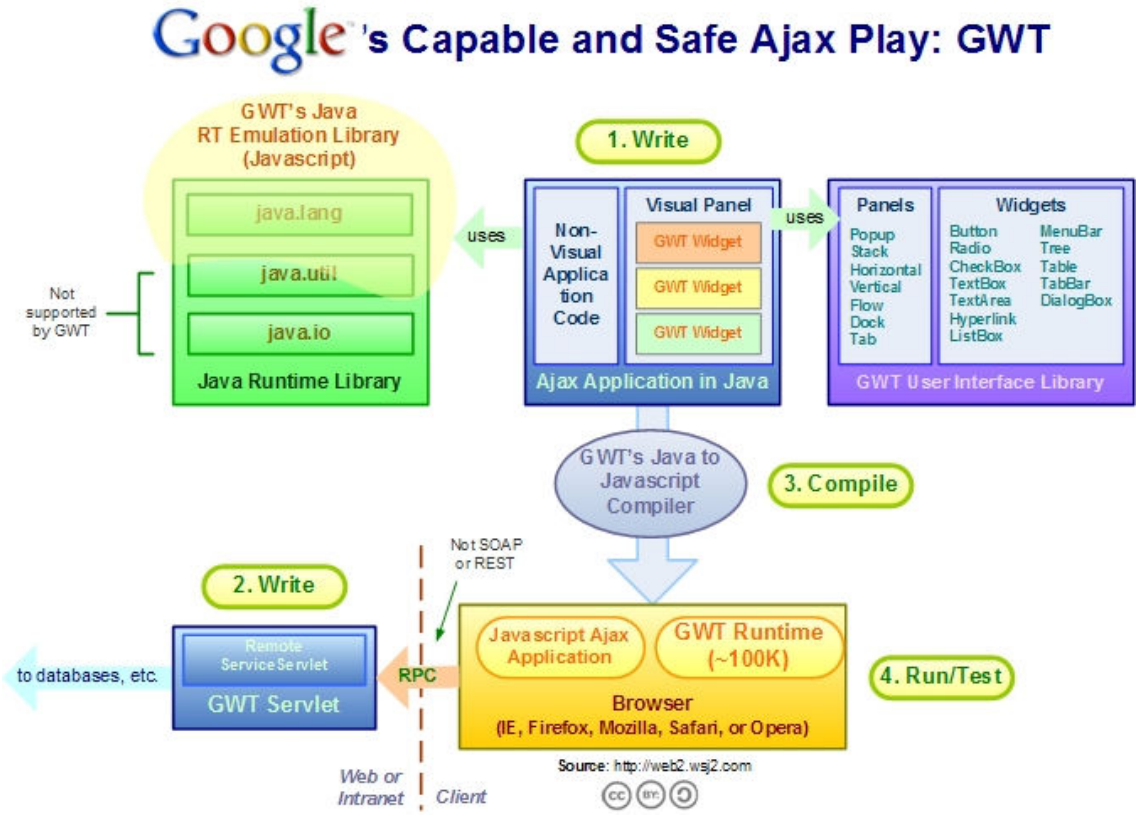

Figura 2.5.: Diagrama completo de los componentes de GWT

2.2.4. Herramientas

Existen multitud de herramientas y plugins FLOSS para realizar aplicaciones en GWT para diferentes entornos de desarrollo de una forma sencilla y rápida, ayudando a los desarrolladores a crear y organizar sus proyectos, diseñar las interfaces gráficas, implementar servicios remotos, ejecutar las aplicaciones en modo hosted, compilar el código fuente y desplegar el proyecto en un servidor Web, además de proporcionar otro tipo de herramientas que permiten interactuar con bases de datos, sistemas de bug-tracking, repositorios de control de versiones, etc.

Para este proyecto se han investigado dos posibilidades:

- **Plugin GWT4NB [44] para NetBeans [45]**
  Este proyecto está licenciado bajo licencia CDDL-1.0[21] [46] y tiene dos objetivos básicos, proporcionar un soporte mejorado para el desarrollo de aplicaciones web y la posibilidad de utilizar el framework GWT desde el IDE NetBeans.

  Sus principales características son las siguientes:

  - Permite utilizar GWT en proyectos nuevos o existentes.
  - Ejecutar, depurar y probar proyectos basados en Ant [47].
  - Soporte para Google App Engine [48].
  - Asistente y autocompletado de código para ficheros GWT.

- **Plugin Google para Eclipse [49]**
  Esta herramienta se encuentra bajo la licencia Eclipse Public 1.0 [50] y proporciona una serie de herramientas para que los desarrolladores puedan realizar el diseño, la construcción, la optimización y el despliegue de aplicaciones web de una forma rápida y sencilla

[21] Common Development and Distribution License

en Eclipse.

Las características principales de este plugin son las siguientes:

- Asistente de aplicaciones web.
- Importación de los APIs de Google más actuales.
- Despliegue rápido en App Engine.
- Integración con GWT Designer.

2.3. Wave

Uno de los principales frameworks que los que Kune se apoya, y en el cual está fuertemente influenciado, es Google Wave (que actualmente se llama Apache Wave debido a que esta fundación se encargó del proyecto desde finales del año 2010).

Esta plataforma es un software para la edición colaborativa en tiempo real, fue anunciada por Google en la conferencia Google I/O [51] de Mayo de 2009. Su principal cometido es proporcionar una herramienta web que combina las características principales de varias aplicaciones utilizadas para la comunicación (correo electrónico, mensajería instantánea, wikis, redes sociales, etc.), ya sea de forma síncrona o asíncrona y con la posibilidad de añadir extensiones (por ejemplo gadgets, robots, etc.) para realizar gran cantidad de funciones adicionales.

A finales de 2010 Google anunció el cierre del proyecto, y poco después la Fundación Apache [52] acepto el proyecto en su incubadora, con el nombre de Apache Wave. A partir de ese momento fue entregado a la fundación Apache para continuar el proyecto, que se basó principalmente en el desarrollo de un producto llamado Wave in a box (WIAB) [53], que es el servidor de referencia de varios protocolos y tecnologías de Wave.

2.3.1. Características

Las principales características de esta plataforma son las siguientes:

- Escrito en Java, utilizando OpenJDK [54] y GWT.
- Incluye funcionalidades de diferentes herramientas de comunicación, como el correo electrónico, mensajería instantánea, etc., pero de una forma mejorada mediante documentos de escritura colaborativa en tiempo real llamados Waves.
- Las Waves pueden ser compartidas con otros usuarios en cualquier momento.
- Permite la edición simultánea de cualquier parte del Wave por cualquier usuario y sin problemas de latencia.
- Notificaciones para cada cambio en la Wave.
- Permite ver los cambios en tiempo real.
- Edición simple, similar a una wiki.
- La historia de la Wave se almacena dentro de ella. De esta forma los colaboradores pueden utilizar la función de reproducción para ver el orden en que se editó una Wave o los cambios que se han añadido y que colaborador lo hizo.
- Dispone de gran cantidad de extensiones, gadgets y robots.
- Herramienta FLOSS bajo la licencia Apache.

2.4. Plataforma Kune

Kune es una herramienta web FLOSS que actúa como una red social para la promoción de métodos colaborativos entre distintos grupos de trabajo. Su principal objetivo es permitir la creación de espacios de trabajo colaborativo online, donde las organizaciones y las personas puedan crear sus proyectos, comunicarse entre ellos, gestionar el grupo, publicar contenidos sobre el proyecto y por supuesto contactar/comunicarse con personas y proyectos con intereses similares para poder intercambiar conocimientos y poder colaborar en conjunto.

Esta plataforma unifica las características de las redes sociales con las capacidades del software colaborativo, con el propósito de que cualquier grupo pueda desarrollar sus actividades y dispongan de una forma sencilla de publicar y promocionar sus contenidos.

2.4.1. Versiones

Kune es un proyecto que está aún en fase de desarrollo, los primeros prototipos se hicieron en Ruby on Rails [55] y Pyjamas [56], pero coincidiendo con la liberación de Java se implementó Kune utilizando Java y GWT. A día de hoy dispone de 4 versiones estables:

- **0.0.1**

  Es la primera versión estable del proyecto, se liberó en el año 2007 y contenía las funcionalidades básicas para la creación de usuarios, grupos, edición de contenido básico, navegación entre directorios, un chat basado en XMPP[22] [60], búsquedas, herramientas de traducción, y sobre todo la función de red social.

- **0.0.9 [59]**

  Esta versión fue liberada en agosto de 2011, e incluye grandes cambios, ya que tras la primera versión de Kune y la inclusión de muchas funcionalidades Google anunció el proyecto Google Wave, con la firme promesa de que sería un proyecto FLOSS. Google

[22] Extensible Messaging and Presence Protocol

Wave utilizaba gran parte de las mismas tecnologías que utiliza Kune, por tanto se decidió integrarlo en Kune, a la vez que se desarrollaban una serie de gadgets. Esta versión fue bautizada con el nombre de "15M", en honor al movimiento social ocurrido ese mismo año. La versión incluía una serie de características nuevas como son un servicio de correo de nueva generación alojado en el servidor de Kune para los proyectos, chat compatible con gmail y similares, posibilidad de convocar asambleas online, edición colaborativa y simultánea de documentos, gestión de contacto personales, espacio multimedia y la inclusión de extensiones útiles para colectivos.

- **0.1.0 [58]**

  Esta versión se liberó en abril de 2012, y fue bautizada con el nombre de "99 %" y que era una versión beta de la plataforma Kune. Esta versión se centró sobre todo en integrar el código fuente de Apache Wave (antes Google Wave), que fue liberado a finales de 2010 por la Fundación Apache. Además incluía una gran cantidad de cambios como son mejoras en la estabilidad, velocidad y usabilidad de la plataforma, compartición de contenidos, calendario colaborativo en tiempo real, nuevas herramientas para grupos, notificaciones por correo electrónico, tutoriales para cada herramienta y accesos directos.

- **0.2.0 [57]**

  Esta es la última versión liberada de Kune, se subió a producción en octubre de 2012 con el nombre clave "Ostrom", en homenaje al premio nobel de economía del 2009 Elinor Ostrom [61]. Esta versión contiene una gran cantidad de cambios, entre los cuales están el soporte multilenguaje, mejoras de interfaz y usabilidad, cambios de diseño, búsquedas mejoradas, integración con Apache Wave mejorada, soporte RTL, mejoras en las prestaciones y la resolución de gran cantidad de bugs.

2.4.2. Características

Las características principales de la plataforma Kune son las siguientes:

- Incluye todas las funcionalidades de Apache Wave.

- Proporciona múltiples herramientas de comunicación, como salas de chat y funciones de red social.

- Colaboración en tiempo real para grupos, tanto en documentos, wikis, calendarios, blogs, espacio web, intercambios, etc.

- Funciones de correo avanzado, con Waves, inbox donde se almacenan todas las conversaciones y documentos.

- Posibilidad de utilizar gran cantidad de gadgets (mapas, mindmaps, complementos de twitter, etc.).

- Federación.

- Herramienta orientada a la usabilidad.

- Software libre bajo la licencia AGPLv3[23] [62].

- Orientado a promover la cultura libre.

2.4.3. Arquitectura

Kune tiene una arquitectura que se divide en tres niveles, una capa para la parte cliente (navegador), otra capa para el contenedor web y una tercera con librerías para los diferentes servicios que se utilizan.

En la figura 2.6, se puede apreciar la arquitectura completa de Kune:

[23] Affero General Public License version 3

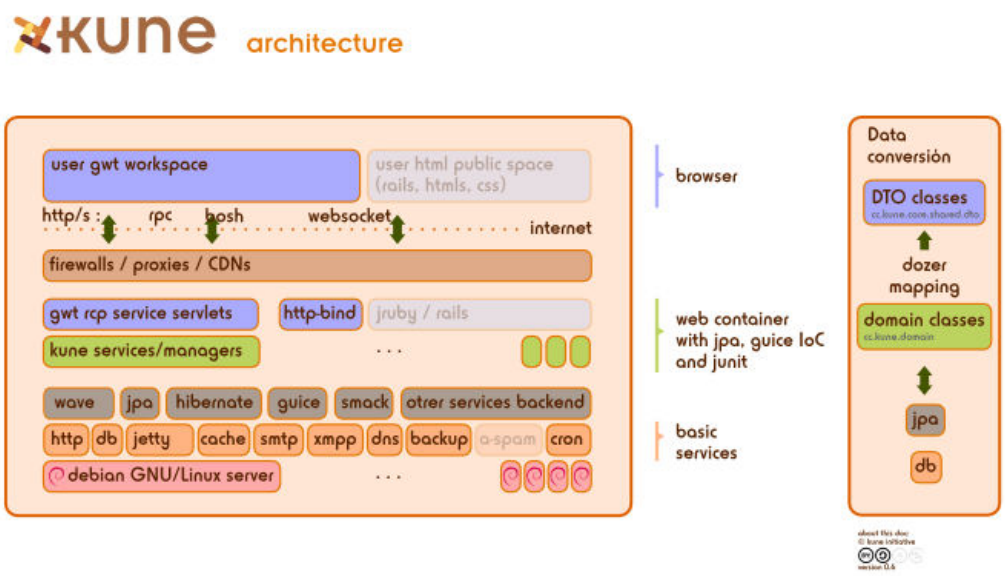

Figura 2.6.: Arquitectura de Kune

En resumen, Kune está desarrollado en GWT en la parte del cliente e integrado con Apache Wave, utilizando los protocolos abiertos de XMPP y Wave Federation Protocol [63]. De esta forma GWT genera en el cliente código JavaScript ofuscado y optimizado creando una aplicación de página única. Las extensiones de las Waves se ejecutan sobre Kune como aplicaciones y pueden ser desarrolladas en diferentes lenguajes como GWT, JavaScript o Python [64].

2.5. MySQL

El sistema de base de datos que utiliza actualmente la plataforma Kune es MySQL, y por tanto Troco, al ser un widget para esta plataforma, utilizará este mismo sistema de gestión de bases de datos para la persistencia de la información de cada intercambio entre los usuarios.

Como es bien conocido, MySQL es un SGBD[24] relacional, multihilo y multiusuario, desarrollado principalmente en C/C++ y que fue creado por la empresa MySQL AB (comprada por la compañía Sun Microsystems en el año 2008, posteriormente adquirida por la multinacional

[24]Sistema de Gestión de Bases de Datos

Oracle en abril de 2009, como se ha mencionado anteriormente).

MySQL dispone de doble licenciamiento, por un lado se distribuye como FLOSS bajo la licencia GNU GPL y por otro lado se ofrece con una licencia privativa en su versión enterprise. Esto se consigue debido a que el desarrollo principal de MySQL lo realiza una empresa privada que mantiene el copyright del software, a diferencia de otros proyectos FLOSS donde el software es realizado por una comunidad pública y los derechos de autor del código son del autor de esos cambios.

MySQL ha tenido un gran auge en torno a las aplicaciones Web, debido a su gran rendimiento y velocidad en acciones de lectura de datos en modelos no transaccionales.

2.5.1. Versiones

La primera versión de este proyecto salió en el año 1996, desde ese momento se han liberado gran cantidad de versiones, aunque actualmente sólo se conservan información de las versiones 3.x, 4.x y la actual 5.x. La versión más reciente es la 5.6, donde todavía se encuentra el desarrollo activo.

A continuación, se expondrán brevemente las diferentes versiones que existen de esta herramienta, así como los cambios más relevantes que han incluido en cada una de ellas:

- **3.23**

  La primera versión de esta revisión fue liberada en julio de 1999, contenía gran cantidad de cambios y nuevas características, entre todas ellas se pueden destacar la incorporación de tres nuevos tipos de tablas:

  - MyISAM[25].

[25] http://dev.mysql.com/doc/refman/5.0/es/myisam-storage-engine.html

- InnoDB[26].

- BDB o BerkeleyDB[27].

También se añadió soporte para la replicación de bases de datos entre una base de datos maestra y varias esclavas o la indexación de texto completo. Este ciclo de desarrollo estuvo activo hasta septiembre de 2003 y tuvo 59 revisiones menores donde se corrigieron diferentes problemas y fallos de la versión. Actualmente es una versión obsoleta y carece de soporte.

- **4.0**

  Esta amplia versión se liberó en octubre de 2001, incluía multitud de cambios y características nuevas entre las que destacan, la inclusión del motor de almacenamiento de InnoDB, caché de consultas para mejorar el rendimiento de las aplicaciones, mejoras en la indexación de texto completo, sintaxis de UNION SELECT, borrado de múltiples tablas, biblioteca de servidor (libmysqld) incluida, opciones adicionales para el control de privilegios, gestión de recursos de usuario, variables dinámicas en el servidor, mejoras de código de replicación y cambios para la mejora del rendimiento y la fiabilidad. Este ciclo de desarrollo llegó a su fin con la versión menor 4.0.31, donde se corrigieron diferentes errores de la versión, la última se liberó en febrero de 2007. Actualmente es una versión obsoleta y carece de soporte.

- **4.1**

  Esta versión comenzó en abril de 2003, su ciclo de desarrollo duró hasta diciembre de 2008, y cuenta con 25 versiones menores. Algunos de los principales cambios son modificaciones en la función SUBSTRING, mejoras en las subconsultas y las tablas derivadas, soporte para UTF-8, nueva caché de claves para tablas de tipo MyISAM, índice BTREE en las tablas HEAP, soporte para tipos de datos geográficos, mostrar advertencias de sintaxis y nuevas opciones para la línea de comandos. Actualmente es una versión obsoleta y carece de soporte.

[26] http://dev.mysql.com/doc/refman/5.0/es/innodb.html
[27] http://dev.mysql.com/doc/refman/5.0/es/bdb-storage-engine.html

- **5.0**

  El ciclo de desarrollo de esta versión comenzó en diciembre de 2003, y duró hasta marzo de 2012. Actualmente no dispone de soporte por parte de la comunidad. Esta versión es una versión bastante amplia, con gran cantidad de cambios relevantes como por ejemplo, mejoras en la replicación, soporte para cursores de sólo lectura a nivel de servidor, soporte para vistas, soporte para procedimientos almacenados, mejoras en el log del servidor, eliminación del soporte para tablas ISAM, soporte para opciones RAID[28] de tablas MyISAM, así como el arreglo de gran cantidad de bugs.

- **5.1**

  En esta versión se comenzó en noviembre de 2005, su ciclo de desarrollo no ha finalizado, dispone de 70 versiones menores y la última se ha liberado en abril de 2013. Los principales cambios de esta versión son mejoras en la partición permitiendo distribuir porciones de tablas a lo largo del sistema de ficheros, se permite la replicación basada en filas que ha diferencia de la replicación basada en sentencias SQL permite replicar determinadas filas de tabla, se añade soporte para incorporar plugins, planificación de eventos, tablas de log de servidor en lugar de ficheros de log, un sistema para la actualización del programa que chequea incompatibilidades entre versiones, se separa MySQL Cluster como producto independiente y se añade soporte para funciones XML con Xpath[29].

- **5.5**

  El ciclo de desarrollo de esta versión comenzó en diciembre de 2009, y continúa activo actualmente. Tiene 32 versiones menores la última de ellas liberada en abril de 2013. Los principales cambios y mejoras que tiene esta versión son nuevas opciones de autenticación, mejoras en la escalabilidad de procesadores con varios núcleos, mejor acceso a información de ejecución y rendimiento, mejoras para Solaris, cambio del motor de almacenamiento por defecto a InnoDB en lugar de MyISAM, replicación semi síncrona, soporte para caracteres complementarios en Unicode, mejoras en la partición de tablas, soporte para IPv6 y mejoras en funcionalidades XML.

[28] Redundant Array of Independent Disks

[29] http://www.w3schools.com/xpath/

- **5.6**

  Esta versión comenzó su desarrollo en abril de 2011 y se encuentra en pleno desarrollo, la última versión menor liberada fue la versión 5.6.11 en abril de 2013. Los cambios y mejoras más relevantes que incluye esta versión son los siguientes, mejoras en la seguridad, mejoras en el particionado de tablas, nueva versión de MySQL Cluster, mejoras en la replicación, mejora de la optimización de las querys y se añaden características en algunos tipos de datos de fecha para obtener mayor precisión.

2.5.2. Características

Las características principales de este SGBD que están incluidas tanto en la versión FLOSS como en la versión comercial son las siguientes:

- Desarrollado en C/C++.
- Multiplataforma.
- Dispone de APIs para gran cantidad de lenguajes de programación, entre otros C, C++, Perl, Java, Php, Python y Ruby.
- Optimizado para equipos con múltiples procesadores.
- Gran velocidad y robustez.
- Permite la utilización como cliente-servidor o incrustado en aplicaciones.
- Su administración se basa en usuarios y privilegios.
- Soporta gran cantidad de tipos de datos para las columnas.

- Uso completo de multi-threads.
- Proporciona sistemas de almacenamiento transaccionales y no transaccionales.
- Usa tablas en disco B-tree muy rápidas con compresión de índice.
- Sistema de reserva de memoria basado en threads.
- Funciones SQL implementadas usando una librería optimizada.
- Multilenguaje.
- Dispone de una versión FLOSS, licenciada bajo GNU GPL.

2.5.3. Arquitectura

MySQL separa el motor de almacenamiento, que es el encargado de las acciones de entrada/salida y la representación de la información, del resto de componentes. A continuación, en la figura 2.7, observamos una representación de la arquitectura de MySQL:

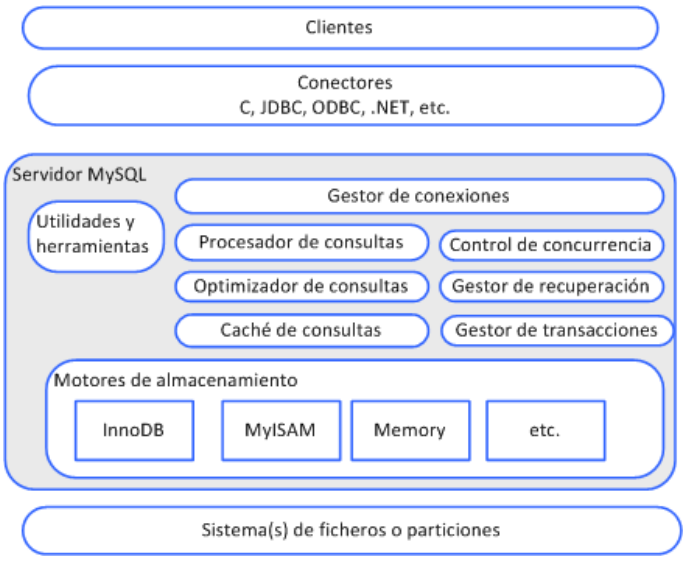

Figura 2.7.: Arquitectura de MySQL [65]

- **Gestor de conexiones:** este componente es el encargado de mantener y gestionar las múltiples conexiones de los clientes. Debido a que crear y destruir conexiones son procesos costosos que consumen muchos recursos, desde este gestor se permite establecer el límite de conexiones concurrentes para no perder rendimiento del servidor.

- **Procesador de consultas:** cuando una consulta llega al gestor de MySQL, este componente analiza la sentencia sintácticamente y se produce una representación intermedia de la misma, mediante la cual MySQL realiza una serie de acciones para determinar el orden de lectura de las tablas, el uso de índices o la reescritura de la consulta de forma más eficiente.

- **Optimizador de consultas:** es el encargado de preguntar al gestor el tipo de características que soporta para la optimización de consultas. Estas características de optimización dependen del motor de almacenamiento seleccionado.

- **Caché de consultas:** este componente implementa una caché de consultas con los resultados que devuelven, de forma que antes de que el procesador de consultas llame al optimizador, busca la consulta en este componente para un acceso más rápido los resultados si se encuentra almacenada en esta caché.

- **Control de concurrencia:** es un mecanismo que se utiliza para evitar que acciones de consulta o escritura de forma simultánea sobre los mismos datos produzcan una inconsistencia de los datos o pérdidas en los mismos. Este componente se encarga de realizar bloqueos sobre esos datos cuando se realiza una acción sobre estos y no se libera el bloqueo hasta que finalice.

- **Gestor de transacciones y gestor de recuperación:** este componente es el encargado de proporcionar la funcionalidad de una operación transaccional, donde todos los cambios realizados a nivel de datos son efectivos si todo es correcto, pero donde se puede volver atrás o hacer una recuperación de los datos anteriores en caso de que se produzca algún fallo (rollback).

- **Motores de almacenamiento:** uno de los principales componentes de MySQL, es la denominada "pluggable storage engine architecture", donde se define una interfaz común para la gestión de datos a nivel físico, permitiendo de esta forma que el gestor de almacenamiento pueda intercambiarse, y así poder utilizar el motor más adecuado para cada necesidad específica, aprovechando las ventajas de cada uno de ellos.

2.6. Git y Gitorious

Uno de los objetivos de este proyecto ha sido migrar el sistema de control de versiones (SCM[30]) de la herramienta Troco del repositorio actual en Subversion [66] a un repositorio Git hospedado en la plataforma Gitorious.org [67].

Git es un sistema de control de versiones distribuido FLOSS, se encuentra bajo la licencia GPLv2, y que fue diseñado y desarrollado en 2005 por Linus Torvalds [68], cuyos principales objetivos son la gestión eficiente y rápida de proyectos software, de forma que proporciona una copia completa y funcional del repositorio en local, independizado de la red y de la comunicación con el repositorio remoto central.

La principal diferencia entre este SCM y la gran mayoría de los que existen en el mercado, entre ellos Subversion, es que modela sus datos de una forma completamente diferente, ya que la mayor parte de los demás SCM modelan la información que almacenan como un conjunto de archivos y cada una de las modificaciones realizadas sobre cada uno de ellos a lo largo del proyecto. Sin embargo, Git modela sus datos como un conjunto de instantáneas de un pequeño sistema de archivos, de forma que cada vez que se confirma un cambio, este sistema guarda el estado de cada uno de los ficheros modificados y una referencia para aquellos que no han sido modificados.

[30] Source Code Management

2.6.1. Versiones

Desde la primera versión que fue liberada en abril de 2005, se han liberado una gran cantidad de versiones, donde se han ido añadiendo gran cantidad de cambios y funcionalidades nuevas, hasta la versión actual que es la 1.8.3 y que ha sido liberada recientemente en mayo de 2013.

A continuación se expondrán los detalles de algunas de las versiones más recientes de este SCM:

- **1.8.0**

  Esta versión fue liberada en octubre de 2012, donde se incluyen varias mejoras y cambios, como mejoras en las credenciales para GNOME[31] y Win32, mejoras en el demonio de git, ampliación de funcionalidades y opciones de gran cantidad de comandos, soporte para la versión SVN 1.7[32] para la herramienta git svn, mejoras internas de funcionamiento y rendimiento y corrección de errores de versiones anteriores.

- **1.8.1**

  Esta versión, que se liberó en diciembre de 2012 y donde se añadían diferentes cambios con respecto a la versión anterior como la ampliación de la documentación a formato HTML, mayores opciones en los comandos de git, mejoras en la utilidad csvimport, mejoras en la interfaz remota para Mercurial, optimización de la lógica de búsqueda de atributos y resolución de problemas de la versión anterior.

- **1.8.2**

  En esta versión, liberada en febrero de 2013, se incluyeron varios cambios, desde corrección de bugs de la versión anterior como mejoras en los test para indicar su resultado, mejoras en los patrones de .gitignore, mejoras en el soporte con Bazaar [69] y Mercurial [70], cambio en variables de entorno internas y mejoras en las funciones de limpieza del repositorio.

[31] http://www.gnome.org/
[32] Subversion

- **1.8.3**

  Esta es la última versión liberada, en mayo de 2013, que incluye actualizaciones en el soporte remoto para Mercurial y Bazaar, mejoras en las opciones de gran cantidad de comandos de Git, mejoras para construir bajo Microsoft Visual C++ y el arreglo de bugs de versiones anteriores.

2.6.2. Características

Las características principales de Git son las siguientes:

- Facilita el desarrollo no lineal, proporcionando rapidez en la gestión de branches y mezcla de diferentes versiones.

- Ocupa poco espacio y sus operaciones son extremadamente rápidas.

- Gestión distribuida, proporciona una copia local del repositorio completo.

- Los repositorios pueden publicar a través de HTTP, FTP, rsync o mediante un protocolo nativo, ya sea a través de TCP/IP o SSH[33].

- Herramientas de integración con Subversion, Bazaar y CVS.

- Gran eficiencia en la gestión de proyectos grandes.

- Todas las versiones previas a un cambio determinado, implican la notificación de un cambio posterior en cualquiera de ellas a ese cambio.

- Es FLOSS, bajo licencia GPLv2[34].

[33] Secure SHell

[34] General Public License version 2

- Realmacenamiento periódico en paquetes (ficheros).

2.6.3. Arquitectura

Como se puede apreciar en la figura 2.8, la arquitectura de Git se divide en los componentes locales y el repositorio remoto, donde se almacenan todos los cambios de cada uno de los desarrolladores que mezclan las diferentes ramas en él.

Por otro lado, y como se ha mencionado anteriormente, se dispone de una parte local que se compone de varias áreas diferenciadas, por un lado se dispone de un directorio de trabajo local, que es un directorio del sistema de ficheros donde se encuentra una copia completa del repositorio remoto.

Otro área que se encarga de almacenar los ficheros nuevos que se desea añadir al repositorio, llamada "staging area", que es como una zona temporal donde se guardan las referencias a los nuevos ficheros hasta que son confirmados en el repositorio local. Por último la tercera área, que como hemos dicho es una copia completa del repositorio remoto, donde se almacenarán los cambios realizados por el desarrollador y que podrán sincronizarse con el repositorio remoto de una manera sencilla y rápida.

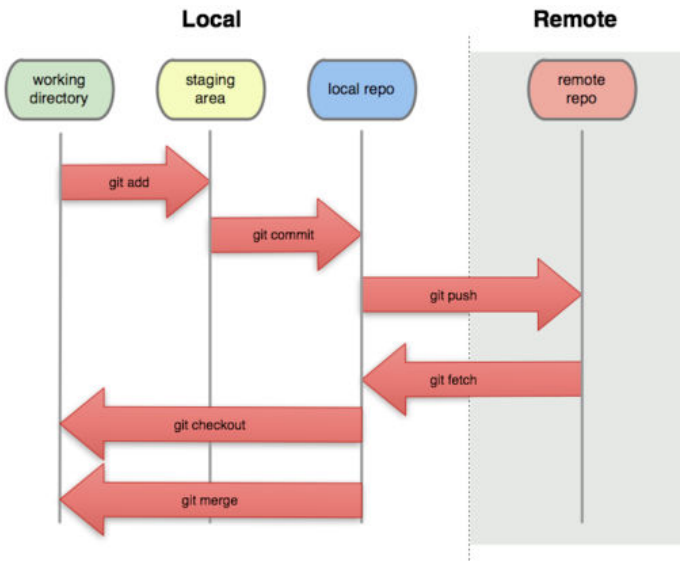

Figura 2.8.: Arquitectura de Git

2.6.4. Gitorious

Como proveedor de servicios en la nube para la gestión de proyectos FLOSS que utilizan como repositorio Git, se ha seleccionado Gitorious.

Esta infraestructura, que a su vez es FLOSS, permite crear proyectos donde se pueden almacenar uno o varios repositorios que se pueden compartir, gestionar y utilizar por parte de todos los colaboradores del proyecto. Al trabajar con Git permite descargar una copia local del repositorio para cada uno de los colaboradores, de forma que puedan trabajar de forma autónoma y después subir y fusionar sus contribuciones con el repositorio general del proyecto, quedando un registro de cada cambio realizado por cada uno de los colaboradores y permitiendo establecer notificaciones para su seguimiento.

Las características principales de Gitorious son las siguientes:

- Permite el alojamiento de proyectos.
- Posibilidad de alojar diferentes repositorios oficiales de los proyectos.
- Wiki de proyectos.
- Peticiones de fusión públicas y revisión del código.
- Actividad del proyecto en línea.
- Perfiles de desarrolladores y registro de actividades.
- Sistema de notificaciones.
- Gratuito para proyectos FLOSS.

- Licenciado bajo GPLv3[35].

2.7. gwt-pectin

Una de las herramientas FLOSS elegidas para este proyecto para la capa de presentación, es la librería para la creación y validación de interfaces de usuario para GWT, gwt-pectin. Esta librería está basada en los patrones Value Model, Presentation Model y Passive View, y nos proporciona un API para la definición de los modelos, los comandos y los formularios así como la comunicación con los widgets de GWT. De esta forma nos permite definir y reutilizar gran cantidad de validaciones y comportamientos para los distintos formularios que tendrá la nueva versión del gadget de Troco.

2.7.1. Versiones

A continuación, se expondrán brevemente las diferentes versiones que existen de esta librería, así como los cambios más relevantes que han incluido en cada una de ellas:

- **0.1.0**

  Esta fue la primera versión de la librería, fue liberada en agosto de 2009. Contenía la funcionalidad principal para la creación y validación de formularios en GWT.

- **0.1.1**

  Esta versión se publicó en septiembre del 2009 y contenía una serie de parches y mejoras, así como la simplificación del código fuente.

- **0.1.2**

  Esta versión fue liberada muy poco después que la versión 0.1.1, e incluía nuevas carac-

[35] General Public License version 3

terísticas como el soporte para JUnit en GWT y mejora en las excepciones de los diferentes componentes, así como actualizaciones en la documentación.

- **0.1.3**

  En esta versión, que fue publicada en noviembre de 2009, se realizaron cambios en las funciones de los componentes de la librería, se añadió el soporte para marcas de agua en las cajas de texto, se actualizó el entorno para trabajar con Snow Leopard y se resolvieron varios errores de versiones anteriores.

- **0.4**

  Esta versión se liberó también en noviembre de 2009, y supuso un cambio en la denominación de las versiones, para que fuese más sencillo. Añade varias características nuevas características como la integración con UiBinder, cambios el la validación de estilos, nuevos validadores básicos y el arreglo de algunos bugs.

- **0.5**

  En enero de 2010, se publicó esta versión que no incluía grandes cambios, sobre todo se centró en mejorar algunos componentes, resolviendo algunos problemas, añadiendo nuevas funciones internas y simplificando el plugin de validación.

- **0.6**

  Esta versión fue liberada en febrero de 2010 e incluía gran cantidad de cambios entre los cuales había refactorización de código y recolocación de fuentes, mejoras en el acceso a los mensajes de validación externos, arreglo de varios errores de versiones anteriores y se añaden clases para facilitar los test unitarios.

- **0.7**

  Esta versión se liberó en abril de 2010. Es la primera versión que dispone de UiCommands, además se realizan diferentes arreglos de problemas de la versión anterior y se añaden más test.

- **0.8**

Esta versión se lanza en julio de 2010 y su objetivo es mejorar la interacción entre los componentes, realizando cambios en cada uno de ellos y refactorizando funcionalidades. Además se añadieron nuevas características como extender la marca de agua para otros widgets, corrección de fallos y la creación de más test.

- **0.8.1**

  Esta versión surge en junio de 2011 para solventar los problemas de compatibilidad con la versión 2.2 de GWT, recompilando el código con esa versión de GWT.

2.7.2. Características

Las características principales de esta librería son las siguientes:

- API definido para crear y unir modelos.
- Soporte para componentes básicos como valores, listas, condiciones, funciones, comandos, etc.
- API para formularios avanzados.
- No se requieren widgets especiales, trabaja con los widgets de GWT.
- Plugins para metadatos y validaciones.
- Enlaces adicionales para controlar los estilos CSS[36].

[36]Cascading Style Sheets

3. Otras Plataformas

En esta sección se expondrán en detalle varias de las herramientas que actualmente existen para el intercambio de bienes y servicios de diferente índole, y que se han mencionado anteriormente en la sección 1.3. Además se realizará una comparativa entre las características de las que disponen estas herramientas y las que proporcionará el gadget de Troco.

A continuación explicamos brevemente las herramientas que existen en la actualidad para realizar intercambios online con otras personas:

- **Trocobuy**

  Esta plataforma online comenzó en marzo de 2012 por la empresa Vigiliam Business Social Network, y su propósito principal es proporcionar una red de intercambio de bienes y servicios alternativo e innovador para PYMES, autónomos y emprendedores, que se basa en el compromiso y solidaridad entre sus miembros. De esta forma proporciona las ventajas de acceder a un crédito comercial responsable, ganar nuevos clientes entre los miembros de la plataforma y mejorar la tesorería de la empresa al no utilizar dinero real.

  Las características principales de esta herramientas son las siguientes:

  - Es una herramienta centralizada.

  - Utiliza un sistema de registro de usuarios propio y está restringido a pymes, autónomos y emprendedores.

- Permite el intercambio de cualquier tipo de bienes y servicios entre los usuarios de la plataforma.

- La dificultad para la utilización de la herramienta es medio, ya que implica conocimientos financieros y empresariales.

- No dispone de sistema de valoración, se basa en la confianza de los miembros.

- Dispone de una guía rápida, de página de FAQS y de blog.

- No es una herramienta FLOSS, dispone de licencia privativa.

- Es multilenguaje.

- **Truequeweb**

  Esta página web comenzó en el año 2007 y proporciona una plataforma para poder realizar intercambios de bienes y/o servicios sin intercambio económico de una manera rápida y sencilla, una vez aceptado el intercambio por ambas partes proporciona los datos de contacto del emisor del intercambio para que puedan acordar la forma de entrega de lo pactado en el trueque.

  Esta herramienta tiene las siguientes características:

  - Centralizada.

  - Utiliza un sistema de registro de usuarios propio.

  - Permite el intercambio de cualquier tipo de bienes y servicios legales entre los usuarios de la plataforma.

  - No dispone de una gran dificultad ni para el registro ni para el propio intercambio.

- Dispone de sistema de valoración de los usuarios de la plataforma.
- Sólo dispone de una guía rápida.
- Tampoco es una herramienta FLOSS.
- Sólo disponible en español.

- **Bookmooch**

  Esta comunidad online internacional fue creada en el año 2006 por John Buckman [73] con el propósito de proveer una herramienta para crear una red mundial abierta y gratuita de intercambio de libros de segunda mano. Actualmente tiene miembros de más de 90 países y funciona con un sistema de puntos que se obtienen al ofrecer libres y que se pueden canjear por libros que ofrecen otros usuarios de la comunidad.

  Esta comunidad dispone de las siguientes características:

  - Es centralizada.
  - Utiliza un sistema de registro de usuarios propio.
  - Es una plataforma exclusivamente para el intercambio de libros.
  - El registro es sencillo y la utilización también.
  - Dispone de sistema de feedback para valorar cada intercambio.
  - Entre su documentación hay una guía rápida y una página de FAQS.
  - No es software libre.
  - Múltiples idiomas.

- **Titletrader**

  Esta plataforma online fue creada en el año 2008 como un club de intercambio de libros, películas y música. Al igual que la comunidad Bookmooch utiliza un sistema de puntos denominado "Request Points", que se adquieren al subir los bienes que deseas intercambiar con otros usuarios y sirven para poder realizar intercambios con otros usuarios.

  Las características principales de esta plataforma son las siguientes:

  - Es una herramienta centralizada.
  - Utiliza un sistema de registro de usuarios propio.
  - Es una plataforma exclusivamente para el intercambio de libros.
  - De fácil registro y utilización.
  - Dispone de sistema de feedback para valorar cada intercambio.
  - Dispone de guía rápida, página de FAQS y foros.
  - No es una herramienta FLOSS.
  - Sólo disponible en inglés.

- **Etruekko**

  Esta página es una red de comunidades autogestionadas en la que los miembros de las distintas comunidades pueden intercambiar todo tipo de bienes y servicios mediante los sistemas de trueque y bancos de tiempo a través de su moneda social el "truekko", aportando valor a su comunidad.

  Esta plataforma dispone de las siguientes características:

- Es una herramienta centralizada.
- Utiliza un sistema de registro de usuarios propio.
- Permite el intercambio de bienes y servicios entre los miembros de las diferentes comunidades a través de moneda social.
- De fácil registro y utilización.
- Dispone de sistema de valoración de los usuarios.
- Dispone de guía rápida, página de FAQS y foros.
- No es una herramienta FLOSS, aunque está desarrollado con el framework FLOSS django que está bajo la licencia BSD.
- Sólo disponible en español.

- **Cadenadecambios**

  Esta plataforma web surgió con el propósito principal de ofrecer un sistema de intercambio de bienes y servicios basado en el concepto de comercio social pero en cadena, y no con el sistema de trueque convencional entre dos personas. De esta forma los miembros de la plataforma pueden ofrecer cualquier objeto o servicio con el objetivo de conseguir algo en particular, en ese mismo momento se inicia una búsqueda de cadena o bien se inicia una nueva hasta que esta se cierra consiguiendo cada miembro el objeto o el servicio deseado.

  Las características de esta herramienta son las siguientes:

  - Es una herramienta centralizada.
  - Utiliza un sistema de registro de usuarios a través de la red social Facebook.

- Permite el intercambio en cadena de bienes y servicios entre los usuarios.

- La dificultad de registro es media, ya que depende de una plataforma externa, sin embargo es de fácil utilización.

- Dispone de sistema de feedback de los usuarios.

- Dispone de guía rápida, página de FAQS y blog.

- No es una herramienta FLOSS.

- Solo disponible en español.

- **Polyglot**

  Esta comunidad proporciona un marco para aprender idiomas mediante el intercambio de conocimientos lingüísticos. Para ello los usuarios indican los idiomas que dominan y el nivel que tienen, así como los que desean aprender y contactan con otros usuarios que quieran aprender idiomas que ellos dominan y a cambio les enseñan alguno de los idiomas que quería aprender.

  Las características de esta comunidad son las siguientes:

  - Centralizada.

  - Utiliza un sistema de registro de usuarios propio.

  - Permite el intercambio de habilidades y conocimientos lingüísticos.

  - Es sencillo de utilizar y para registrarse.

  - No dispone de sistema de valoración.

- Sólo dispone de página de FAQS.
- No es una herramienta FLOSS.
- Es multilenguaje.

- **Home for Home**

  Esta comunidad online está orientada hacia el intercambio temporal de casas en diferentes países de Europa, permitiendo que los usuarios ofrezcan sus viviendas a cambio de la vivienda de otro usuario durante el mismo periodo de tiempo.

  Las características de esta comunidad son:

  - Es una herramienta centralizada.
  - Utiliza un sistema de registro de usuarios propio.
  - Permite el intercambio de habilidades y conocimientos lingüísticos.
  - Es sencillo de utilizar y para registrarse.
  - Dispone de un sistema de valoración de los usuarios.
  - Sólo dispone de página de FAQS.
  - No es una herramienta FLOSS.
  - Múltiples idiomas.

Como hemos podido ver todas las herramientas que existen en el mercado son herramientas centralizadas, no son herramientas FLOSS, y en la gran parte de las ocasiones tienen un propósito específico, lo que hace que el ámbito de intercambio se reduzca a un tipo específico.

Todas utilizan un sistema de registro en algunos casos con usuarios de la propia plataforma y en otros casos a través de redes sociales. Además de que estas herramientas no son FLOSS, la mayor diferencia que existe con Troco es que éste es un sistema descentralizado, donde la gestión del sistema queda supeditado a los grupos de trabajo o las asociaciones que lo quieran utilizar, y en su defecto a los propios usuarios que son los que determinan las reglas y condiciones de los intercambios, así como los productos y servicios que desean cambiar.

El resto de características son compartidas por Troco. Este gadget es multilenguaje, está planteado para ser sencillo de utilizar, dispone de documentación, etc., y aunque no dispone actualmente de un sistema de valoración, se implementará en un futuro como otro gadget para la plataforma, que no sólo sirva para Troco, sino que sea independiente de esta herramienta y reutilizable en cualquier otro tipo de componente.

4. Objetivos

En este capítulo se expondrán los objetivos principales que se pretenden alcanzar inicialmente en el desarrollo del nuevo gadget de Troco, que como se mencionaron brevemente en el capítulo 1 son los siguientes:

- Nuevo repositorio de código en Gitorious.org.
- Replanteamiento de la funcionalidad del gadget.
- Rediseño de la interfaz de usuario.
- Participación en la implementación de la nueva funcionalidad del gadget.

4.1. Nuevo repositorio de código en Gitorious.org

La primera versión del gadget de Troco se encuentra alojado actualmente en el repositorio de código de ourproject.org que utiliza el sistema de control de versiones Subversion, donde ha estado alojado desde el incio del desarrollo. Para esta evolución uno de los requerimientos era alojar la nueva versión del gadget en la plataforma Gitorious.org, utilizando Git como gestor de versiones del código fuente, por lo que se procedió a crear un nuevo proyecto llamado *"troco"* [71], donde se ha subido la documentación inicial, como los mockups, hasta el código fuente generado.

4.2. Replanteamiento de la funcionalidad del gadget

Como se ha mencionado anteriormente, la primera versión del gadget de Troco permite un intercambio simple y directo entre dos usuarios. Un objetivo principal es replantear esta funcionalidad y dar cabida a un mayor abanico de intercambios, como pueden ser intercambio temporal de bienes, petición de productos y servicios a otros usuarios, oferta de intercambios, pujas, cadena de intercambios, etc.

Además, para esta nueva versión del gadget, uno de los objetivos es ofrecer la posibilidad de utilizar la herramienta a través de un asistente, orientado a facilitar su utilización a los nuevos usuarios de la plataforma de una forma guiada, y también la posibilidad de realizar directamente el intercambio, más rápido y adecuado para usuarios experimentados.

4.3. Rediseño de la interfaz de usuario

Uno de los objetivos más destacados de esta versión es mejorar el aspecto visual del gadget, no sólo para resultar una herramienta más atractiva y actual que el gadget existente, sino para mejorar el flujo y la usabilidad del nuevo gadget. Este rediseño nos obliga a rehacer y rediseñar toda la lógica interna de la aplicación, para dar cabida a las nuevas funcionalidades y para poder adaptarnos a las especificaciones de la librería para la creación y validación de interfaces de usuario pectin, que hemos seleccionado para nuestro proyecto.

4.4. Participación en la implementación de la nueva funcionalidad de gadget

Como hemos comentado anteriormente, otro de los objetivos es añadir mayor funcionalidad a la nueva aplicación de Troco, ya que el flujo del que dispone actualmente es extremadamente

básico. Para ello será necesario finalizar tanto el replanteamiento de la funcionalidad como el rediseño de la interfaz de usuario para poder abordar este objetivo del proyecto.

5. Descripción del Proyecto

5.1. Metodología

En este apartado se explicará detalladamente la metodología utilizada para el desarrollo del proyecto, desde las primeras tomas de contacto hasta llegar al estado actual del proyecto.

Como es necesario en cualquier proyecto software, se deben realizar una serie de pasos y acciones para transformar el planteamiento inicial en la herramienta final que se obtiene cuando finaliza ese desarrollo y es posible su implantación.

Habitualmente se sigue un plan de trabajo marcado para la realización de un proyecto siguiendo las pautas definidas por la metodología seleccionada, pero en este caso no se ha llevado un plan de trabajo determinado, sino que se ha ido debatiendo entre todos los participantes del proyecto los pasos a seguir y cada una de las acciones que se debían realizar en cada etapa.

Se ha seguido un modelo de desarrollo lineal o en cascada con retroalimentación, que define una serie de etapas donde cada una de ellas debe finalizar para poder continuar con la siguiente etapa del desarrollo, y en el caso de que se produzca un cambio o se encuentre algún problema durante esa etapa permite retroceder a cualquier etapa anterior para poder readaptar el proyecto. Este modelo es una variante del tradicional modelo en cascada, que es ampliamente utilizado en adaptaciones o mejoras de proyectos software cuando se tienen claros los requerimientos y permanecen estables.

Este modelo no tiene definidas de forma estándar las fases del proyecto, pero habitualmente se pueden establecer una serie de fases que son comunes a cada proyecto y son las siguientes:

- Análisis de requerimientos.
- Análisis de riesgos.
- Diseño del sistema.
- Desarrollo.
- Verificación.
- Implantación.

Este modelo de desarrollo es uno de los modelos más antiguos que se utilizan en la ingeniería del software y ha sido puesto en tela de juicio constantemente por los problemas que tiene, que son los siguientes:

- Este modelo puede llevar a la confusión y a la incertidumbre del proyecto, debido a que en raras ocasiones los requerimientos y especificaciones de los proyectos software están completamente claros y cerrados y por tanto suelen sufrir cambios a medida que avanza el tiempo.
- En este tipo de proyecto no es sencillo readaptar el proyecto ante algún cambio o problema, debido a que se deben repetir las fases anteriores, por lo que es necesario detectar estos problemas en las primeras etapas.
- No se dispone de una versión estable hasta casi la finalización del proyecto, por lo que el cliente puede perder la paciencia si se dilata mucho en el tiempo.

Sin embargo, y a pesar de estos inconvenientes, es un modelo de desarrollo sencillo de aplicar

y rápido para proyectos con los requerimientos establecidos de una forma clara y constante, que si se planifica bien o el equipo de desarrollo es pequeño nos permite abordar cada fase fácilmente, estableciendo puntos de control y detección de problemas para las siguientes fases, y permitiendo volver a fases anteriores para realizar cambios de una manera rápida y sencilla, minimizando los riesgos del proyecto en la fase de desarrollo.

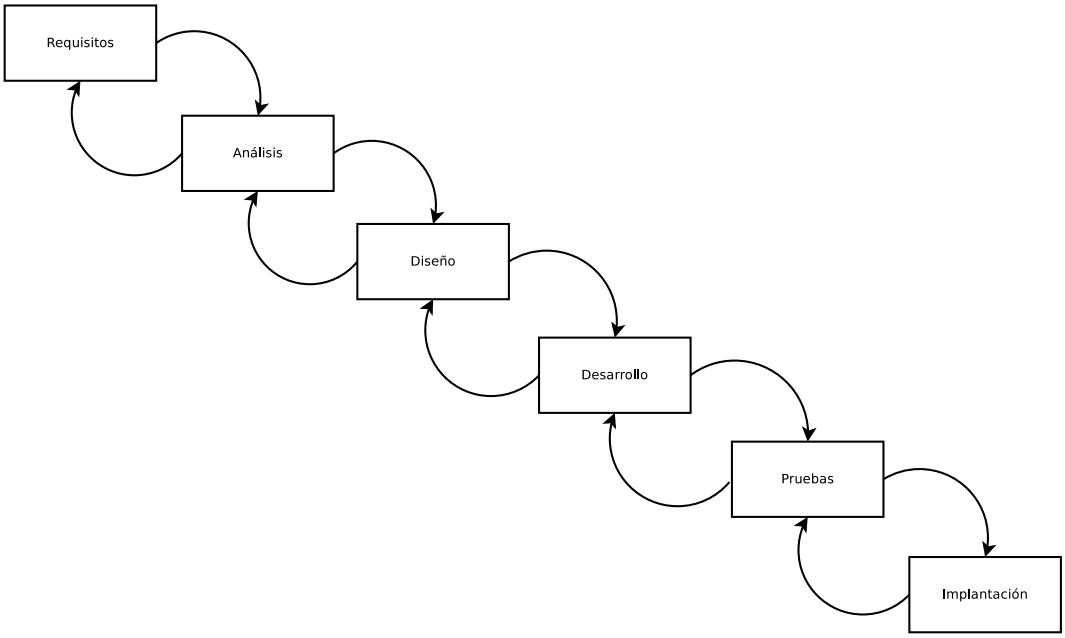

Figura 5.1.: Modelo en cascada retroalimentado

En la actualidad sigue siendo uno de los métodos más utilizado por los desarrolladores, debido a su facilidad para aplicarlo a casi cualquier proyecto y gracias a que las etapas se pueden adaptar a las necesidades de cada proyecto.

Este modelo en cascada se ha adaptado a nuestro proyecto para que cubrir nuestras necesidades:

- **Análisis de requerimientos.**

  Al igual que en cualquier otro proyecto, sea del tipo que sea, una de las primeras cuestiones o etapas que se debe realizar donde se realizar la captura de necesidades y el análisis de los requisitos del proyecto, y el análisis de cada uno de ellos. De esta forma el cliente, en nuestro caso el colectivo Comunes, realizar la enumeración de cada requisito y ex-

pone los objetivos que debe cumplir el programa con el mayor detalle posible, para poder así establecer el alcance del proyecto de una manera consensuada entre ambas partes, y de esta forma poder continuar con la siguiente fase. Es indispensable que los requerimientos queden especificados en esta fase y no se cambien durante el resto del proceso, aunque si surgen variaciones siempre se puede volver a esta fase para realizar el reajuste pertinente.

- **Análisis de riesgos.**

  Una vez especificados y analizados los requerimientos dados por el cliente y aclaradas todas las posibles dudas al respecto, se procede a estudiar y analizar los posibles riesgos que puedan existir a lo largo del proyecto, con el objetivo de establecer una o varias soluciones para reducir o eliminar esos riesgos. En esta fase además se realiza una planificación del tiempo que llevará cada fase posterior y la viabilidad de las mismas. Una vez analizado esto podemos proceder al diseño de la aplicación de acuerdo a los requerimientos establecidos en la anterior fase y teniendo en cuenta los riesgos surgidos en esta.

- **Diseño del sistema.**

  En esta etapa se realizan una serie de diagramas y estudios para obtener la estructura global del sistema y la especificación de lo que debe hacer cada una de sus partes, así como la forma de relacionarse que tiene cada una de ellas. Es muy importante identificar cada componente del sistema que se adecue a la estructura del proyecto y cada uno de los elementos que deben interactuar en el mismo. Además en este apartado se realiza el diseño de los prototipos de las pantallas que debe tener el programa, que nos permitirán hacernos una idea visual y además nos facilitará la labor de identificar la transición entre las mismas.

  Este es el paso previo al desarrollo, por tanto se debe analizar cada solución expuesta en este punto desgranando lo máximo posible para facilitar el desarrollo y los comportamientos que debe tener la aplicación.

- **Desarrollo.**

  Cuando se dispone del diseño inicial del sistema, comienza la fase de implementación o desarrollo de la aplicación, adaptando los requisitos del cliente en funcionalidades del

programa que deben cubrir esos objetivos. Aunque el tiempo empleado en cada fase depende de las dimensiones del proyecto, en proporción esta es la parte que más tiempo requiere de todo el proceso, ya que se trata de transformar todas las ideas, los prototipos y los diagramas creados anteriormente a código fuente funcional que realice las acciones necesarias para aportar la funcionalidad que el cliente espera en la herramienta.

- **Verificación.**
  Una vez finalizado el desarrollo se debe verificar o testear toda la aplicación completamente, asegurando de esta manera que funciona correctamente y que cumple con cada uno de los requerimientos especificados en la primera etapa por el cliente. Para ello es necesario disponer de un entorno de pruebas, donde se pueda realizar una simulación completa y realista de la aplicación. En esta fase es importante involucrar al cliente para que pueda realizar su propio flujo de pruebas y confirmar que el software funciona correctamente.

  Es una etapa con gran importancia, ya que permite detectar fallos de desarrollo y/o diseño y de esta forma permite realizar ajustes para solventar esos fallos.

- **Implantación.**
  Como última fase del proyecto, una vez verificado completamente, queda la implantación o el despliegue de la aplicación en el entorno de producción, de forma que se pone a disposición de los usuarios para su utilización.

5.2. Seguimiento del proyecto

Como hemos mencionado en el apartado anterior se ha seguido un modelo de desarrollo que establece una serie de pasos o etapas para la realización del proyecto, por lo que a continuación se detallarán cada una de estas fases exponiendo todos las tareas realizadas para avanzar en el proyecto y conseguir el objetivo de este desarrollo.

5.2.1. Análisis de requisitos

Como se ha mencionado anteriormente, este es el primer paso del proyecto, y por tanto es donde se realiza la toma de contacto con los usuarios, en este caso con la comunidad, y donde su principal misión es obtener los requisitos que debe cumplir el nuevo software, descomponiéndolos y detallando cada uno de ellos lo máximo posible para evitar fallos en la definición del proyecto.

En este sentido, se mantuvo una reunión inicial a través del chat de la plataforma Kune con los miembros de Comunes, Samer Hassan y Vicente J. Ruiz, para comentar brevemente los objetivos principales del proyecto, donde quedó patente, que la idea principal era renovar la primera versión del gadget de Troco. Además se vio brevemente el funcionamiento de esta primera versión para conocer algunas de las características de las que dispone y apreciar alguna de las carencias que tiene esta versión. Como resultado de esta primera reunión se obtuvo la siguiente lista inicial de requerimientos para poder avanzar en el proyecto:

- Cambiar el aspecto del gadget.
- Mejorar el funcionamiento de Troco.

Tras esta toma de contacto, se evaluaron estos requerimientos y se desglosaron una serie de dudas y cuestiones para obtener un mayor grado de detalle de cada uno de ellos. A través del correo electrónico se remitieron estas dudas a los miembros de Comunes para establecer una fecha para la siguiente reunión y poder cerrar así una lista de requerimientos completa y detallada. Las cuestiones que surgieron son las siguientes:

- Especificación de los fallos y/o carencias de la primera versión de Troco.
- Definición de nuevas funcionalidades.
- Nuevo gadget independiente de Kune o integrado en la plataforma.

- Detalles sobre el código fuente de la primera versión.

- Acceso al repositorio de código de Troco.

En esta fase además se realizó una primera planificación del trabajo que se debía realizar a lo largo de todo el proceso, de forma que se identificaron las siguientes tareas:

- Análisis de la primera versión de Troco.

- Búsqueda de herramientas similares.

- Definición de requerimientos para la nueva versión del gadget.

- Realizar mockups de la nueva herramienta.

- Desarrollo del nuevo gadget.

- Pruebas.

Tras esta comunicación, se mantuvo una reunión presencial con Samer Hassan para revisar las dudas surgidas y realizar la definición detallada de los requerimientos, para poder realizar el análisis de cada uno de ellos, y poder pasar a la siguiente fase. En esta reunión se resolvieron las dudas que se han mencionado antes y que habían sido planteadas por correo electrónico, se establecieron los requerimientos principales y se las herramienta de comunicación y trabajo para favorecer la correcta progresión del proyecto.

La lista de requerimientos y herramientas que se acordó en dicha reunión es la siguiente:

1- Crear un nuevo gadget de Troco basándose en las siguientes premisas:

- La nueva herramienta debe permitir realizar diferentes tipos de intercambios entre dos usuarios de Kune, tanto intercambios directos, como intercambios temporales, además se

debe plantear el sistemas de la forma más flexible posible para que en un futuro se pueda ampliar con un sistema de pujas, intercambios múltiples o en cadena, karma, etc.

- Esta herramienta debe permitir ofrecer el intercambio de productos y/o servicios, estableciendo cada usuario las condiciones del intercambio, como los bienes que se desea cambiar el emisor, los valores de retorno que establece el receptor como "precio" o "pago" por lo ofrecido por el emisor y permitiendo definir los distintos términos del intercambio (fecha de fin del intercambio, descripción de los valores que se intercambian, añadir fotografías de los productos ofrecidos, etc.).

- Para cada intercambio se debe poder especificar una descripción breve, que explique de forma rápida en qué consiste el bien o servicio ofrecido, una descripción más detallada, que aporte mayor información sobre lo que se quiere intercambiar y una fotografía o imagen que permita tener una visualización del bien o el servicio que se está intentando cambiar con otros usuarios.

- Posibilidad de personalizar el gadget, de forma que ofrezca un aspecto distinto para aquellos proyectos o colectivos que lo quieran utilizar de una forma personalizada.

- Internacionalización, el gadget debe permitir la configuración en diferentes idiomas, de forma que sea accesible para diferentes tipos de usuarios.

- Ayuda en línea, debe contener un tutorial que permita a los usuarios disponer de una ayuda rápida en la propia herramienta.

2- Crear un nuevo repositorio de código en Git, para almacenar todos los recursos del proyecto, desde los prototipos hasta el código fuente que se genere.

3- Utilizar la plataforma Kune como herramienta para la comunicación del proyecto, creando un proyecto para la nueva versión de Troco e incluyendo la información generada y recopilada, y por supuesto agregando a los participantes del proyecto. De esta forma conseguimos establecer un único canal de comunicación y definir un espacio de trabajo colaborativo online para el

nuevo gadget de Troco.

5.2.2. Análisis de riesgos

Una vez establecidos, aclarados y analizados los requerimientos del desarrollo pasamos a realizar una evaluación de los posibles riesgos y problemas que pueden surgir a lo largo del proceso, así como indicar las soluciones para paliar cada uno de estos riesgos.

La lista de riesgos detectada en esta fase, teniendo en cuenta los requerimientos definidos para el proyecto, es la siguiente:

- **Gran alcance del proyecto.**
  Uno de los mayores riesgos del proyecto es que tiene un alcance bastante amplio, ya que se trata de realizar una herramienta de gran envergadura y ambiciosa, que permita cualquier tipo de intercambio, integrada sobre un sistema que está aún en desarrollo y que además pueda ampliar su ámbito de aplicación en un futuro.

  La única forma de paliar este problemas es acotar mi participación en la fase de desarrollo del nuevo gadget de Troco, de forma que aporte código en alguna de las fases de desarrollo sin tener como objetivo acabar la herramienta por completo.

- **Larga duración del proyecto.**
  Como consecuencia del gran alcance del proyecto, y debido a que el tiempo disponible para la realización del proyecto de fin de Máster es finito, los tiempos del proyecto pueden exceder los tiempos marcados, no consiguiendo finalizar el desarrollo del nuevo gadget.

  Como en el riesgo anterior la solución parte del mismo concepto, es decir, acotar la participación de forma que esté involucrado en todas las fases de definición y en aquellas de implementación que sea posible.

- **Equipo de desarrollo no determinado.**

  En principio este desarrollo estaba planteado para más de una persona. Al ser un desarrollo de software libre no existe un equipo determinado y cerrado de personas que puedan aportar en cada una de las fases, sino que pueden colaborar voluntarios, pero también existe el riesgo de que no se disponga de otros colaboradores.

  La solución de este posible riesgo es complicada, ya que no se puede obligar a nadie a que colabore en el proyecto. Por tanto, como los riesgos anteriores sólo nos queda acotar mi participación hasta donde sea posible teniendo en cuenta el tiempo y los recursos.

- **Poca experiencia en GWT.**

  Uno de mis objetivos personales es mejorar mis conocimientos el framework de GWT, lo que a su vez se convierte en un riesgo para el proyecto debido a mi poca experiencia con esta librería y con las herramientas de desarrollo utilizadas para el proyecto, lo que puede provocar fallos de diseño, implementación y por tanto retrasos en el desarrollo del nuevo gadget de Troco.

  Para este riesgo la solución pasa por mejorar y aumentar mis conocimientos en estas herramientas y apoyarme en el resto de desarrolladores de la comunidad para que me ayuden con cualquier duda o problema que me surja.

5.2.3. Diseño del sistema

Esta fase del proyecto tiene gran relevancia, ya que es el paso en el cual se especificarán y se detallarán cada uno de los procesos del sistema, además se realizarán los prototipos del nuevo gadget de Troco y el modelo de base de datos para la persistencia, que marcarán la base para la fase de desarrollo.

En primer lugar se realizó la primera versión del diseño del modelo de datos para el nuevo gadget, de forma que dé cabida a todos los requerimientos que se han mencionado anteriormente.

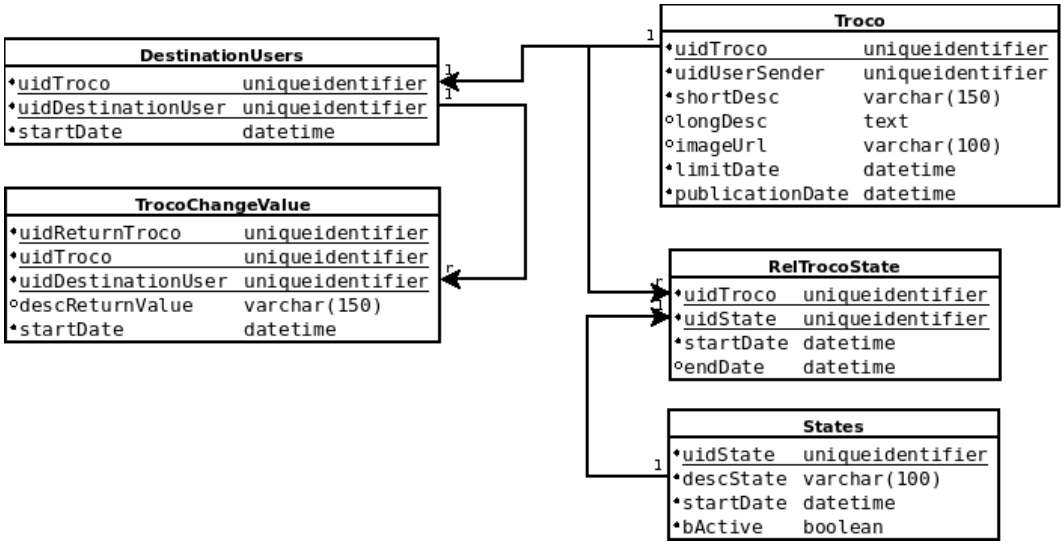

Figura 5.2.: Primera versión diseño del modelo de datos

Como podemos ver en la figura 5.2, se han definido las siguientes tablas:

- **Troco**

  Ésta es la tabla maestra de los Trocos o intercambios, donde se almacenarán cada uno de los datos relacionados con cada intercambios que se inicie en la plataforma. Los datos que quedarán almacenados para cada Troco son los siguientes:

  - Un identificador único para cada intercambio, de forma que lo referencie unívocamente en el sistema. Este campo será la clave primaria de esta tabla.

  - Un identificador de usuario emisor del intercambio, que identifique al usuario de Kune que inicia un Troco en la plataforma.

  - Una descripción breve del intercambio, con el objetivo de indicar de forma rápida que tipo de bien o servicio se desea intercambiar.

  - De manera opcional, será posible establecer una descripción más completa del producto o servicio que se ofrece en el intercambio.

- Una referencia a una imagen, que proporcione una visión del producto que se desea intercambiar.

- Un campo para almacenar la fecha de finalización del intercambio.

- Un campo con la fecha de publicación del intercambio, como dato de control y auditoría.

- **States**

  Es una tabla maestra de estados en los que se puede encontrar en Troco. Los campos son los siguientes:

  - Un identificador para cada estado.

  - Un campo con la descripción del estado.

  - Una campo con la fecha de alta de dicho estado.

  - Un campo para activar o desactivar un estado determinado.

- **DestinationUsers**

  En esta tabla se almacenará la información de cada usuario que es el receptor del intercambio. Los campos de los que dispone son:

  - El identificador del Troco.

  - El identificador del usuario que será el destinatario.

  - Un campo de auditoría con la fecha de alta.

- **TrocoChangeValue**

  Esta tabla almacenará la información del producto o servicio que ofrece el usuario recep-

tor del intercambio al usuario emisor. Los campos que tiene esta tabla son los siguientes:

- Un identificador del valor de vuelta.
- Un identificador del Troco asociado.
- El usuario que remite la oferta de vuelta.
- La descripción del valor devuelto.
- La fecha de alta de la oferta.

- **RelTrocoState**
  Esta tabla es la resultante de relacionar los Trocos con los diferentes estados por los que puede pasar, de forma que se consiga almacenar un histórico de los estados por los que pasa un intercambio. Sus campos son los siguientes:

  - El identificador del Troco.
  - El identificador del estado de la tabla States.
  - La fecha de inicio del estado.
  - La fecha final del estado.

Este modelo de datos tendrá que revisarse cuando se llegue a la fase de desarrollo, debido a que pueden surgir cambios que requieran almacenar nuevos campos o diferente información. Es un modelo ampliable a otro tipo de intercambios, ya que con una tabla nueva se podría ampliar para un intercambio en cadena, reflejando el usuario que emite el intercambio, el que lo recibe inicialmente y a que otro u otros usuarios se deriva el intercambio para finalizarlo.

Uno de los objetivos del proyecto es comenzar un nuevo repositorio de código, con los fuentes

del nuevo gadget de Troco. Por este motivo se creó un nuevo proyecto llamado troco en la plataforma web Gitorious y se otorgaron permisos de acceso y modificación a cada uno de los miembros que estaban involucrados en el proyecto.

Tras crear el repositorio de código, se planteó un diagrama de estados del intercambio básico de un bien y/o un servicio con otro usuario, de forma que se puede establecer un estado determinado para cada etapa del intercambio. Pasando desde el estado inicial hasta el estado final donde el intercambio quedará completado, ya sea debido a que se ha llegado a un acuerdo entre los usuario o bien porque se ha cancelado debido a que no se ha encontrado otro usuario con el que realizar el intercambio. Podemos apreciar estos estados en la figura 5.3:

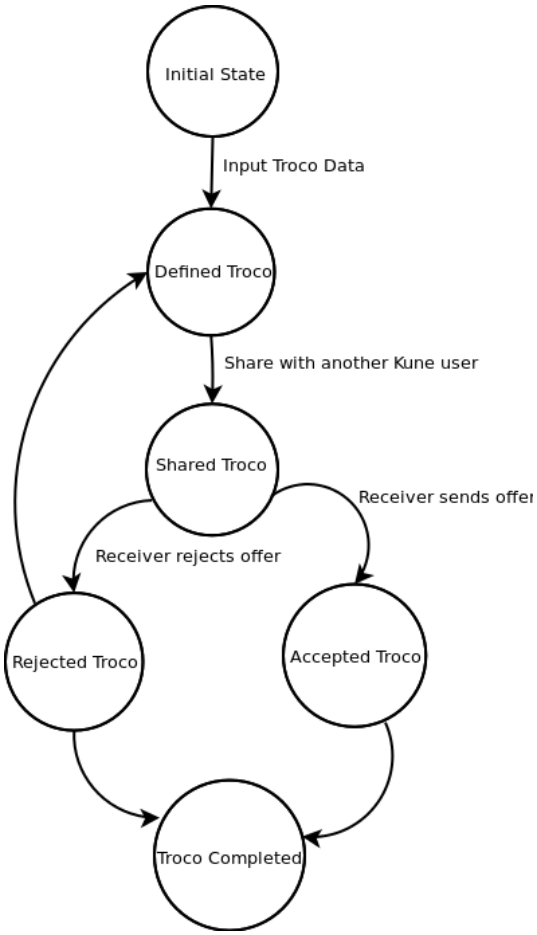

Figura 5.3.: Diagrama de estados Troco básico

Además se realizaron unos mockups de las pantallas de la aplicación nueva, basados en los requerimientos, el diagrama de estados y sobre todo centrados en definir el flujo completo de

un intercambio simple entre dos usuarios, como punto de partida para el desarrollo.

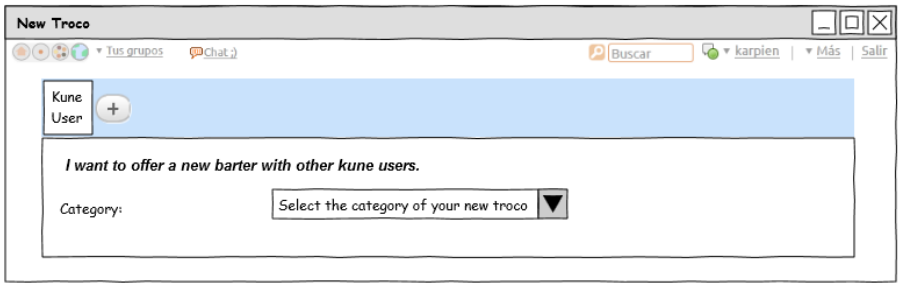

Figura 5.4.: Prototipo nuevo intercambio paso 1

Como podemos apreciar en la figura 5.4, la idea es que inicialmente al crear el intercambio se seleccione la categoría del intercambio. Al categorizar los intercambios nos permite, en un futuro, poder ampliar la herramienta para establecer un buscador de intercambios que permita a los usuarios de Kune realizar búsquedas más específicas de diferentes tipos de conceptos y servicios, posibilitando así encontrar de una forma sencilla y rápida aquello que desean obtener o intercambiar.

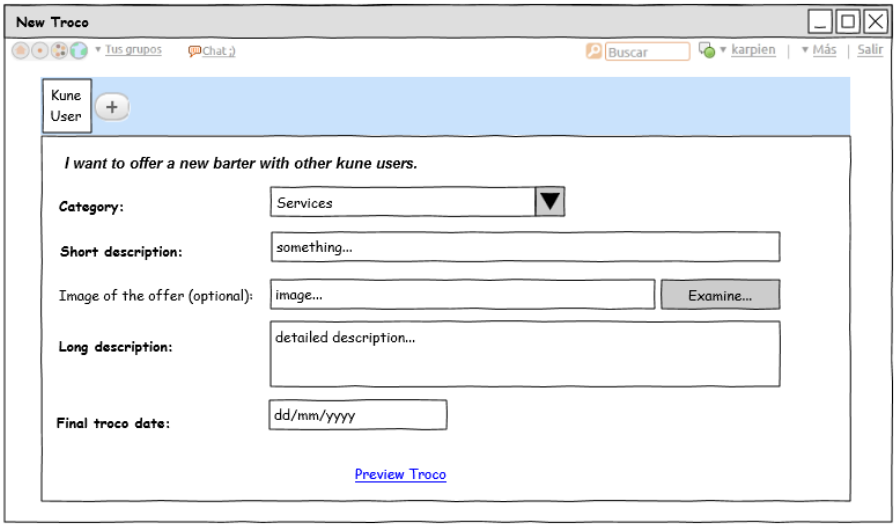

Figura 5.5.: Prototipo nuevo intercambio paso 2

Una vez seleccionada la categoría será posible establecer tanto una descripción breve del intercambio, como una imagen que dé una visión del producto o servicio que se desea ofrecer, como un texto más amplio que proporcione mayor detalle y por supuesto una fecha que

defina hasta cuándo estará activo el intercambio. Será posible visualizar el Troco en cualquier momento.

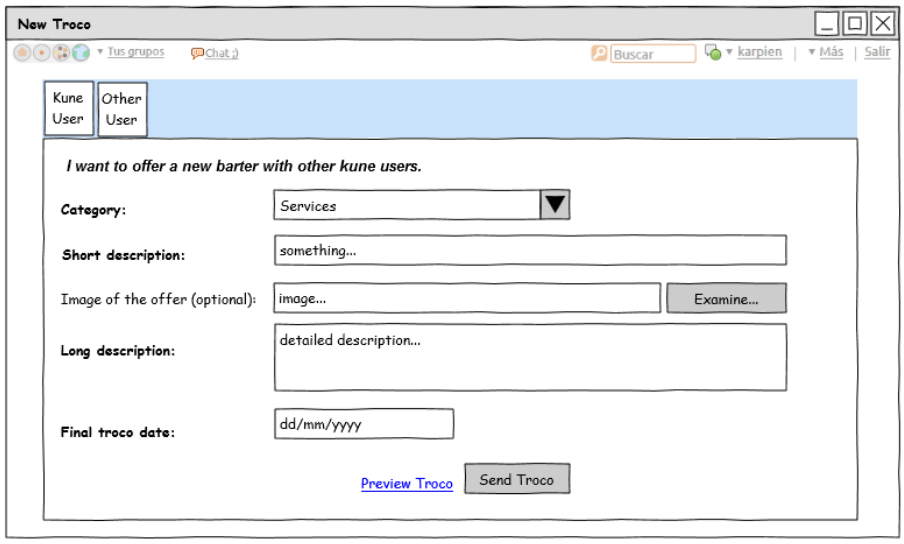

Figura 5.6.: Prototipo nuevo intercambio paso 3

Una vez cumplimentados todos los datos y añadido el usuario con el que se desea realizar el intercambio se podrá enviar el Troco, pasando al estado compartido.

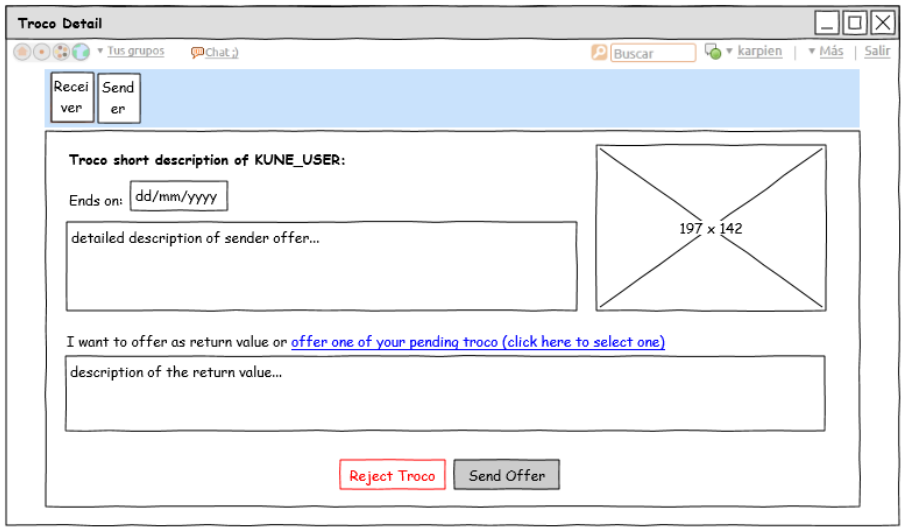

Figura 5.7.: Detalle del intercambio

Una vez compartido el intercambio con el otro usuario, éste podrá ver la información introducida por parte del emisor de Troco y realizar una oferta con el valor de retorno que ofrece

para fijar un intercambio o bien puede rechazar el intercambio.

Tras remitir los prototipos y subirlos al repositorio, se revisaron por parte de todos y se determinó que además de la funcionalidad básica de intercambios entre dos usuarios, sería recomendable realizar un asistente para los intercambios con el objetivo de facilitar cada tipo de intercambio a los usuarios y guiarlos de forma que sea sencillo para ellos realizar el intercambio que realmente necesitan.

En este sentido, a partir de ese momento, se partiría de esta premisa como parte fundamental del desarrollo, de forma que las diferentes pantallas y opciones del asistente permitan establecer el estado del intercambio y los valores para realizar el cambio, y además los componentes desarrollados para desgranar cada una de las opciones del asistente se puedan reutilizar en la pantalla de intercambio básico.

A priori, este cambio no altera en exceso la planificación y el flujo del proyecto, ya que es el mismo funcionamiento pero desglosado en pasos, que permitan guiar al usuario hasta conseguir crear el Troco. Por tanto las fases más afectadas son la fase en la que estamos y sobre todo la implementación de la herramienta, que es la siguiente fase que abordaremos.

Aunque sería recomendable realizar un cambio en los prototipos explicados anteriormente, otra de las cosas que se decidió en conjunto con Comunes, fue comenzar el desarrollo del asistente de Troco como parte de la nueva herramienta. Por tanto se comenzó con el diseño de nuevo de los estados por los que puede pasar el Troco y la definición mediante una máquina de estados de cada una de las funcionalidades del gadget.

A continuación se puede apreciar en la figura 5.8 la máquina de estados generada para continuar el proyecto:

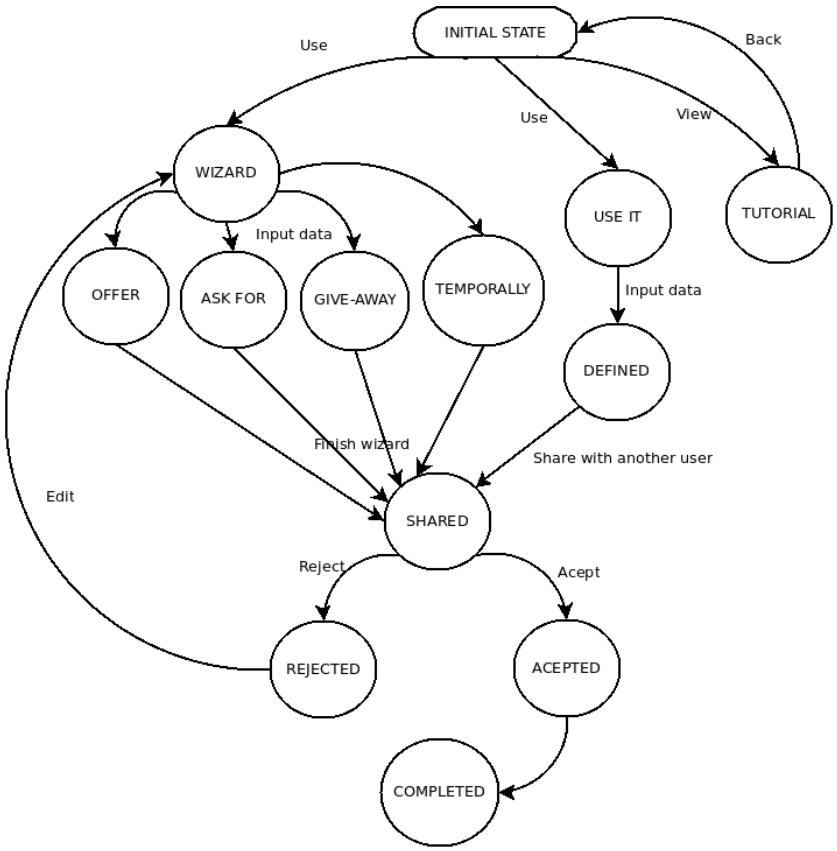

Figura 5.8.: Máquina de estados para Troco

Además se especificaron los diálogos de las nuevas pantallas, tanto para el wizard como para el intercambio directo, que se detallan a continuación:

- **Initial Screen:**

  (configurable logo)

  - Use the wizard
  Use the wizard if it's the first time you use troco or you need some help using it.

  - or directly start to use it and share
  Perfect if you know how this system works and just wanna start to share goods and services.

- or see our tutorial

The tutorial will show you how troco works.

- **Wizard:**

(configurable intro)

So, let's start, what do you want to do?

- you want to offer some service or good

Appropiate if you want to offer/shell/provide a material good (e.g. a sofa, a laptop) or a service from your skills (e.g. painting a house, translation), and receive something in exchange.

- you want to ask for some service or good

Appropiate if you need a material good (e.g. clothes, organic vegetable) or a service (e.g. doctor, guitar lessons, some translation) and you would give something in exchange.

- you want to give-away some good for free

Appropiate if you would like to give away some of your stuff (e.g. clothes, a fridge, furniture, etc) or offer your skills for free (e.g. walking the dogs, make some haircut practices, make some shiatsu practices).

- you want to share some good temporally

Ideal if you want to share books/music/seeds/your flat/ your car, with people during some time but you want them back after an agreed period or time.

- **Wizard (Offer 1/2): What do you want to offer?**

- What do you want to offer in few words?

[example: Some sofa, bicycle, furniture, etc.]

- Now give people a full description if you want.

[example: I give my bicycle, a model XXX, because I have a new one and no space to both]

-“A picture speaks a thousand words” Do you have some photo of that?
[Upload]

Next

- **Wizard (Offer 2/2): What do you want in exchange?**

Options:

[X] I want some other stuff [example: barter with some service, or 5€plus delivery]

[X] I want to bid, starting with [X] [currency] til [date]

Finish

- **Wizard (Ask for 1/2): What do you want to ask for?**

- What do you want ask for in few words?
[example: some furniture, or some seeds, or some service (translations), etc.]

- Now give people a full description if you want.
[example: I want get some seeds of tomatoes of type cherry]

- “A picture speaks a thousand words” Do you have some photo of that?

Next

- **Wizard (Ask for 2/2): What do you want to ask for?**

- When do you want to do the exchange?

Finish

- **Wizard (Give-away): What do you want to give away?**

  - What do you want give away in few words?
  [example: some furniture, or some seeds, or some service (translations), etc.]

  - Now give people a full description if you want.
  [example: I want get some seeds of tomatoes of type cherry]

  Finish

- **Wizard (temporally): What do you want to share temporally?**

  - What do you want share temporally in few words?
  [example: books, music, etc.]

  - Now give people a full description if you want.
  [example: I want share temporally my book “The Dispossessed”]

  Finish

- **Use it:**

  (configurable intro) Select type of exchange:

  - Options:

    - [X] offer or give away something.

    - Short description.
    - Full description (optional).
    - Images (optional).
    - What do you want in exchange?
        - [x] The same value or service.
        - [x] Another barter (in future).
        - [x] Other value or service.
        - [x] Nothing (only for give-away option).
    - Finish date.
- [X] ask for something.
    - Short description about you want to find.
    - Full description (optional).
    - Images (optional).
    - What do you offer in exchange?
        - [x] The same good or service.
        - [x] Another good or service.

        - ◇ [x] Other barter.
        - ◇ [x] Nothing.
    - Estimated date of exchange.
- [X] share something.
    - Short description about you want to share.
    - Full description (optional).
    - Images (optional).
    - Initial date.
    - Final date.

5.2.4. Desarrollo

Esta etapa es la fase más larga de todo el proyecto, donde se realiza la implementación de la herramienta partiendo de los requerimientos y análisis realizados anteriormente, utilizando cada una de las herramientas que se han mencionado en el capítulo 2.

Por una cuestión de tiempos, la parte en la que me he visto involucrado dentro de esta fase del proyecto es en la de la implementación y creación de la capa de interfaz de usuario del nuevo gadget de Troco, donde se ha desarrollado las principales opciones del wizard del gadget. Para esta parte se ha utilizando la librería para la creación de formularios y validaciones GWT Pectin.

Utilizando una estructura básica realizada por Vicente J. Ruiz de Comunes, se fueron añadi-

endo cada una de las pantallas principales del wizard de Troco, incluyendo cada una de las clases que son necesarias para utilizar la librería de validación que se ha mencionado anteriormente.

Uno de los primeros problemas que me encontré en esta fase fue desplegar los gadgets para poder testearlos y ver el resultado final. Desde el IDE de desarrollo no es posible desplegar estos directamente, lo que dificulta en gran medida la implementación de las interfaces de usuario.

Dentro de la estructura iniciada por Vicente J. Ruiz se encuentran una serie de clases que utilizan para la validación y el despliegue de gadgets creados en GWT para la plataforma Kune. Estas clases heredan de las clases de Kune que internamente utilizan la librería GIN (GWT INjection) que se utiliza para la inyección automática de dependencias de código GWT en el cliente. De esta forma se puede desplegar desde el propio entorno de desarrollo el código del gadget permitiendo así realizar pruebas y comprobar en caliente el funcionamiento del mismo.

En esta fase se han creado una serie de clases para la creación de los estados de la aplicación, tanto para definir en sí las pantallas como para indicar el estado en el que se encuentra el gadget. A continuación se listan cada uno de los conjuntos de clases que se han creado para la implementación del gadget:

- Clases para los estados, donde se definirán los comportamientos que debe tener la nueva versión del gadget para cada uno de los estados en los que se puede encontrar a lo largo de su utilización, como por ejemplo la pantalla inicial, el wizard, el tutorial, etc. Además se incluyen las clases que definen el modelo y la presentación del gadget según las especificaciones de la librería gwt-pectin.

- Clases del cliente, donde están las clases que interactúan con la librería GIN y donde se especifican la configuración básica de la herramienta y las constantes utilizadas en el resto de clases.

- Clases de internacionalización, para la traducción del gadget a diferentes lenguajes, inicialmente a inglés y español, pero de esta forma es posible extenderlo a cualquier otro idioma.

- Clases del modelo, donde se definen las clases para las estructuras de datos utilizadas en la capa del modelo de la aplicación.

- Clases para las acciones, donde se definen las clases de la botonera que permitirá la transición entre las diferentes opciones del gadget y las distintas acciones que se pueden realizar en cada una de las pantallas.

- Clases para la gestión, donde se implementan las clases con los managers de las acciones, la máquina de estados, etc.

- Clases de la interfaz de usuario, donde se encuentran las clases que definen la estructura básica de las pantallas, como la cabecera, el pie, la barra de botones, etc. Además se definen las clases que se utilizan para los distintos efectos de visualización de la herramienta.

- Clases útiles, que implementa clases que proporcionan utilidades necesarias en el resto de clases de la herramienta, como son clases para el tratamiento de fechas, herramientas para la internacionalización, utilidades para controles de formularios, etc.

- Clases para el testing y el despliegue, que como hemos comentado antes han sido desarrolladas para poder probar el gadget desde el entorno de desarrollo.

Como veremos más adelante en la sección 5.3, estas clases implementan las funcionalidades básicas para poder realizar las pantallas principales del wizard de Troco, mostrando cada una de las posibilidades de intercambio que permite esta herramienta y donde se generan una serie de componentes que será posible reutilizar para la opción directa de intercambio para usuarios avanzados, utilizando la librería GWT pectin. También permiten realizar algunas configuraciones específicas del gadget, como un logotipo personalizado o un nombre diferente más orientado a las necesidades de cada proyecto o colectivo, y realiza de forma sencilla la adaptación a diferentes idiomas y que de esta forma sea más accesible a cualquier tipo de usuario y mayor número de personas.

5.2.5. Verificación

Esta fase es fundamental para la validación y prueba de la aplicación, de forma que se comprueba que cada una de las funcionalidades implementadas en la fase anterior realiza las acciones y flujos que se especificaron al comienzo del desarrollo.

Debido a los problemas de tiempo, no se ha finalizado el desarrollo de la nueva versión del gadget de Troco, por tanto no se ha realizado la fase de verificación de la herramienta. Esta fase de pruebas será realizada al final del desarrollo, con el objetivo de confirmar el correcto funcionamiento del gadget para cada una de sus opciones, y sobre todo para corregir aquellos fallos y errores que se encuentren en esta fase, realizando los reajustes en la implementación que sea necesario.

Habitualmente para esta fase se suele generar un documento de pruebas que sirva de guía para probar la aplicación, en el cual se indican cada uno de los casos de prueba que se deben realizar para verificar todas las funcionalidades que están incluidas en la nueva versión del gadget de Troco.

5.2.6. Implantación

Ésta es la última de las fases del proyecto, donde una vez finalizado el desarrollo de la nueva aplicación y verificadas cada una de las funcionalidades de la herramienta, se realiza una subida al servidor de aplicaciones de producción, de forma que desde ese momento los usuarios puedan utilizar el nuevo gadget sobre la plataforma Kune para realizar el intercambio de bienes y servicios de una forma rápida y sencilla.

5.3. Detalle Técnico

En este apartado se expondrán la información sobre las clases y paquetes creados en la parte del desarrollo que se ha realizado hasta el momento. Este diagrama permite conocer las agrupaciones lógicas de clases y las dependencias entre éstas al más alto nivel.

En la figura 5.9, se pueden apreciar de forma global todos los paquetes que se han generado, así como las dependencias con librerías externas que se han utilizado para la implementación del nuevo gadget de Troco:

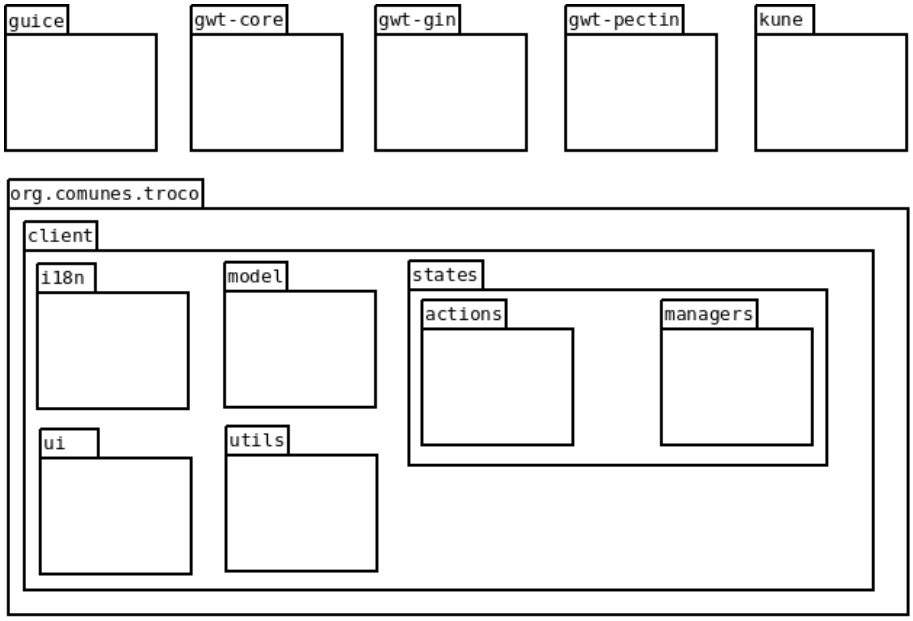

Figura 5.9.: Diagrama completo de clases

A continuación, vamos a mostrar el diseño de clases y los ficheros generados en el proyecto, separado en los diferentes paquetes en los que se ha organizado la aplicación.

5.3.1. org.comunes.troco

Éste es el paquete general de la aplicación, donde se definen los elementos estáticos de la aplicación, como son las imágenes, la hoja de estilos, los ficheros JavaScript, etc, y el fichero de

módulo de GWT, que es un fichero XML donde se define la configuración del módulo a través de la especificación de varios elementos, como son:

- <**inherits name="logical-module-name" /**>
  Hereda las configuraciones desde el módulo especificado en el atributo name, en nuestro caso se añaden las siguientes librerías para GWT: i18n, mvp y pectin, así como la librería anteriormente comentada pectin para la validación de formularios.

- <**entry-point class="classname" /**>
  Se especifica la clase inicial a la aplicación. En nuestro caso no es necesario especificar este punto de entrada, ya que esta se utiliza o define en el módulo de despliegue.

- <**source path="path" /**>
  Especifica el path de las clases principales del módulo, en el caso de nuestra herramienta se indica el subpaquete "client" que es donde se implementan cada una de las clases principales del nuevo gadget de Troco.

- <**public path="path" /**>
  Este elemento añade paquetes al path público para ser referenciados en el módulo, cualquier archivo que aparece en este paquete o subpaquetes será tratado como un recurso de acceso público. En nuestro caso se utiliza la configuración por defecto.

- <**servlet path="url-path" class="classname" /**>
  Este elemento carga la clase servlet indicada, para el uso de RPC. En el proyecto no se han utilizado servlets.

- <**script src="js-url" /**>
  Inyecta los archivos JavaScript externos especificados en el atributo src, como pueden ser librerías de JQuery[1], extJs[2], etc. En nuestro caso no se han añadido ficheros JavaScript externos a la aplicación.

[1] http://jquery.com/
[2] http://www.sencha.com/products/extjs

- <**stylesheet src="css-url" /**>

  Añade los ficheros de hojas de estilo especificadas. Para Troco se ha definido una nueva hoja de estilos donde se especifican cada uno de los estilos que se aplicarán a los diferentes elementos del gadget.

- <**extend-property name="client-property-name" values="separated-values" /**> Permite establecer un conjunto de valores que pueden ser utilizados como propiedades en el gadget. En el caso de Troco de esta forma se han especificado los distintos idiomas en los que se puede configurar la herramienta, inicialmente en inglés y español.

5.3.2. org.comunes.troco.client

Este paquete es el paquete principal de la aplicación, donde se definen las clases para la interfaz gráfica de usuario y de donde cuelgan cada uno de los subpaquetes donde se definen los estados de la aplicación, los modelos de cada estado, la configuración, etc.

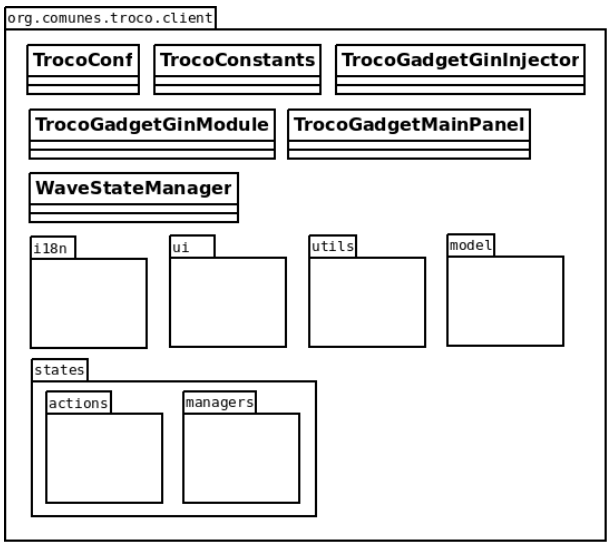

Figura 5.10.: Clases del paquete org.comunes.troco.client

En la figura 5.10 podemos apreciar la estructura de este paquete, las clases de las que está compuesto y los subpaquetes que lo forman.

- **TrocoConf:** es la clase que se utiliza para leer toda la configuración del gadget de Troco, desde el lenguaje por defecto hasta los elementos personalizables del mismo, como son los logotipos y el nombre de la herramienta. Obtiene la configuración del fichero *trocoConf.js*, que se encuentra en la carpeta de elementos públicos del proyecto.

- **TrocoConstants:** esta clase publica define una serie de constantes que son utilizadas a lo largo de la aplicación.

- **TrocoGadgetGinInjector:** define una interfaz donde se definen los métodos necesarios para inyectar cada una de las módulos de presentación de la aplicación. Define todos los métodos necesarios para obtener las vistas de cada uno de los componentes del gadget. Hereda de la clase *KuneGadgetGinInjector*, que tiene ese propósito en la plataforma Kune.

- **TrocoGadgetGinModule:** es una clase que enlaza cada una de las acciones tanto de Kune como del gadget, así como la secuencia entre los elementos de presentación del gadget.

- **TrocoGadgetMainPanel:** es la clase que inicializa el gadget y presenta los paneles que se han definido. Hereda de la clase *Composite*, que es una clase de interfaz de usuario de gwt, e implementa las interfaces *IndexedPanel.ForIsWidget* y *InsertPanel.ForIsWidget*, que proporcionan utilidades necesarias para insertar gadgets, obtenerlos y ejecutarlos.

- **WaveStateManager:** es la clase donde se define el manager de la Wave y las funciones para interactuar con esta.

5.3.3. org.comunes.troco.client.i18n

Este subpaquete del paquete client, es donde se definen las clases necesarias para la gestión de la internacionalización en el gadget. Además contiene dos ficheros de propiedades para la configuración de los textos de toda la aplicación en ambos idiomas.

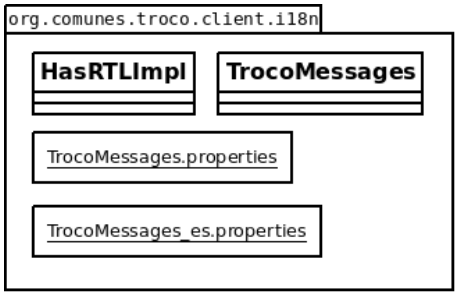

Figura 5.11.: Subpaquete org.comunes.troco.client.i18n

- **HasRTLimpl:** esta clase permite conocer si el lenguaje utilizado es de tipo "Right to Left" (RTL). Hereda de la clase *HasRTL* de la librería de Kune.

- **TrocoMessages:** esta interfaz define los métodos de acceso a cada uno de los mensajes y textos que se muestran a lo largo de las pantallas del gadget, teniendo en cuenta la configuración del idioma que se está utilizando. Esta interfaz hereda de la interfaz *Messages* de la librería para internacionalización de GWT.

- **TrocoMessages.properties:** fichero de configuración donde se almacenan cada uno de los mensajes del gadget en el lenguaje por defecto que es el inglés, que se obtienen a través de los métodos definidos en la interfaz *TrocoMessages*.

- **TrocoMessages_es.properties:** lo mismo que el anterior pero para su traducción al castellano.

5.3.4. org.comunes.troco.client.model

En este subpaquete se especifican aquellas clases donde se definen las diferentes estructuras de datos utilizadas en las diferentes clases de la capa de modelo para almacenar y reflejar los campos que se utilizarán en la interfaz.

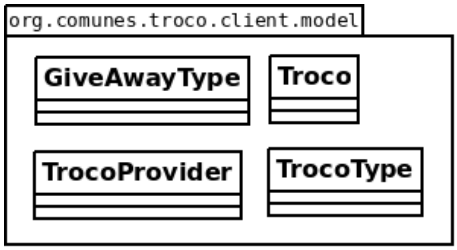

Figura 5.12.: Subpaquete org.comunes.troco.client.model

- **GiveAwayType y TrocoType:** realmente definen un tipo de estructura enum que define atributos para las diferentes opciones del asistente de Troco. Se utilizan para referenciar a los campos de tipo radio button de los formularios.

- **Troco:** en esta clase se definen cada uno de los campos que se utilizarán en los formularios del asistente, así como sus getters y setters.

- **TrocoProvider:** es una clase abstracta que hereda de la clase *BeanModelProvider* de la librería de gwt-pectin y que actúa como factoría de enlace entre el ValueModel y las propiedades de los Java Beans.

5.3.5. org.comunes.troco.client.states

En este subpaquete se definen las clases que implementan cada una de las pantallas asociadas a los distintos estados por los que puede pasar el intercambio, desde las pantallas iniciales donde se muestran las opciones básicas (wizard, tutorial y use it) y cada una de las opciones del wizard. La estructura de estas clases viene definida por los patrones utilizados en la librería gwt-pectin, donde se define el modelo del formulario, otra clase donde se enlazan los componentes visuales y otra clase para la capa de presentación donde se recuperan los datos del modelo y los formatea para su visualización. A continuación, en la figura 5.13 se muestran las clases y subpaquetes contenidos en este paquete:

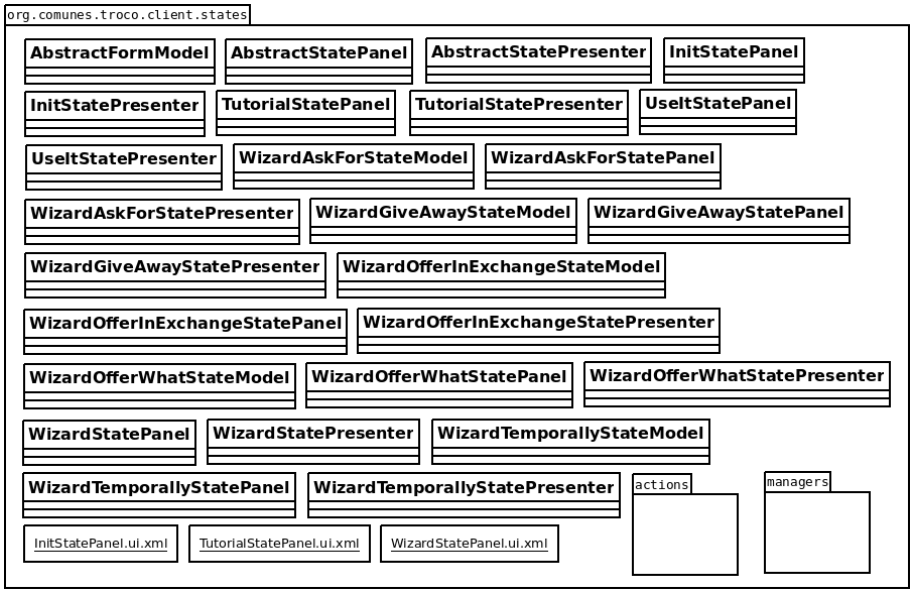

Figura 5.13.: Subpaquete org.comunes.troco.client.states

- **AbstractFormModel:** esta clase abstracta extiende de la clase *FormModel*, de la librería gwt-pectin, donde se definen los diferentes tipos de campos que serán utilizados en todos los modelos de cada uno de los formularios de la herramienta.

- **AbstractStatePanel:** en esta clase se definen los campos básicos que se enlazan con la vista del formulario, así como la estructura para visualizar los formularios en el gadget. Hereda de la clase *VerySimpleForm*, donde se implementan los métodos necesarios para componer la capa de interfaz de usuario.

- **AbstractStatePresenter:** esta clase abstracta es la que define los métodos básicos del componente de la capa de presentación, donde se enlazan los datos del modelo con las vistas de los formularios. Hereda de la clase *Presenter*, que es parte de la implementación del patrón MVP de GWT.

- **InitStatePanel, TutorialStatePanel y WizardStatePanel:** en estas clases es donde se implementa la vista de los distintos formularios del gadget, que sólo incluyen contenido estático y por tanto no requieren de un modelo. Estas clases instancian un objeto de tipo UiBinder, que permite construir aplicaciones como páginas html a través de widgets de GWT, separando la lógica de programación de la interfaz de usuario.

- **InitStatePresenter,TutorialStatePresenter y WizardStatePresenter:** en estas clases se definen los elementos configurables para la vista asociada e implementa cada uno de los eventos que se utilizan en la capa de presentación que interactúan con los componentes de la interfaz.

- **UseItStatePanel y UseItStatePresenter:** estas clases, que se encuentran en construcción, contendrán la implementación del formulario básico de intercambio, formado por los componentes necesarios que son utilizados en las clases del Wizard.

- **WizardOfferWhatStatePresenter, WizardOfferWhatStatePanel y WizardOfferWhatStateModel:** estas clases definen tanto el modelo, como la vista, como la clase de presentación para la opción del asistente para ofrecer un bien o servicio para intercambiar. Una vez fijados los valores de lo que se ofrece y transiciona a la pantalla para definir el valor que se desea obtener a cambio, definido en las clases *WizardOfferInExchange**.

- **WizardAskForStatePresenter, WizardAskForStatePanel y WizardAskForStateModel:** al igual que las clases anteriores estas definen el modelo gwt-pectin para la opción del asistente para pedir algún bien o servicio a otro usuario.

- **WizardGiveAwayStatePresenter, WizardGiveAwayStatePanel y WizardGiveAwayStateModel:** define los campos específicos para la opción del asistente para regalar un bien a otro usuario.

- **WizardTemporallyStatePresenter, WizardTemporallyStatePanel y WizardTemporallyStateModel:** donde se define los campos específicos para la opción del asistente para intercambiar de forma temporal algún bien o servicio.

- **InitStatePanel.ui.xml, TutorialStatePanel.ui.xml y WizardStatePanel.ui.xml:** que son los ficheros para el framework UiBinder para cada una de las opciones principales del gadget y donde se define la estructura del formulario mediante widgets GWT.

5.3.6. org.comunes.troco.client.states.actions

Este subpaquete incluye las clases que definen las acciones que se pueden realizar para cada uno de los estados del nuevo gadget.

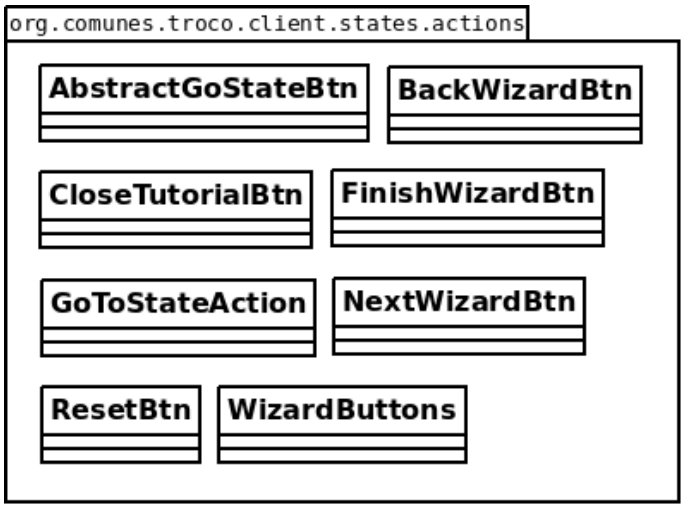

Figura 5.14.: Subpaquete org.comunes.troco.client.states.actions

- **AbstractGoStateBtn:** clase abstracta que define los métodos para transitar entre los diferentes estados.

- **BackWizardBtn:** clase que extiende de la clase abstracta *AbstractGoStateBtn*, y que define la funcionalidad del botón para volver al asistente.

- **CloseTutorialBtn:** también hereda de la clase *AbstractGoStateBtn*, y sirve para volver al estado inicial del gadget.

- **FinishWizardBtn:** al igual que las anteriores hereda de la clase abstracta y define la funcionalidad para dejar el troco en estado definido y preparado para que el usuario receptor establezca el valor de retorno.

- **GoToStateAction:** esta clase almacena un estado y transita a este estado si la acción está completada.

- **NextWizardBtn:** otra clase donde se define la acción de pasar a la siguiente pantallas a lo largo del asistente.

- **ResetBtn:** en esta clase de define el botón para resetear el gadget.

- **WizardButtons:** en esta clase se define la botonera para para el asistente del gadget.

5.3.7. org.comunes.troco.client.states.managers

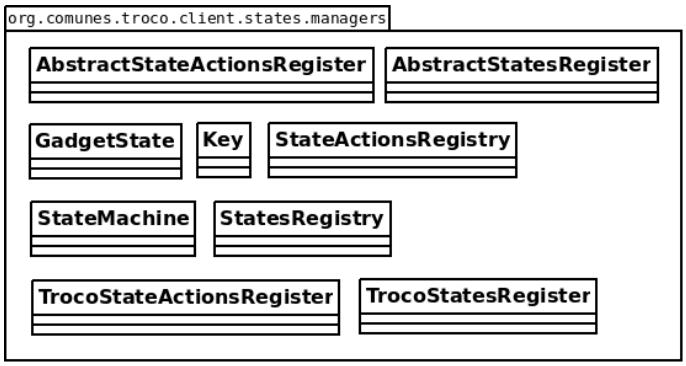

Figura 5.15.: Subpaquete org.comunes.troco.client.states.managers

- **AbstractStateActionsRegister y AbstractStatesRegister:** son las clases abstractas para el registro tanto de los estados como de las acciones asociadas a estos.

- **GadgetState:** es la interfaz que debe implementar cada uno de los estados del gadget.

- **Key:** calcula las claves de los estados.

- **StateActionsRegistry:** esta clase es un almacén de acciones por estado, que se pueden recuperar para cada cambio de estado.

- **StateMachine:** en esta clase se define la lógica de los cambios de estado, así como el guardado y la recuperación de los estados.

- **StatesRegistry:** esta clase se utiliza para registrar los diferentes estado del gadget.

- **TrocoStateActionRegister:** esta clase, que hereda de la clase abstracta *AbstractStateActionsRegister*, y es donde se enlazan las diferentes acciones con los diferentes estados.

- **TrocoStatesRegister:** en esta clase se definen cada una de los estados del gadget, hereda de la clase *AbstractStatesRegister*.

5.3.8. org.comunes.troco.client.ui

En este paquete se definen las clases para la interfaz de usuario, como la cabecera, el pie y algunas funcionalidades y efectos para los campos del formulario.

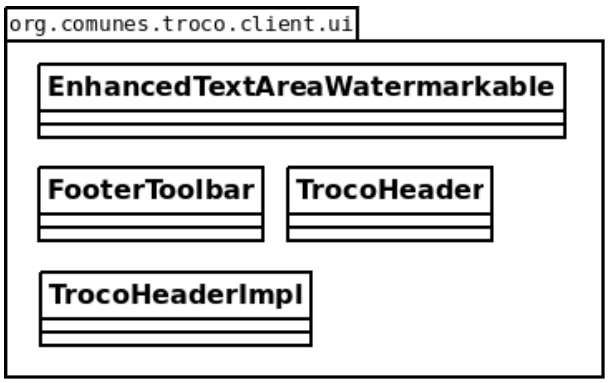

Figura 5.16.: Subpaquete org.comunes.troco.client.ui

- **EnhancedTextAreaWatermarkable:** esta clase hereda de la clase *Enhanced* e implementa la interfaz *Watermarkable*, de la librería gwt-pectin. Esta clase permite utilizar el efecto de marca de agua para los campos de formulario de tipo textArea.

- **FooterToolbar:** donde se implementa el pie del gadget.

- **TrocoHeader:** interfaz donde se definen las funciones para utilizar las distintas opciones para la cabecera del gadget.

- **TrocoHeaderImpl:** clase que implementa la interfaz *TrocoHeader* y donde se establecen las características de la cabecera del gadget de Troco.

5.3.9. org.comunes.troco.client.utils

Este paquete contiene las clases donde se definen diferentes utilidades utilizadas en el resto del proyecto. Entre estas utilidades tenemos clases para internacionalización, validaciones, etc.

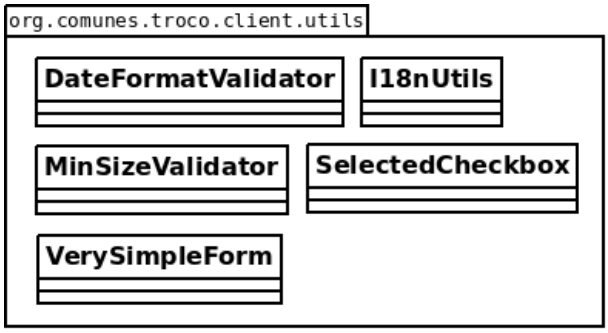

Figura 5.17.: Subpaquete org.comunes.troco.client.utils

- **DateFormatValidator:** esta clase define la validación para los campos de tipo fecha.
- **I18nUtils:** en esta clase se definen utilidades básicas para la internacionalización.
- **MinSizeValidatos:** clase para la validación del tamaño de los campos de los formularios.
- **SelectedCheckbox:** clase para la validación de campos desplegables.
- **VerySimpleForm:** clase donde se define la estructura básica de los formularios del gadget.

5.4. Licencia

Al igual que la plataforma Kune, la licencia que hemos utilizado para el código fuente de esta herramienta es AGPL v3[3], por eso todos los ficheros que se han generado en este desarrollo incorporan una cabecera que indica la licencia que estamos utilizando, que se muestra a continuación:

```
/*****************************************************************************
* Copyright (C) 2007, 2013 The kune development team (see CREDITS for details)
* This file is part of kune.
*
* This program is free software: you can redistribute it and/or modify
* it under the terms of the GNU Affero General Public License as
* published by the Free Software Foundation, either version 3 of the
* License, or (at your option) any later version.
*
* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
* GNU Affero General Public License for more details.
*
* You should have received a copy of the GNU Affero General Public License
* along with this program. If not, see <http://www.gnu.org/licenses/>.
*****************************************************************************/
```

[3] Affero General Public License version 3

6. Conclusiones

En esta sección se detallarán cada una de las conclusiones obtenidas en la realización de este proyecto FLOSS y una pequeña comparativa con el flujo de trabajo de una desarrollo privativo, mostrando las diferencias que se han podido apreciar en ambas formas de realizar los proyectos. Además se explicarán los pasos futuros que se realizarán en el proyecto para finalizarlo y dotarlo de toda la funcionalidad que se planteó en un inicio.

6.1. Comparativa con un desarrollo privado

A lo largo de toda mi vida laboral, esta es prácticamente mi primera experiencia real en la participación de un desarrollo FLOSS. Como he podido comprobar existen gran cantidad de similitudes y también grandes diferencias entre un proyecto FLOSS y uno de software privado.

- **Metodología de trabajo**

  Como se ha explicado en los apartados anteriores, para esta colaboración se han establecido una serie de pasos que nos permiten establecer las bases para llevar a buen puerto el desarrollo de la herramienta planteada, basándonos en una metodología de trabajo. En este punto, ambos tipos de proyectos convergen, de esta forma en cualquier tipo de desarrollo el primer paso es elegir y utilizar una metodología de desarrollo. Sin embargo, divergen en cuanto a la selección de dicha metodología, que en un proyecto FLOSS se puede usar la metodología que uno desee o que considere más adecuada y sin embargo en empresas privadas, al menos a lo largo de mi experiencia, es más habitual que exista

una metodología establecida que se aplica a todos los proyectos software.

- **Riesgos**

  Como hemos podido ver anteriormente uno de los grandes problemas que se han visto en el proceso de desarrollo ha sido la falta de tiempo para completar el desarrollo debido a varios motivos. Por un lado, el gran alcance del proyecto, ya que pasa de ser una herramienta relativamente sencilla y con una funcionalidad muy acotada, a ser una herramienta que dispone de un gran número de funcionalidades (asistente, ayuda en línea, diferentes tipos de intercambios) y cuya intención es crecer hasta convertirse en una gran red de intercambio descentralizada. Este tipo de problemas suelen aparecer también también en los proyectos de software privativo, ya que en muchas ocasiones el alcance de un proyecto cambia sobre la marcha o bien no se evalúa de forma correcta y los tiempos se disparan, haciendo que el proyecto tenga una duración mayor que la estimada a priori al inicio del desarrollo.

  Por otro lado, como último problema surgido ha sido el equipo de trabajo, al ser un proyecto FLOSS se basa principalmente en la colaboración voluntaria de miembros de la comunidad que deseen participar en este proyecto. En el caso de Troco, comenzamos el proyecto varias personas, pero a medida que avanzó el proyecto sólo nos quedamos dos personas, Vicente J. Ruiz y yo. Este tipo de riesgos es prácticamente inexistente en los proyectos de software privativo, ya que se establece un grupo de trabajo desde el principio que no suele variar en exceso, por tanto no surgen riesgos de falta de recursos.

- **Comunicación**

  Un punto importante, o al menos lo considero importante, es la forma de comunicarse. En este proyecto principalmente se han utilizado herramientas de comunicación como son el correo electrónico, la plataforma Kune y sólo en una ocasión una reunión presencial con algún miembro de Comunes, que suelen ser herramientas habituales en los proyectos FLOSS debido a que los colaboradores pueden ser de cualquier lugar del mundo con lo que realizar reuniones presenciales es algo bastante complejo. Sin embargo en proyectos de software privativo la comunicación con el cliente suele realizarse de forma presencial en el inicio del proyecto y para los aspectos importantes, y paulatinamente se tiende a

utilizar el correo electrónico para detalles menores. Entre el equipo de desarrollo suele ser presencial también, habitualmente debido a que este equipo se sitúa físicamente en el mismo lugar de trabajo.

- **Herramientas**

  Esta experiencia me ha permitido no sólo trabajar en el desarrollo de una herramienta FLOSS junto con miembros de una comunidad, sino además utilizando exclusivamente herramientas FLOSS para cada una de las fases, desde el sistema operativo, las herramientas de gestión, el entorno de desarrollo, etc. Tras este periodo de tiempo, puedo decir que no he tenido ningún problema con herramientas FLOSS y me han permitido desarrollar mi trabajo de una manera satisfactoria, con un rendimiento similar a otras herramientas privativas utilizadas a lo largo de mi vida.

- **Toma de decisiones**

  En los proyectos FLOSS habitualmente la toma de decisiones se realiza en conjunto y por consenso entre los miembros de la comunidad, sin embargo en la mayor parte de los proyectos de software privativo la toma de decisiones del proyectos las realiza la capa de analistas o el jefe de proyecto, y sólo se consensúan en común las decisiones puramente técnicas.

6.2. Conclusiones y lecciones aprendidas

La participación en este proyecto me ha permitido aprender gran cantidad de cosas, no sólo a nivel técnico, sino sobre los proyectos de software libre, su organización, su comunicación, etc. Para esto, han sido fundamentales todos los conocimientos adquiridos durante cada una de las asignaturas del máster [72].

Las conclusiones obtenidas de mi participación en este proyecto son las siguientes:

- Troco proporcionará una herramienta completamente descentralizada para el intercambio

libre de productos y servicios, de forma que ofrecerá una plataforma perfecta para que tanto colectivos como personas individuales puedan intercambiar diferentes productos y habilidades sin depender del sistema de comercio convencional, y así establecer las bases para colaboraciones y ayudas futuras.

- Este proyecto me ha permitido ver y conocer una amplia parte de la organización de una comunidad de software libre desde dentro, participando tanto en las decisiones como en la elaboración de documentación y contenido para dicha comunidad.

- Participar en este desarrollo me ha permitido aprender bastantes cosas sobre el framework GWT, permitiendo ampliar mis conocimientos de Java y brindándome los cimientos necesarios para seguir avanzando en este aspecto en un futuro.

- Además del lenguaje de programación, este proyecto me ha permitido conocer en detalle y utilizar varias herramientas FLOSS como el IDE eclipse, el gestor de versiones Git, herramientas como pencil para el prototipado o como dia para la realización de diagramas.

- La comunicación siempre es un tema importante en cualquier tipo de proyecto, y sobre todo en un proyecto FLOSS, por esto se hace fundamental establecer un canal claro y sencillo para permitir una comunicación fluida y eficiente entre todos los miembros del proyecto, y es primordial establecerlo desde el principio y hacerlo extensible a todos los miembros para su utilización.

- He podido apreciar que existen gran cantidad de herramientas para intercambio de bienes y servicios en línea, pero la gran mayoría están orientadas a un intercambio específico y son centralizadas, por tanto no proporcionan una herramienta completa y que puede ser gestionada por cada grupo de trabajo para orientarla a sus características y necesidades.

6.3. Conocimientos y competencias adquiridos en el Máster que han sido de utilidad en este proyecto

- Introducción al software libre me ha permitido conocer y comprender cómo funciona el modelo del software libre y el contexto en el que se ha desarrollado a lo largo de su corta historia, y sobre todo conocer bien las libertades que otorga a los usuarios. En este proyecto he podido apreciar que varias cuestiones que se asocian al software libre son completamente falsas, como que las herramientas FLOSS son más complejas que las privativas o que funcionan peor que las de pago.

- La asignatura Aspectos legales me permitió conocer todas las implicaciones legales que tiene el licenciamiento de software libre, sobre todo en cuanto a la compatibilidad con otras herramientas que son utilizadas en el proyecto. En nuestro caso se ha elegido la licencia AGPL v3 por dos motivos, en primer lugar debido a que es la licencia utilizada en la plataforma Kune y en segundo porque es compatible con las licencias utilizadas en el resto de librerías y herramientas que se utilizan en el proyecto.

- En Aspectos Económicos del software libre me permitió comprender las diferentes motivaciones que tienen los distintos tipos de colaboradores que hay en proyectos de software libre. En este proyecto la mayor motivación ha sido mejorar mis habilidades técnicas en un lenguaje de programación en el que no dispongo de gran experiencia ni grandes conocimientos.

- La asignatura Gestión de proyectos me permitió conocer el papel fundamental que tiene la comunicación y la documentación de proyectos FLOSS, y que he tenido la ocasión de comprobar en este proyecto, que avanzó en gran medida cuando se estableció un canal eficaz para la comunicación y la documentación del proyecto.

- Las asignaturas de Herramientas de desarrollo y Desarrollo avanzado, son las dos asignaturas que más me han ayudado en este proyecto, debido a que en ambos casos utilizamos con diferentes plugins el IDE eclipse y el gestor de versiones Git, lo que me ha dado un

conocimiento amplio de estas herramientas lo que me ha facilitado las cosas a lo hora de usarlos y para la integración de los plugins para GWT y Git.

- El Practicum fue mi primera toma de contacto con GWT y con la asociación Comunes, aunque en esa ocasión el desarrollo fue menor y más sencillo puesto que se trataba de contenido estático, me sirvió para conocer a varias personas de la comunidad que me han ayudado muchísimo y para tener mi primera experiencia con GWT.

- En general, todas las asignaturas me han proporcionado una visión general de todo lo relacionado con el software libre, pero sobretodo me han dado a conocer todas las posibilidades que existen entorno a este modelo, no sólo para los desarrolladores, sino también para otros perfiles como traductores, gestores, personas del mundo del marketing y la publicidad, etc. En mi caso, me abre un ábanico de posibilidades técnicas casi ilimitado, ya que, como he podido experimentar al participar en un proyecto FLOSS proporciona el escenario perfecto para aprender nuevos lenguajes y técnicas de programación, y además de aplicarlos a un entorno real donde ver sus resultados.

6.4. Futuros pasos

Como se ha explicado anteriormente, por problemas de tiempo y por el amplio alcance del proyecto, no ha sido posible finalizar el desarrollo completo del nuevo gadget de Troco. Sin embargo, se han sentado las bases para poder avanzar de una manera relativamente rápida en el desarrollo de la aplicación. Entre mis propósitos está mantener este proyecto tras la finalización de este máster.

Los pasos que se deben realizar a partir de este momento son los siguientes:

- Finalizar la capa de presentación, tanto de las pantallas del asistente como de la herramienta básica de intercambios, donde se reutilizarán muchos de los componentes a nivel de interfaz de usuario implementados para el wizard que sea necesario para poder

completar esta funcionalidad de la herramienta.

- Realizar el desarrollo de la capa de negocio, donde se implementarán cada una de las funciones para la validación de datos, actualización de estados, conexión con el repositorio de datos, etc.

- Crear las tablas reflejadas en el modelo de datos expuesto en el apartado X, para que a diferencia de la primera versión del gadget el almacenamiento no se realice a nivel de fichero sino a nivel de base de datos.

- Desarrollar la lógica de la capa de acceso a datos, para almacenar la información de cada uno de los intercambios en las tablas mencionadas anteriormente. En este punto se establecerán cada una de las funciones y procedimientos que realizarán el traspaso de datos desde las pantallas a las tablas de la base de datos.

- Investigar otras funcionalidades como el buscador de intercambios o los intercambios en cadena, dotando a la herramienta de más funcionalidades que permitan a los usuarios abordar cualquier tipo de intercambio que necesiten.

A. Apéndice 1

En este anexo se mostrarán algunas capturas de la aplicación en estado en el que se detuvo la colaboración para la defensa de este proyecto. Actualmente el proyecto está en plena fase de desarrollo, ya que se está refactorizando el código para generar los componentes básicos para la opción básica de intercambio y de está forma finalizar la capa de interfaz de usuario, con el objetivo de comenzar con toda la lógica de negocio de la herramienta.

A continuación en la figura A.1, se puede apreciar la pantalla inicial del nuevo gadget de Troco:

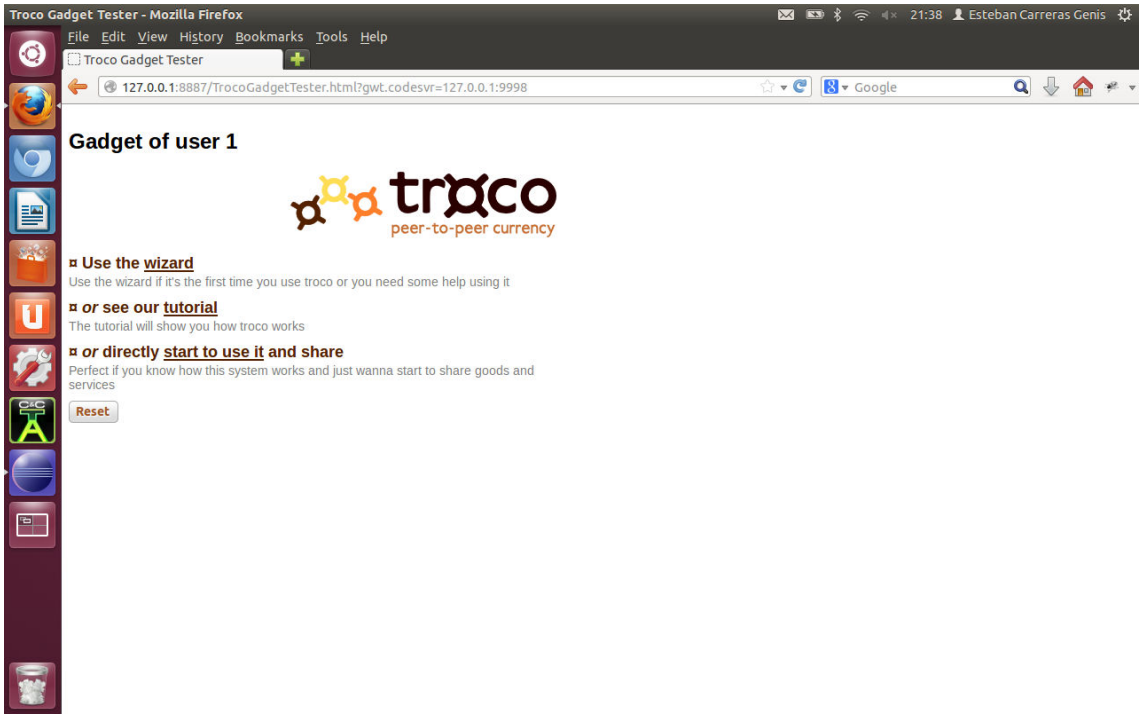

Figura A.1.: Pantalla incial del gadget de Troco

En la figura A.2 podemos ver la pantalla del asistente cuando se selecciona esta opción en la pantalla principal del gadget:

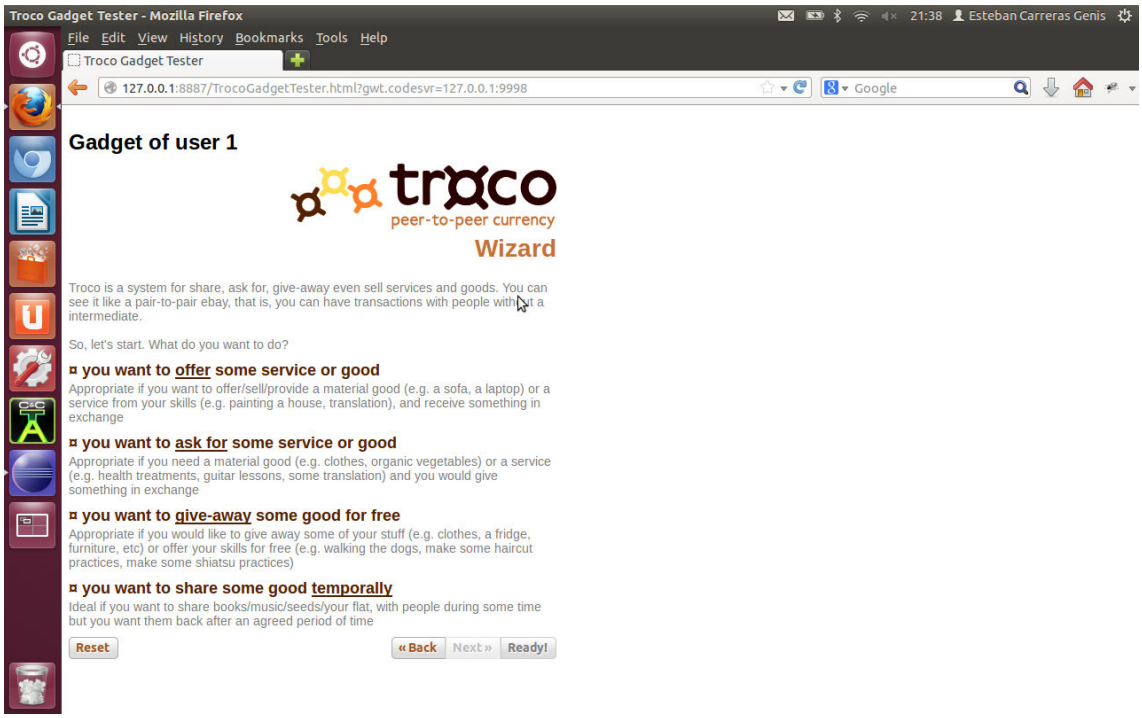

Figura A.2.: Pantalla de opciones del asistente

A continuación, se mostrarán las pantallas donde se definen cada uno de los formularios de las opciones de asistente de Troco, desde la opción para ofrecer una producto o servicio, la opción para solicitar algún producto que desees, otra opción para regalar bienes a otro usuario, hasta la opción para compartir temporalmente bienes:

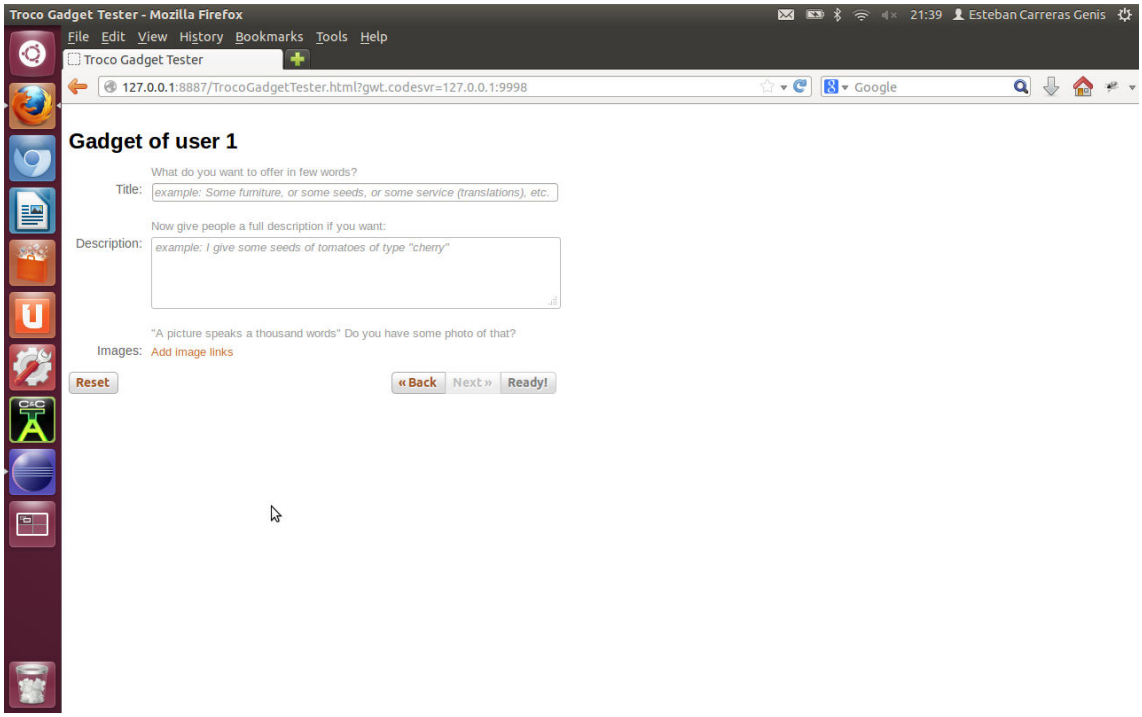

Figura A.3.: Opción ofrecer del asistente

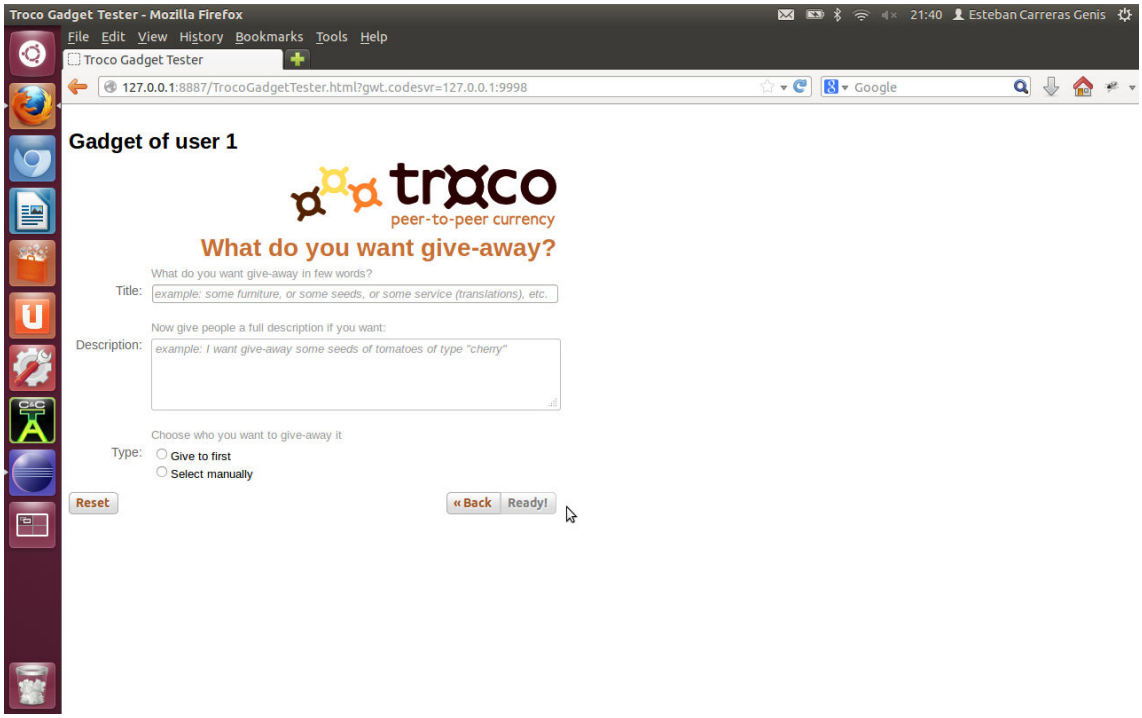

Figura A.4.: Opción regalar del asistente

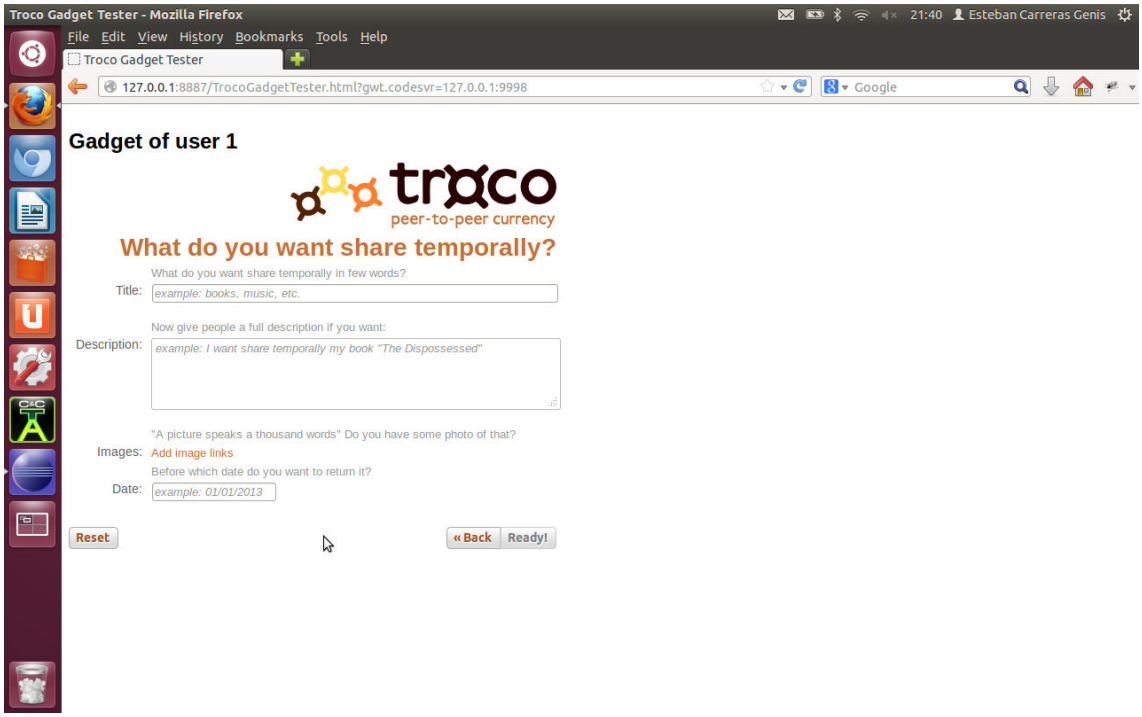

Figura A.5.: Opción compartir temporalmente del asistente

Bibliografía

[1] Proyecto Kune. http://kune.ourproject.org/es/

[2] Colectivo Comunes. http://comunes.org/es/

[3] Primera versión gadget Troco. http://troco.ourproject.org/

[4] Free Software Foundation. http://www.fsf.org/

[5] Google Web Toolkit. https://developers.google.com/web-toolkit/?hl=es

[6] Apache Wave. http://incubator.apache.org/wave/

[7] WAT System. http://www.watsystems.net/

[8] i-WAT. http://www.media-art-online.org/iwat/help/about.html

[9] Información sobre el sistema WAT. http://imaginationforpeople.org/en/project/wat-complementary-currency/

[10] Eiichi Morino. http://www.zoominfo.com/p/Eiichi-Morino/1167867042

[11] Sociedad para la Investigación de Gesell en el Japón. http://www3.plala.or.jp/mig/society-es.html

[12] GNU GPL v1. http://www.gnu.org/licenses/old-licenses/gpl-1.0.html

[13] Trocobuy. http://www.trocobuy.com/

[14] Truequeweb. http://www.truequeweb.com/

[15] Bookmooch. http://es.bookmooch.com/

[16] Titlerader. http://www.titletrader.com/

[17] Etruekko. http://www.etruekko.com/

[18] Cadena de cambios. http://www.cadenadecambios.com/

[19] Polyglot. http://polyglotclub.com/

[20] Home for Home. http://www.homeforhome.com/

[21] MySQL. http://www.mysql.com/

[22] Git. http://git-scm.com/

[23] MVP. http://www.wildcrest.com/Potel/Portfolio/mvp.pdf

[24] Librería gwt-pectin. http://code.google.com/p/gwt-pectin/

[25] Java. http://www.java.com/es/

[26] Ranking TIBOE. http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

[27] Sun Microsystems. http://www.sun.com

[28] Oracle. http://www.oracle.com/index.html

[29] C/C++ resources. http://www.cplusplus.com/

[30] Google. https://www.google.es/intl/es/about/

[31] Licencia Apache versión 2. http://www.apache.org/licenses/LICENSE-2.0.html

[32] Versiones GWT. https://developers.google.com/web-toolkit/versions

[33] JavaOne. http://www.oracle.com/javaone/index.html

[34] JUnit. http://junit.org/

[35] GWT Bug tracking. http://code.google.com/p/google-web-toolkit/issues/list

[36] Subversion. http://subversion.apache.org/

[37] SafeHTML. https://developers.google.com/web-toolkit/doc/latest/DevGuideSecuritySafeHtml

[38] SpringSource. http://www.springsource.org/

[39] Eclipse. http://www.eclipse.org/

[40] GWT UI Designer. https://developers.google.com/web-toolkit/tools/gwtdesigner/

[41] Android. http://www.android.com/

[42] Google Play. https://play.google.com

[43] UiBinder. https://developers.google.com/eclipse/docs/gwt_uibinder

[44] Plugin GWT4NB. https://java.net/projects/gwt4nb/pages/Home

[45] NetBeans. https://netbeans.org/

[46] Common Development and Distribution License versión 1. http://opensource.org/licenses/CDDL-1.0

[47] Apache Ant. http://ant.apache.org/

[48] Google App Engine. https://appengine.google.com/

[49] Plugin GWT para Eclipse. https://developers.google.com/eclipse/

[50] Licencia EPL versión 1. http://www.eclipse.org/legal/epl-v10.html

[51] Google I/O. https://developers.google.com/events/io/

[52] Fundación Apache. http://www.apache.org/

[53] Wave in a Box. http://waveinabox.net/auth/signin?r=/

[54] OpenJDK. http://openjdk.java.net/

[55] Ruby on Rails. http://rubyonrails.org/

[56] Pyjamas. http://code.google.com/p/pyjamas/

[57] Versión Ostrom de Kune. http://kune.ourproject.org/es/2012/10/new-release-of-collaborative-distributed-social-network-kune-ostrom/

[58] Versión 99 % de Kune. http://kune.ourproject.org/2012/04/new-release-website-launch-kune-cc/

[59] Versión 15-M de Kune. http://comunes.org/es/747/new-release-of-kune-published-codename-15m/

[60] XMPP. http://xmpp.org/

[61] Elinor Ostrom. http://www.indiana.edu/~workshop/people/lostromcv.htm

[62] Affero General Public License. http://www.gnu.org/licenses/agpl.html

[63] Google Wave Protocol. http://www.waveprotocol.org/

[64] Python. http://www.python.org/

[65] Arquitectura de MySQL. http://cnx.org/content/m18938/latest/

[66] Repositorio de la primera versión del gadget de Troco. http://ourproject.org/scm/?group_id=784

[67] Gitorious. https://gitorious.org/

[68] Linus Torvalds. http://www.internethalloffame.org/inductees/

linus-torvalds

[69] Bazaar. http://bazaar.canonical.com/en/

[70] Mercurial. http://mercurial.selenic.com/

[71] Nuevo repositorio de Troco. https://gitorious.org/troco/troco

[72] Moodle del Máster Universitario en Software Libre. http://docencia.etsit.urjc.es/moodle/course/view.php?id=135

[73] Página personal de John Redmood. http://john.redmood.com/