

PROYECTO DEL PRIMER CUATRIMESTRE



UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INFORMÁTICA

INGENIERÍA
EN INFORMÁTICA

PROYECTO FIN DE CARRERA

Proyecto del Primer Cuatrimestre

Buggin' n Debuggin':
Cristian Del Cerro Gómez
Laura Del Río Avilés
Javier García Simón
Raúl Reguillo Carmona

Mayo, 2013



UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INFORMÁTICA
Departamento de Tecnologías y Sistemas de Información

PROYECTO FIN DE CARRERA

Proyecto del Primer Cuatrimestre

Autor: Buggin' n Debuggin':

Cristian Del Cerro Gómez

Laura Del Río Avilés

Javier García Simón

Raúl Reguillo Carmona

Director: Dr. Macario Polo Usaola

Mayo, 2013

Buggin' n Debuggin':
Cristian Del Cerro Gómez
Laura Del Río Avilés
Javier García Simón
Raúl Reguillo Carmona

Ciudad Real – Spain

E-mail: Buggin.n.Debuggin@gmail.com

Teléfono: 123 456 789

Web site: <http://esi.uclm.es>

© 2013 Buggin' n Debuggin':
Cristian Del Cerro Gómez
Laura Del Río Avilés
Javier García Simón
Raúl Reguillo Carmona

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Se permite la copia, distribución y/o modificación de este documento bajo los términos de la Licencia de Documentación Libre GNU, versión 1.3 o cualquier versión posterior publicada por la *Free Software Foundation*; sin secciones invariantes. Una copia de esta licencia esta incluida en el apéndice titulado «GNU Free Documentation License».

Muchos de los nombres usados por las compañías para diferenciar sus productos y servicios son reclamados como marcas registradas. Allí donde estos nombres aparezcan en este documento, y cuando el autor haya sido informado de esas marcas registradas, los nombres estarán escritos en mayúsculas o como nombres propios.

TRIBUNAL:

Presidente:

Vocal 1:

Vocal 2:

Secretario:

FECHA DE DEFENSA:

CALIFICACIÓN:

PRESIDENTE

VOCAL 1

VOCAL 2

SECRETARIO

Fdo.:

Fdo.:

Fdo.:

Fdo.:

Resumen

El presente documento pretende exponer el proyecto del primer cuatrimestre para la asignatura *Ingeniería del Software II*.

Este documento es una reedición del original, que convenientemente hemos sustituido dada las deficiencias del anterior. Se expone en formato Proyecto Fin de Carrera.

Sobre el proyecto, nos centraremos en exponer cómo ha sido el desarrollo de una red social, en este caso sin ayuda de ningún framework. El motivo de la red social es del de conectar a autores de artículos (que se puede extender a cualquier tipo de publicación) con editores y revisores.

Se explicará a su vez la metodología usada, el marco tecnológico, diseño de la herramienta y finalmente se expondrán las conclusiones en base a todo el trabajo realizado.

Índice general

Resumen	VI
Índice general	VII
Índice de cuadros	X
Índice de figuras	XI
Índice de listados	XIII
Listado de acrónimos	XIV
Agradecimientos	XV
1. Introducción	1
2. Objetivos	3
2.1. Objetivos del proyecto	3
2.2. Captura de Requisitos	3
3. Estado del Arte	6
3.1. Una primera aproximación	6
3.2. Redes Sociales en Internet	6
4. Metodologías	10
4.1. Proceso Unificado de Desarrollo	10
4.2. Tareas Preparativas	11
4.3. Estructura e Iteraciones del PUD	12
5. Marco Tecnológico	17
5.1. Java	17
5.2. Patrones	17

5.2.1.	Patrón <i>Singleton</i>	17
5.2.2.	Patrón <i>DAO</i>	18
5.2.3.	Patrón <i>Abstract Factory</i>	18
5.3.	Struts2	18
5.4.	Apache Tomcat	19
5.5.	MySQL	20
5.6.	Virtualizaciones	22
5.6.1.	Virtualización de la base de datos	22
5.6.2.	Software usado para la virtualización	22
5.6.3.	Método de trabajo	23
6.	Diseño de la Herramienta	24
6.1.	Struts 2	24
6.2.	Patrones	25
6.2.1.	Patrón <i>Singleton</i> con pool de conexiones	25
6.2.2.	Patrón <i>DAO</i>	25
6.2.3.	Patrón <i>Abstract Factory</i>	26
6.3.	Diseño y diagramas	27
7.	Pruebas	38
7.1.	Pruebas con junit	38
8.	Conclusiones	41
8.1.	Conclusiones generales	41
8.2.	Trabajo futuro	42
8.3.	Notas sobre la segunda entrega	42
A.	Definición de la empresa	44
A.1.	Componente organizacional	44
A.1.1.	Organigrama	44
A.1.2.	Acerca de la empresa Buggin' & Debuggin'	45
A.1.3.	Plan de inversión	46
A.1.4.	Estrategia de marketing y publicidad	47
B.	Informes de Microsoft Project	48
C.	Informes de pruebas	53

Índice de cuadros

7.1. Ejemplo de informe de pruebas	40
--	----

Índice de figuras

4.1.	Lista de tareas para la creación de la empresa e instalación del sistema . . .	11
4.2.	Diagrama de Gantt correspondiente a las tareas de creación de la empresa e instalación del sistema	12
4.3.	Esquema de esfuerzo siguiendo Proceso Unificado de Desarrollo	13
4.4.	Lista de iteraciones y tareas generales de todo el proyecto	13
4.5.	Diagrama de Gantt correspondiente a las tareas e iteraciones de todo el proyecto	14
4.6.	Ejemplo de especificación detallada de las tareas que forman cada iteración	14
4.7.	Diagrama de Gantt correspondiente al ejemplo de especificación detallada de las tareas de cada iteración	15
5.1.	Logotipo Apache Tomcat 7	20
5.2.	Logotipo MySQL	21
6.1.	Modelo de la Base de Datos	27
6.2.	Caso de Uso: Usuario	28
6.3.	Caso de Uso: Admin	29
6.4.	Caso de Uso: Autor	29
6.5.	Caso de Uso: Revisor	30
6.6.	Caso de Uso: Editor	30
6.7.	Diagrama de Clases: Árbol de herencia	30
6.8.	Diagrama de Clases: Usuario	31
6.9.	Diagrama de Clases: Autor y Sistema	32
6.10.	Diagrama de Clases: Revisor	33
6.11.	Diagrama de Clases: Admin y Editor	33
6.12.	Diagrama de Clases: Artículo	33
6.13.	Diagrama de Clases: Revisión de Artículo	34
6.14.	Diagrama de Clases: Editor con Revisor y Tag	34
6.15.	Diagrama de Secuencia: Realizar una Evaluación	35
6.16.	Diagrama de Secuencia: Enviar publicación	36
6.17.	Diagrama de Secuencia: Notificar seguidor (I)	36

6.18. Diagrama de Secuencia: Notificar seguidor (y II)	37
A.1. Organigrama de la empresa	44
B.1. Informe MS Project: General	48
B.2. Informe MS Project: Hitos	49
B.3. Informe MS Project: Flujo de caja	49
B.4. Informe MS Project: Reparto de tareas (I)	50
B.5. Informe MS Project: Reparto de tareas (II)	50
B.6. Informe MS Project: Reparto de tareas (III)	51
B.7. Informe MS Project: Reparto de tareas (y IV)	52
C.1. Informe de pruebas (I)	53
C.2. Informe de pruebas (II)	54

Índice de listados

6.1. Implementación de CrudDAO.java	26
---	----

Listado de acrónimos

Agradecimientos

A Macario, por dejarnos corregir los errores de la anterior entrega y ser lo suficientemente paciente como para esperar esta otra.

Buggin' n Debuggin'

Por amor al arte y exigencias del gui3n.

Capítulo 1

Introducción

RED Social, hoy día es sinónimo de comunicación, conexión entre usuarios y archivos multimedia. El auge de estas tecnologías se ha convertido en un auténtico fenómeno en Internet que lleva durante años pegando fuerte entre usuarios de todas las edades.

Podemos encontrar una amplia variedad de redes sociales que no únicamente se limitan a ser contenedores de fotos de las más variopintas situaciones. Dentro de todas las posibles particularizaciones de esta aplicación, encontramos redes sociales que se orientan a mantenerse en contacto con amigos, administrar fotografías, de mensajes cortos o incluso de contactos de empleo.

Todo esto se fundamenta en sólidos conceptos de persistencia. El manejo de los datos en una red social es primordial, pues se generan cantidades ingentes de información que debe ser eficientemente almacenada y rescatada para dar al usuario la mejor de las experiencias.

El uso de un framework podría resultar interesante a la hora de desarrollar la aplicación. No obstante hemos procedido de la manera más *artesanal* con el fin de profundizar en las herramientas indicadas durante el curso: **Struts 2**.

Como tecnologías adyacentes explicaremos *Apache Tomcat* y *MySQL*, servidor de aplicaciones web y sistema gestor de base de datos respectivamente.

Para finalizar esta introducción, incluiremos una pequeña descripción del documento a partir de este punto, en relación con el proyecto llevado a cabo.

- **Capítulo 2: Objetivos del proyecto.** Se abordarán los objetivos que se persiguen con la realización del proyecto.
- **Capítulo 3: Estado del Arte.** Donde se indicará la panorámica actual de las Redes Sociales así como las tendencias en el sector.
- **Capítulo 4: Metodologías usadas.** Se pondrá de manifiesto la metodología escogida para el particular, así como una serie de datos de documentación al margen de este documento, en relación a gestión del proyecto y del personal.
- **Capítulo 5: Marco tecnológico.** Se describirán las tecnologías utilizadas para la implantación..

- **Capítulo 6: Diseño de la herramienta.** Donde detallaremos cómo se ha implementado la herramienta.
- **Capítulo 7: Pruebas.** Se describirán las pruebas realizadas para probar la consistencia de la aplicación.
- **Capítulo 8: Conclusiones.** Donde se pondrá de manifiesto una perspectiva global acerca del desarrollo y el sector.

Capítulo 2

Objetivos

2.1. Objetivos del proyecto

Los objetivos para este proyecto es la realización de una red social con una funcionalidad completa para la gestión de artículos. Para ello se tiene que realizar una aplicación web siguiendo el Proceso Unificado de Desarrollo. El desarrollo de la web se hará bajo el framework *Struts2* y será necesario tener un servidor con *Apache Tomcat* donde alojar la web. Para programar el dominio de la web será necesario programar varios patrones de diseño de los estudiados en clases.

Se debe alcanzar unos niveles de usabilidad y de apariencia mínimos de la web.

La herramienta se desarrollará en ausencia de frameworks específicos de desarrollo de redes sociales.

2.2. Captura de Requisitos

Debido a la moda en alza del sector, se decidió llevar a cabo la implementación de una aplicación a modo de red social. La idea subyacente es sencilla: pongamos que un colectivo de escritores *amateurs* quieren dar a conocerse en el mundo. La red social da cabida gratuita tanto a Autores y Editores. Dichos Editores pondrán a sus propios revisores al cargo de revisión de documentos, a modo de *cazatalentos*. Si dichos revisores encuentran un artículo (referido a un topic o tag en concreto) que resulte interesante, pueden remitirlo a su editor que tendrá la opción de comprarlo directamente al autor (operación mediante la cual, el equipo gestor de la red social se lleva una pequeña comisión).

Por otro lado existe la función de los llamados *Karmapoints*. Estos puntos sirven como indicativo de nivel de autores y revisores. Así, un autor novato empezará con pocos *Karmapoints* que irán incrementándose a medida que sus artículos sean revisados. Cuanto mejor sea la nota, más karmapoints será capaz de sumar. De este modo existe un control sobre la popularidad de ciertos autores y revisores. De la misma forma, a un autor solamente le revisarán sus publicaciones aquellos revisores cuyo nivel medido en *Karmapoints* sea similar al suyo, buscando un equilibrio y cierto rigor en la evaluación de documentos.

Un autor también puede solicitar que su artículo sea revisado por un profesional (VIP, es decir, no amateur). En ese caso, la suma de *Karmapoints* será mayor. Se considera que esta solicitud también tenga un cargo económico.

La idea de crear una red social abarca una extensión que sólo tiene límites allá donde nosotros los pongamos. Tras una concepción de idea principal, se fueron desarrollando una serie de **requisitos** que más adelante para el particular del proyecto, **fueron refinados hasta minimizar los objetivos a un esquema de funcionamiento sencillo, ajustándolos así al objetivo del curso.**

La total implementación de la red social, una vez definida la arquitectura así como la metodología, solamente sería cuestión de tiempo y café.

A continuación los requisitos que, tras ser refinados, se han modelado en la red social.

- El usuario se debe loguear mediante un correo y contraseña
- En el perfil de usuario se especificarán los siguientes campos
 - Nombre
 - Apellidos
 - Edad
 - Lugar de residencia
 - Correo electrónico
 - Contraseña
- Cualquier usuario podrá modificar sus datos personales, ver su historial, puntuaciones y visitas
- Habrá tres roles diferentes: Autor, Revisor y Editor
- Los usuarios se podrán mandar mensajes privados entre sí
- Un autor podrá:
 - Realizar envíos de artículos para que puedan ser revisados
 - Cancelar la revisión de cualquier artículo de su propiedad
 - Realizar comentarios en sus propios artículos
- Un revisor podrá:
 - Comentar cualquier artículo de autores que le haya asignado el editor
 - Aceptar la revisión de ciertos artículos que el editor le ha encomendado
 - Puntuar artículos que el sistema le ha encargado
- Un editor podrá:
 - Gestionar su lista de revisores

- Contactar con cualquiera de los demás usuarios
- Visualizar los artículos y comentarios de sus revisores
- Cuando un autor realiza la publicación de un artículo debe rellenar ciertos campos:
 - Tags: palabras claves que se refieren al tema del que versa el artículo
 - Título
 - Resumen: breve resumen del artículo
- Los campos de un artículo publicado no podrán modificarse (a excepción de los comentarios)
- Un revisor podrá revisar un artículo una sola vez
- Las únicas personas capaces de realizar comentarios sobre los artículos serán los revisores y el propietario
- Debe guardarse un histórico de publicaciones enviadas/revisadas
- El usuario recibirá una notificación en cualquiera de estos casos
 - Su publicación ha sido revisada
 - Ha recibido un encargo de revisión
 - Ha recibido un mensaje
 - Tiene un nuevo seguidor
- El sistema otorgará una serie de puntos a autores y revisores en función a la valoración del artículo
- El sistema asignará automáticamente artículos con revisores, atendiendo a su afinidad por puntos
- Un usuario puede solicitar que su publicación sea revisada por un revisor profesional (VIP)

Capítulo 3

Estado del Arte

3.1. Una primera aproximación

Una red social es una estructura social compuesta por un conjunto de actores (tales como individuos u organizaciones) que están conectados por díadas denominadas lazos interpersonales, que se pueden interpretar como relaciones de amistad, parentesco, entre otros. La investigación multidisciplinar ha mostrado que las redes sociales operan en muchos niveles, desde las relaciones de parentesco hasta las relaciones de organizaciones a nivel estatal (se habla en este caso de redes políticas), desempeñando un papel crítico en la determinación de la agenda política y el grado en el cual los individuos o las organizaciones alcanzan sus objetivos o reciben influencias.

El análisis de redes sociales estudia esta estructura social aplicando la teoría de grafos e identificando las entidades como *nodos* o *vértices* y las relaciones como *enlaces* o *aristas*. La estructura del grafo resultante es a menudo muy compleja. Como se ha dicho, En su forma más simple, una red social es un mapa de todos los lazos relevantes entre todos los nodos estudiados. Se habla en este caso de redes *socio céntricas* o *completas*. Otra opción es identificar la red que envuelve a una persona (en los diferentes contextos sociales en los que interactúa); en este caso se habla de *red personal*.

La red social también puede ser utilizada para medir el capital social (es decir, el valor que un individuo obtiene de los recursos accesibles a través de su red social). Estos conceptos se muestran, a menudo, en un diagrama donde los nodos son puntos y los lazos, líneas. Red social también se suele referir a las plataformas en Internet. Las redes sociales de internet cuyo propósito es facilitar la comunicación y otros temas sociales en el sitio web.

3.2. Redes Sociales en Internet

El software germinal de las redes sociales parte de la teoría de los seis grados de separación, según la cual toda la gente del planeta está conectada a través de no más de seis personas. De hecho, existe una patente en EEUU conocida como six degrees patent por la que ya han pagado Tribe y LinkedIn. Hay otras muchas patentes que protegen la tecnología para automatizar la creación de redes y las aplicaciones relacionadas con éstas. Estas redes

sociales se basan en la teoría de los seis grados, Seis grados de separación es la teoría de que cualquiera en la Tierra puede estar conectado a cualquier otra persona en el planeta a través de una cadena de conocidos que no tiene más de seis intermediarios. La teoría fue inicialmente propuesta en 1929 por el escritor húngaro Frigyes Karinthy en una corta historia llamada Chains. El concepto está basado en la idea que el número de conocidos crece exponencialmente con el número de enlaces en la cadena, y sólo un pequeño número de enlaces son necesarios para que el conjunto de conocidos se convierta en la población humana entera. El término red social es acuñado principalmente a los antropólogos ingleses Jhon Barnes y Elizabeth Bott, ya que, para ellos resultaba imprescindible considerar lazos externos a los familiares, residenciales o de pertenencia a algún grupo social.¹⁹ Los fines que han motivado la creación de las llamadas redes sociales son varios, principalmente, es el diseñar un lugar de interacción virtual, en el que millones de personas alrededor del mundo se concentran con diversos intereses en común. Recogida también en el libro *Six Degrees: The Science of a Connected Age* del sociólogo Duncan Watts, y que asegura que es posible acceder a cualquier persona del planeta en tan solo seis saltos.

Según esta Teoría, cada persona conoce de media, entre amigos, familiares y compañeros de trabajo o escuela, a unas 100 personas. Si cada uno de esos amigos o conocidos cercanos se relaciona con otras 100 personas, cualquier individuo puede pasar un recado a 10.000 personas más tan solo pidiendo a un amigo que pase el mensaje a sus amigos.

Estos 10.000 individuos serían contactos de segundo nivel, que un individuo no conoce pero que puede conocer fácilmente pidiendo a sus amigos y familiares que se los presenten, y a los que se suele recurrir para ocupar un puesto de trabajo o realizar una compra. Cuando preguntamos a alguien, por ejemplo, si conoce una secretaria interesada en trabajar estamos tirando de estas redes sociales informales que hacen funcionar nuestra sociedad. Este argumento supone que los

100 amigos de cada persona no son amigos comunes. En la práctica, esto significa que el número de contactos de segundo nivel será sustancialmente menor a 10.000 debido a que es muy usual tener amigos comunes en las redes sociales. Si esos 10.000 conocen a otros 100, la red ya se ampliaría a 1.000.000 de personas conectadas en un tercer nivel, a 100.000.000 en un cuarto nivel, a 10.000.000.000 en un quinto nivel y a 1.000.000.000.000 en un sexto nivel. En seis pasos, y con las tecnologías disponibles, se podría enviar un mensaje a cualquier lugar individuo del planeta. Evidentemente cuanto más pasos haya que dar, más lejana será la conexión entre dos individuos y más difícil la comunicación. Internet, sin embargo, ha eliminado algunas de esas barreras creando verdaderas redes sociales mundiales, especialmente en segmento concreto de profesionales, artistas, etc. En la década de los 50, Ithiel de Sola Pool (MIT) y Manfred Kochen (IBM) se propusieron demostrar la teoría matemáticamente. Aunque eran capaces de enunciar la cuestión *dado un conjunto de N personas, ¿cual es la probabilidad de que cada miembro de estos N estén conectados con otro miembro vía k_1, k_2 ,*

k_3, \dots, k_n enlaces?, después de veinte años todavía eran incapaces de resolver el problema a su propia satisfacción.

En 1967, el psicólogo estadounidense Stanley Milgram ideó una nueva manera de probar la Teoría, que él llamó *el problema del pequeño mundo*. El experimento del mundo pequeño de Milgram consistió en la selección al azar de varias personas del medio oeste estadounidense para que enviaran tarjetas postales a un extraño situado en Massachusetts, situado a varios miles de millas de distancia. Los remitentes conocían el nombre del destinatario, su ocupación y la localización aproximada. Se les indicó que enviaran el paquete a una persona que ellos conocieran directamente y que pensaran que fuera la que más probabilidades tendría, de todos sus amigos, de conocer directamente al destinatario. Esta persona tendría que hacer lo mismo y así sucesivamente hasta que el paquete fuera entregado personalmente a su destinatario final. Aunque los participantes esperaban que la cadena incluyera al menos cientos de intermediarios, la entrega de cada paquete solamente llevó, como promedio, entre cinco y siete intermediarios. Los descubrimientos de Milgram fueron publicados en *Psychology Today* e inspiraron la frase seis grados de separación. En The social software weblog han agrupado 120 sitios web en 10 categorías y QuickBase también ha elaborado un completo cuadro sobre redes sociales en Internet. El origen de las redes sociales se remonta, al menos, a 1995, cuando Randy Conrads crea el sitio web classmates.com. Con esta red social se pretende que la gente pueda recuperar o mantener el contacto con antiguos compañeros del colegio, instituto, universidad, etcétera.

En 2002 comienzan a aparecer sitios web promocionando las redes de círculos de amigos en línea cuando el término se empleaba para describir las relaciones en las comunidades virtuales, y se hizo popular en 2003 con la llegada de sitios tales como MySpace o Xing. Hay más de 200 sitios de redes sociales, aunque Friendster ha sido uno de los que mejor ha sabido emplear la técnica del círculo de amigos. La popularidad de estos sitios creció rápidamente y grandes compañías han entrado en el espacio de las redes sociales en Internet. Por ejemplo, Google lanzó Orkut el 22 de enero de 2004. Otros buscadores como KaZaZZ! y Yahoo crearon redes sociales en 2005. En estas comunidades, un número inicial de participantes envían mensajes a miembros de su propia red social invitándoles a unirse al sitio. Los nuevos participantes repiten el proceso, creciendo el número total de miembros y los enlaces de la red. Los sitios ofrecen características como actualización automática de la libreta de direcciones, perfiles visibles, la capacidad de crear

nuevos enlaces mediante servicios de presentación y otras maneras de conexión social en línea. Las redes sociales también pueden crearse en torno a las relaciones comerciales. Las herramientas informáticas para potenciar la eficacia de las redes sociales online (software social), operan en tres ámbitos, las 3 Cs, de forma cruzada:

- Comunicación (nos ayudan a poner en común conocimientos).

- Comunidad (nos ayudan a encontrar e integrar comunidades).
- Cooperación (nos ayudan a hacer cosas juntos).

El establecimiento combinado de contactos (blended networking) es una aproximación a la red social que combina elementos en línea y del mundo real para crear una mezcla. Una red social de personas es combinada si se establece mediante eventos cara a cara y una comunidad en línea. Los dos elementos de la mezcla se complementan el uno al otro. Vea también computación social. Las redes sociales continúan avanzando en Internet a pasos agigantados, especialmente dentro de lo que se ha denominado Web 2.0 y Web 3.0, y dentro de ellas, cabe destacar un nuevo fenómeno que pretende ayudar al usuario en sus compras en Internet: las redes sociales de compras. Las redes sociales de compras tratan de convertirse en un lugar de consulta y compra. Un espacio en el que los usuarios pueden consultar todas las dudas que tienen sobre los productos en los que están interesados, leer opiniones y escribirlas, votar a sus productos favoritos, conocer gente con sus mismas aficiones y, por supuesto, comprar ese producto en las tiendas más importantes con un solo clic. Esta tendencia tiene nombre, se llama Shopping 2.0. Un repaso por las mejores redes sociales:

- **Facebook:** Ahora mismo el reinado es de Facebook. En los últimos meses Facebook ha llegado al nivel de Twitter. Facebook es más estable técnicamente, y ofrece muchas características que Twitter no.
- **Twitter:** El crecimiento de Twitter ha sido increíble, llegando a estar en el podium de las redes sociales más utilizadas, y en tiempo record. Los desarrolladores de Twitter, están trabajando continuamente en nuevas características y una mejor funcionalidad.
- **LinkedIn:** Es la red social de profesionales. Es un gran recurso para la publicación de un curriculum vitae en línea y buscar trabajo. Tanto empresas como profesionales en busca de empleo, utilizan esta red social.
- **Myspace:** Hasta el año pasado, MySpace era el número uno de la red social. Sin embargo, MySpace se está quedando atrás. Myspace es muy utilizada por músicos y artistas.
- **Orkut:** De la mano de el todo poderoso Google, nos llega Orkut. Esta red social se está haciendo muy fuerte en países, como Brasil y la India.
- **Hi5:** Hi5 se mantiene por detrás de Facebook, MySpace y otras redes sociales. Tiene características muy similares a Facebook o MySpace, sin embargo no es ni mucho menos tan utilizada, salvo en América Central
- **Friendster:** Fue la pionera de las redes sociales. Facebook y MySpace le superaron hace ya varios años.
- **Xanga:** Otra de las pioneras. Nació en 1999. Xanga tiene unos 40 millones de usuarios registrados.

Fuente: [Wik]

Metodologías

4.1. Proceso Unificado de Desarrollo

Proceso Unificado de Desarrollo de Software (PUD) se puede definir como un conjunto de actividades necesarias para transformar los requisitos de usuario en un sistema software.

El Proceso Unificado no es simplemente un proceso, sino un marco de trabajo extensible que puede ser adaptado a organizaciones, proyectos específicos, diferentes áreas de aplicación o diferentes niveles de aptitud.

Sus características son las siguientes:

- **Dirigido por los casos de uso:** En el Proceso Unificado de Desarrollo los casos de uso se utilizan para capturar los requisitos funcionales y para definir los contenidos de las iteraciones. La idea es que cada iteración tome un conjunto de casos de uso o escenarios y desarrolle todo el camino a través de las distintas disciplinas: Análisis de requisitos, Diseño, Implementación y Prueba.
- **Centrado en la arquitectura:** El PUD asume que no existe un modelo único que cubra todos los aspectos del sistema. Por dicho motivo existen múltiples modelos y vistas que definen la arquitectura de software de un sistema, teniendo una percepción, tanto de la estática como de la dinámica de este.
- **Iterativo e Incremental:** El Proceso Unificado de Desarrollo es un marco de desarrollo iterativo e incremental compuesto de cuatro fases denominadas Inicio, Elaboración, Construcción y Transición. Cada una de estas fases es a su vez dividida en una serie de iteraciones. En cada una de las iteraciones los desarrolladores identifican y especifican los casos de uso relevantes, crean un diseño utilizando la arquitectura seleccionada como guía, implementan el diseño mediante componentes, y verifican que los componentes satisfacen los casos de uso. El resultado de cada iteración corresponde con un incremento del producto desarrollado que añade o mejora las funcionalidades del sistema en desarrollo. Cada una de estas iteraciones se divide a su vez en una serie de disciplinas. Análisis de requisitos, Diseño, Implementación y Prueba. Aunque todas las iteraciones suelen incluir trabajo en casi todas las disciplinas, el grado de esfuerzo dentro de cada una de ellas varía a lo largo del proyecto.

4.2. Tareas Preparativas

Antes de pasar a desarrollar las iteraciones propias del PUD, hemos visto necesario valernos de un escenario propio para el desarrollo del proyecto.

Por esta razón, en primer lugar se ha pasado a crear el entorno empresarial necesario. Esto conlleva desde la formación de una pequeña empresa hasta la creación de un lugar de trabajo para ella.

En un segundo lugar, hemos visto conveniente tener todo el sistema base preparado con el entorno de desarrollo adecuado para no interrumpir o molestar el proceso del proyecto.

Todas las tareas que hemos visto necesarias llevar a cabo antes de desarrollo propiamente dicho, junto con el diagrama de Gantt correspondiente a dichas tareas, son las mostradas en la siguiente tabla y diagrama.

Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
18		Iteraciones PUD	70 días	jue 25/10/12	vie 01/02/13		
19		Especificación de cada iteración	11 días	vie 02/11/12	vie 16/11/12		
20		Establecimientos de requisitos	1,75 días	vie 02/11/12	lun 05/11/12	1CC	
21		Obtención y revisión de requisitos	3 horas	vie 02/11/12	vie 02/11/12		Cristian Del Cerro;Javier García;Laura Del Rio;Raul Reguillo
22		Especificación de Casos de Uso	10 horas	vie 02/11/12	lun 05/11/12	21	Cristian Del Cerro;Javier García;Laura Del Rio;Raul Reguillo
23		Análisis de Casos de Uso	1,75 días	mar 06/11/12	mié 07/11/12	20	
24		Revisión de Clases de	3 horas	mar 06/11/12	mar 06/11/12		
25		Identificación de Clases asociadas a un Caso de Uso	5 horas	mar 06/11/12	mié 07/11/12	24	Raul Reguillo;Javier García;Laura Del Rio;Cristian Del Cerro
26		Descripción de la Interacción de	3 horas	mié 07/11/12	mié 07/11/12	25	Javier García;Cristian Del Cerro;Laura Del Rio;Raul Reguillo
27		Análisis de Clases	2,13 días	jue 08/11/12	lun 12/11/12	23	
28		Revisión de Clases	2 horas	jue 08/11/12	jue 08/11/12		
29		Identificación de responsabilidades y atributos	6 horas	jue 08/11/12	vie 09/11/12	28	Cristian Del Cerro;Javier García
30		Identificación de Asociaciones y Agregaciones	5 horas	vie 09/11/12	lun 12/11/12	29FC-5 horas	Laura Del Rio;Raul Reguillo
31		Definición de interfaces de usuario	0,63 días	lun 12/11/12	lun 12/11/12	27	
32		Especificación de inte	5 horas	lun 12/11/12	lun 12/11/12		Laura Del Rio;Javier García;Cristian Del Cerro;Raul R
33		Diseño de Clases	0,88 días	lun 12/11/12	mar 13/11/12	27	
34		Diseño e Identificación para la realización de las	7 horas	lun 12/11/12	mar 13/11/12		Cristian Del Cerro;Raul Reguillo;Javier García;Laura Del Rio
35		Generación del código de los componentes y procedimientos	2,63 días	mar 13/11/12	vie 16/11/12	14;33	Cristian Del Cerro;Javier García;Laura Del Rio;Raul Reguillo
36		Generación del código de	21 horas	mar 13/11/12	vie 16/11/12		
37		Ejecución de las pruebas y test	0,63 días	vie 16/11/12	vie 16/11/12	35	
38		Ejecución de las pruebas unitarias	5 horas	vie 16/11/12	vie 16/11/12		Javier García;Cristian Del Cerro
39		Ejecución de las pruebas de integración	4 horas	vie 16/11/12	vie 16/11/12		Raul Reguillo;Laura Del Rio
40		H- Fin de iteración	0 horas	vie 16/11/12	vie 16/11/12	37	

Figura 4.1: Lista de tareas para la creación de la empresa e instalación del sistema

En el Anexo A se encuentra una descripción detallada de la creación de la empresa.

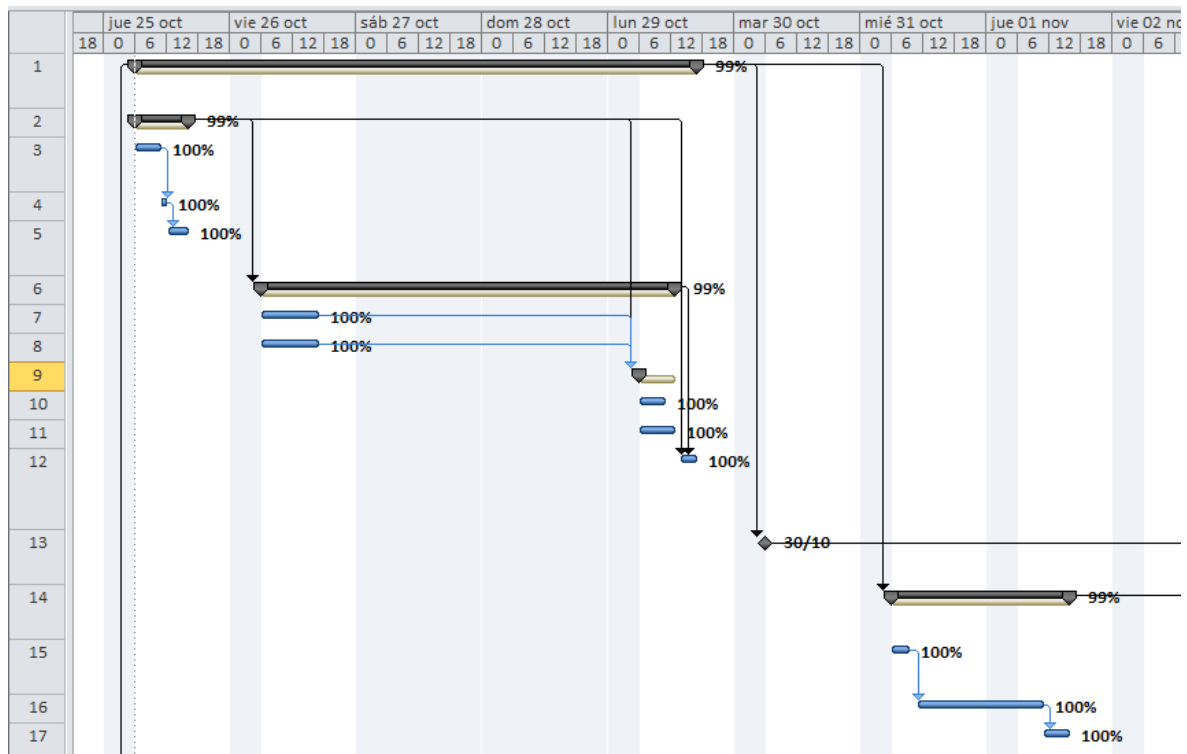


Figura 4.2: Diagrama de Gantt correspondiente a las tareas de creación de la empresa e instalación del sistema

4.3. Estructura e Iteraciones del PUD

El modelo PUD se estructura en ciclos, dando lugar en cada uno de ellos a una versión distinta del producto, en nuestro caso podemos diferenciar claramente dos ciclos ya que se ha realizado una mejora del producto tras la entrega de la primera versión al cliente.

A su vez, cada ciclo ha tenido diferentes fases (inicio, elaboración, construcción y transición) que a su vez se componen de iteraciones que abarcan las disciplinas de análisis, diseño, implementación y pruebas.

Las iteraciones llegadas a cabo para la realización completa del proyecto se han propuesto como tareas periódicas cada 10 a 13 días, variando de unas a otras tan solo el tiempo dedicado para cada una de las tareas.

Como se puede apreciar, el sistema y modelo que se quiere seguir se ve y planifica claramente con el Microsoft Project declarado para este proyecto.

En él hemos detallado las tareas que forman generalmente todas las iteraciones; sin embargo, solamente se ha detallado la duración, fecha de comienzo y fecha de fin.

Esto es debido a que las demás iteraciones se pueden tomar como base la primera cambiando los tiempos y duraciones. Con respecto al nombre de los recursos del personal requerido para cada iteración, no se ha especificado la correspondencia de cada uno ya que se han

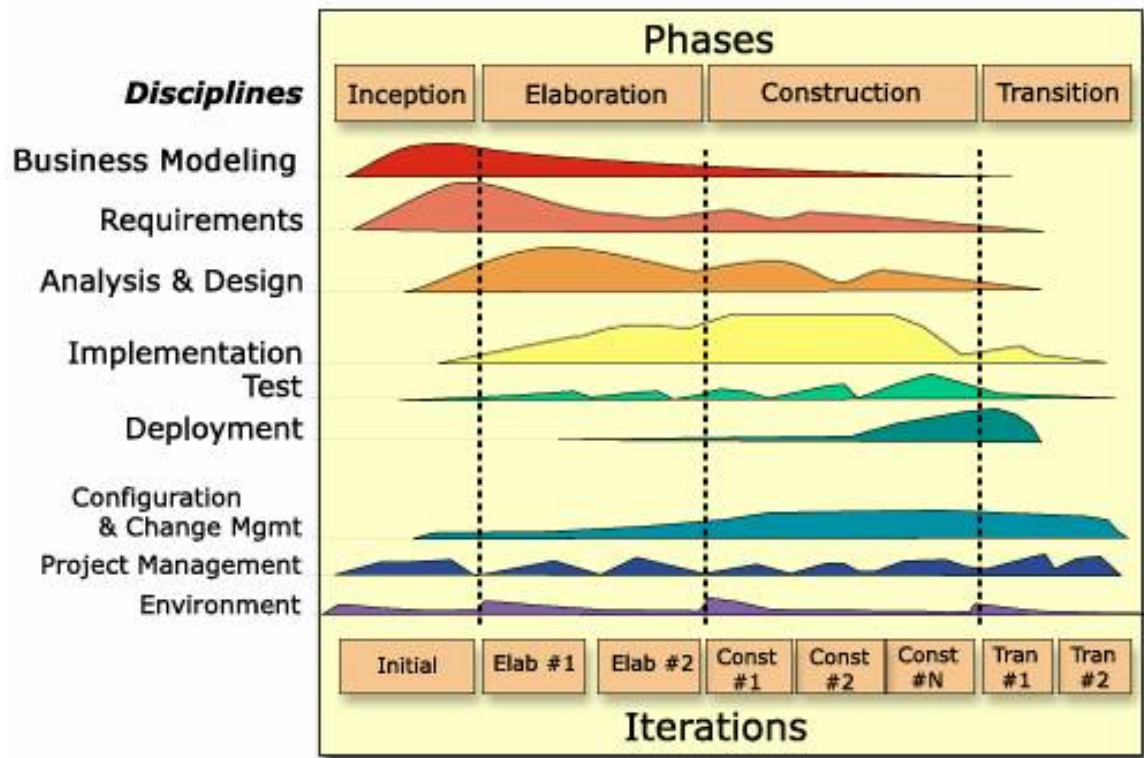


Figura 4.3: Esquema de esfuerzo siguiendo Proceso Unificado de Desarrollo

		Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1			+ Creación de la Empresa "Bugging and Debugging"	2,75 días	jue 25/10/12	lun 29/10/12	
13	✓		Hini - Hito Componente organizacional	0 mins	mar 30/10/12	mar 30/10/12	1
14			+ Instalación del sistema	1,63 días	mié 31/10/12	jue 01/11/12	1
18			- Iteraciones PUD	70 días	jue 25/10/12	vie 01/02/13	
19			+ Especificación de cada iteración	11 días	vie 02/11/12	vie 16/11/12	
41	✓		Iteraciones creación de empresa	4 días	jue 25/10/12	mar 30/10/12	
42	✓		Iteraciones de Instalación de sistema	2 días	mié 31/10/12	jue 01/11/12	
43	✓		Iteraciones PUD 1	11 días	vie 02/11/12	vie 16/11/12	
44	✓		Iteraciones PUD 2	10 días	lun 19/11/12	vie 30/11/12	
45	✓		Iteraciones PUD 3	10 días	lun 03/12/12	vie 14/12/12	
46	✓		Iteraciones PUD 4	10 días	lun 17/12/12	vie 28/12/12	
47	✓		Iteraciones PUD 5	8 días	lun 31/12/12	vie 11/01/13	
48	✓		Iteraciones PUD 6	9 días	lun 14/01/13	jue 24/01/13	
49	✓		Iteraciones PUD 7	6 días	vie 25/01/13	vie 01/02/13	
50	✓		Hfin - Hito Entrega Proyecto/ prototipo	30 mins	dom 03/02/13	dom 03/02/13	13;40

Figura 4.4: Lista de iteraciones y tareas generales de todo el proyecto

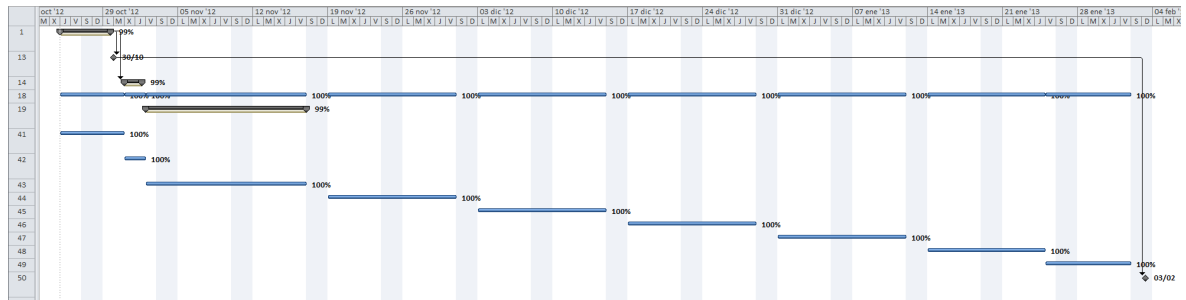


Figura 4.5: Diagrama de Gantt correspondiente a las tareas e iteraciones de todo el proyecto

se organizar para que serán rotativos en cada iteración, con lo que hemos visto intuitivo y simbólico detallar todos los recursos que pueden encargarse de esa tarea.

Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
18		Iteraciones PUD	70 días	jue 25/10/12	vie 01/02/13		
19		Especificación de cada iteración	11 días	vie 02/11/12	vie 16/11/12		
20		Establecimientos de requisitos	1,75 días	vie 02/11/12	lun 05/11/12	1CC	
21		Obtención y revisión de requisitos	3 horas	vie 02/11/12	vie 02/11/12		Cristian Del Cerro;Javier García;Laura Del Rio;Raul Reguillo
22		Especificación de Casos de Uso	10 horas	vie 02/11/12	lun 05/11/12	21	Cristian Del Cerro;Javier García;Laura Del Rio;Raul Reguillo
23		Análisis de Casos de Uso	1,75 días	mar 06/11/12	mié 07/11/12	20	
24		Revisión de Clases de	3 horas	mar 06/11/12	mar 06/11/12		
25		Identificación de Clases asociadas a un Caso de Uso	5 horas	mar 06/11/12	mié 07/11/12	24	Raul Reguillo;Javier García;Laura Del Rio;Cristian Del Cerro
26		Descripción de la Interacción de	3 horas	mié 07/11/12	mié 07/11/12	25	Javier García;Cristian Del Cerro;Laura Del Rio;Raul Reguillo
27		Análisis de Clases	2,13 días	jue 08/11/12	lun 12/11/12	23	
28		Revisión de Clases	2 horas	jue 08/11/12	jue 08/11/12		
29		Identificación de responsabilidades y atributos	6 horas	jue 08/11/12	vie 09/11/12	28	Cristian Del Cerro;Javier García
30		Identificación de Asociaciones y Agregaciones	5 horas	vie 09/11/12	lun 12/11/12	29FC-5 horas	Laura Del Rio;Raul Reguillo
31		Definición de interfaces de usuario	0,63 días	lun 12/11/12	lun 12/11/12	27	
32		Especificación de inte	5 horas	lun 12/11/12	lun 12/11/12		Laura Del Rio;Javier García;Cristian Del Cerro;Raul Re
33		Diseño de Clases	0,88 días	lun 12/11/12	mar 13/11/12	27	
34		Diseño e Identificación para la realización de las	7 horas	lun 12/11/12	mar 13/11/12		Cristian Del Cerro;Raul Reguillo;Javier García;Laura Del Rio
35		Generación del código de los componentes y procedimientos	2,63 días	mar 13/11/12	vie 16/11/12	14;33	Cristian Del Cerro;Javier García;Laura Del Rio;Raul Reguillo
36		Generación del código de	21 horas	mar 13/11/12	vie 16/11/12		
37		Ejecución de las pruebas y test	0,63 días	vie 16/11/12	vie 16/11/12	35	
38		Ejecución de las pruebas unitarias	5 horas	vie 16/11/12	vie 16/11/12		Javier García;Cristian Del Cerro
39		Ejecución de las pruebas de integración	4 horas	vie 16/11/12	vie 16/11/12		Raul Reguillo;Laura Del Rio
40		H- Fin de iteración	0 horas	vie 16/11/12	vie 16/11/12	37	

Figura 4.6: Ejemplo de especificación detallada de las tareas que forman cada iteración

En las primeras iteraciones, más concretamente las dos primeras, se ha conseguido una lista vaga de requisitos que más adelante se irían revisando y completando.

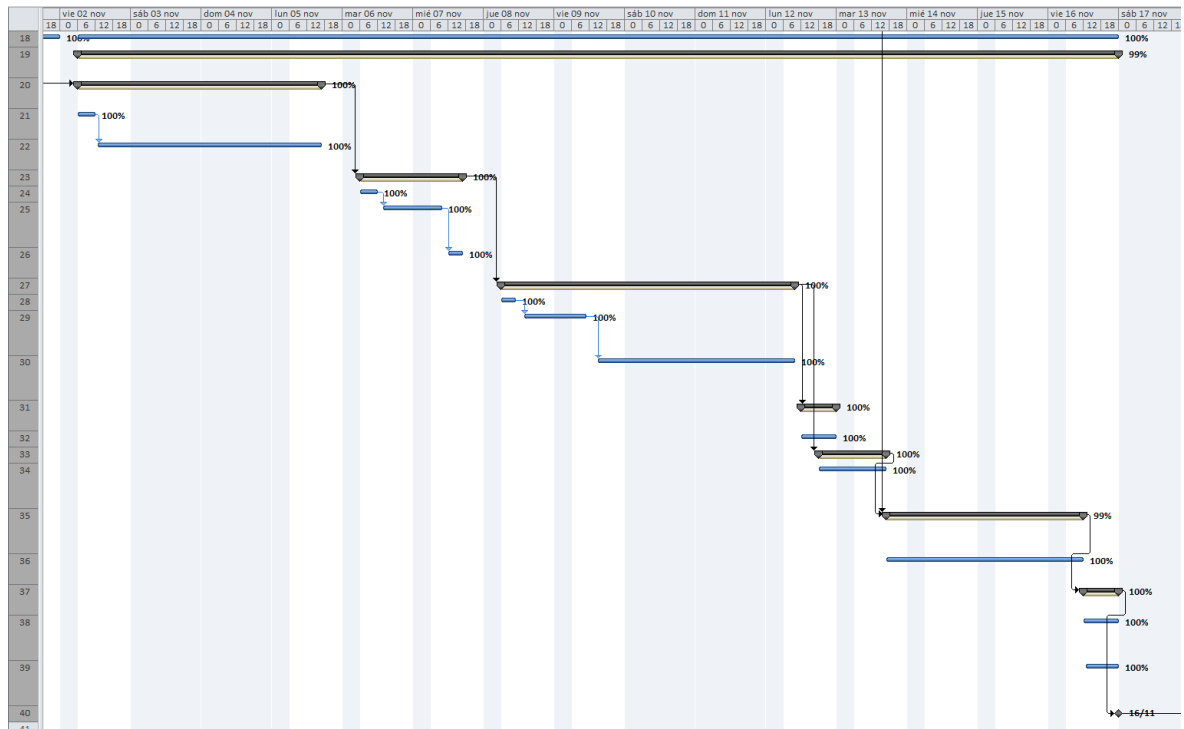


Figura 4.7: Diagrama de Gantt correspondiente al ejemplo de especificación detallada de las tareas de cada iteración

También se ha tomado decisiones de análisis y diseño del producto, la red social, en general. Se ha conseguido definir un poco el ámbito del sistema, logando conseguir los analizar, diseñar y desarrollar los primeros casos de uso a niveles altos de abstracción, como son el Login y registro de un usuario al sistema. A este grupo de las primeras iteraciones se le podría categorizar como la fase de Inicio Las siguientes iteraciones corresponderían con la fase de elaboración; en estas dos iteraciones se han refinado más los requisitos, ya que se ha consultado y mirado cada una de ellas con el cliente, Macario Polo Usaola. Se ha desarrollado el esqueleto de la red social en general, sin especificar a fondo. Y por último se a refinado la implementación de los casos de uso de la fase anterior, también realizándoles pruebas y analizando, diseñando y desarrollando nuevos casos de usos, como el poder subir un artículo al sistema para que un revisor pueda revisarlo. Las iteraciones número 5 y 6 son aquellas las cuales tienen más peso el desarrollo de la red social por completo, realizándole y creándole test para realizar pruebas al sistema. En estas iteraciones también se lleva por completo el desarrollo en más bajo nivel a los casos de uso que lo necesiten. Para terminar, la última iteración nos servirá para finalizar la red social convirtiéndola en la primera versión del producto. Para ello primero en esta iteración realizamos por completo todas las pruebas y testeos necesarios para probar por completo el sistema.

En el caso que encontremos algún tipo de problema o defecto debemos cubrirlo para poder realizar la entrega del producto. Estas pruebas se detallarán más adelante en el apartado de

Pruebas de este mismo documento.

Capítulo 5

Marco Tecnológico

5.1. Java

El proyecto se ha desarrollado utilizando la tecnología Java junto con *Struts2* usando el patrón MVC. Es un patrón de arquitectura de las aplicaciones software que separa la lógica de negocio de la interfaz de usuario. Facilita la evolución por separado de las capas e incrementa la reutilización y flexibilidad.

Para la primera parte del dominio se creó un proyecto *Java*, que es la base del proyecto web dinámico creado después para añadir la capa web con *Struts2*.

El entorno de desarrollo utilizado ha sido *Eclipse* en su versión *Juno* puesto que la versión facilitada en campus virtual no funcionaba correctamente con la última versión de *Apache Tomcat*.

El plugin usado para sincronizar con el repositorio de *Google Code* lo hemos instalado a partir de esta url <http://www.polarion.org/projects/subversive/download/1.1/update-site/>

Más información sobre Java: [Mica]

Más información sobre Eclipse: [Fouc]

5.2. Patrones

En la práctica se ha hecho uso de tres patrones de diseño citados en [GHJV96]:

- Singleton
- DAO
- Abstract Factory

5.2.1. Patrón *Singleton*

Diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella. En nuestro caso esa clase es la

clase Agente que da acceso a la base de datos, no nos interesa más de un punto de acceso a la base de datos y este patrón nos facilita este cometido de forma sencilla.

Al utilizar el agente se comprueba si ya existe una instancia, si existe se devuelve, y si no es así se crea una nueva que será la que retorne.

5.2.2. Patrón DAO

DAO: Data Access Object (Objeto de Acceso a Datos) es un componente software que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos, tales como una Base de datos o un archivo. En nuestro caso disponemos de una interfaz DAO (como por ejemplo, UserDAO) para cada clase del dominio que utilice acceso a base de datos mediante el agente. Entre el agente y las interfaces disponemos de una clase abstracta que implementa estas interfaces (AgentFactory) , los agentes heredan de la clase abstracta e implementan todos los métodos. En nuestro caso solo tenemos un agente.

Debido a la naturaleza del proyecto, donde los objetos que se crean son muy concretos y no generalmente siguen un ciclo de vida CRUD, no se ha aprovechado toda la potencia del patrón DAO, reduciéndolo a un conjunto de interfaces que se fuerza a las clases Agente a implementar.

5.2.3. Patrón *Abstract Factory*

Se crean diferentes familias de objetos, todos pertenecientes a un mismo perfil. Disponemos así de una clase abstracta (AgentFactory) la cual tiene todos los métodos a implementar por las subclases (distintos Agentes), de forma que los agentes se comporten de igual manera y se interactúe con ellos de manera transparente a su naturaleza. Esto da la opción de variar la arquitectura, en este caso, la base de datos (actualmente implementada en MySQL).

De nuevo, dada la naturaleza del proyecto, no es necesario establecer qué factoría usar en tiempo de ejecución (lo que sí hubiese resultado útil en otro tipo de problemas, tales como renderizado de interfaces o similar). Así pues, indicada la factoría abstracta de Agentes, se ha heredado de ella una única factoría genérica (Agente) que es la que gestionará en este caso el acceso a datos en MySQL.

5.3. Struts2

Struts 2: es, como el nombre sugiere, la nueva versión del popular framework de desarrollo web en Java Apache Struts.

Struts 2 está basado en el patrón MVC (Modelo-Vista-Controlador), una arquitectura que busca reducir el acoplamiento dividiendo las responsabilidades en 3 capas claramente diferenciadas:

- El **modelo**, que hace referencia a los datos que maneja la aplicación y las reglas de negocio que operan sobre ellos y que se traducen en Struts 2 mediante acciones
- La **vista**, encargada de generar la interfaz con la que la aplicación interacciona con el usuario
- El **controlador**, que comunica la vista y el modelo respondiendo a eventos generados por el usuario en la vista, invocando cambios en el modelo, y devolviendo a la vista la información del modelo necesaria para que pueda generar la respuesta adecuada para el usuario

Se ha implementado una acción de Struts 2 para cada caso de uso que se quiere implementar. De esta manera identificamos una parte específica de la web con una acción concreta. En algunas acciones además del método `execute()` utilizamos también métodos auxiliares debido a decisiones de diseño: damos prioridad a la semántica del modelo englobando en una acción un conjunto de operaciones que estén relacionadas.

Más información sobre Struts 2: [Foua]

5.4. Apache Tomcat

Para el desarrollo del proyecto se ha decidido utilizar Apache Tomcat 7.0. Apache Tomcat es un servidor web multiplataforma que funciona como contenedor de servlets y que se desarrolla bajo el proyecto denominado Jakarta perteneciente a la Apache Software Foundation bajo la licencia Apache 2.0 y que implementa las especificaciones de los servlets y de JavaServer Pages o JSP de Sun Microsystems. Por lo tanto, Tomcat puede funcionar como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad.

Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java, y por lo que no tendríamos problemas al manejar diferentes sistemas operativos. La principal característica que soporta la versión 7.0 utilizada en el proyecto son las siguientes:

- Implementado de Servlet 3.0 JSP 2.2 y EL 2.2
- Mejoras para detectar y prevenir fugas de memoria en las aplicaciones web
- Limpieza interna de código
- Soporte para la inclusión de contenidos externos directamente en una aplicación web
- Modo embebido simplificado

Al soportar la nueva especificación Servlet 3.0 también permite:

- Soporte asíncrono
- Configuración dinámica: fragmentos web (librerías pueden embeber partes de un `web.xml` de modo que no sea necesario añadirlos al `web.xml` de la aplicación

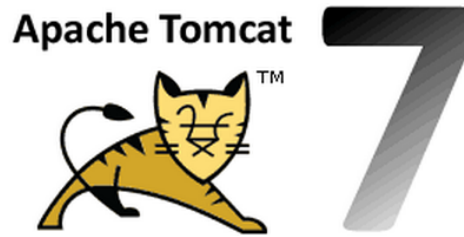


Figura 5.1: Logotipo Apache Tomcat 7

- Soporta anotaciones: los filtros, Servlets y Listeners pueden definirse por anotaciones, sin necesidad de crear un web.xml
- API Servlet extendida: permite añadir Servlets y Filter después del arranque de la aplicación
- Mejora del soporte de las sesiones
- Mejoras en Seguridad

Más información sobre Apache Tomcat: [Foub]

5.5. MySQL

MySQL es un sistema de gestión de bases de datos relacional. Su diseño multihilo le permite soportar una gran carga de forma muy eficiente. Las principales características de este gestor de bases de datos son las siguientes:

1. Aprovecha la potencia de sistemas multiprocesador, gracias a su implementación multihilo
2. Soporta gran cantidad de tipos de datos para los atributos
3. Dispone de API's en gran cantidad de lenguajes (C, C++, Java, PHP, etc)
4. Gran portabilidad entre sistemas
5. Soporta hasta 32 índices por tabla
6. Gestión de usuarios y passwords, manteniendo un muy buen nivel de seguridad en los datos

Para el manejo de MySQL se ha utilizado la herramienta *MySQL Workbench*.

MySQL Workbench es una herramienta visual de diseño de base de datos con una interfaz gráfica de usuario muy fácil de usar. Esta integra desarrollo de software, administración, diseño, creación y mantenimiento de bases de datos MySQL. *MySQL Workbench* permite realizar fácilmente operaciones como la configuración de los servidores, la administración



Figura 5.2: Logotipo MySQL

de usuarios, exportación e importación, y la visualización de los registros. Además, *MySQL Workbench* es totalmente gratuito en su versión Community.

La base de datos diseñada para el proyecto dispone de 14 tablas que lo describen completamente:

- User: Tabla que corresponde a todos los usuarios del sistema de la red social
- Autor: Tipo de usuario que se diferencia en su atributo karma que aumenta según la puntuación de los revisores a su artículo
- Reviewer: Tabla que corresponde a los usuarios de tipo revisor, estos tendrán un karma, que describirá la actividad de este y una etiqueta indicando si es un revisor VIP o no
- Publisher: Tipo de usuario que poseerá a diferencia de los demás más información en el sistema (journal y web)
- PublisherReviewer: Cada editorial podrá tener varios revisores asociados. Esto se describe en esta tabla
- Article: Tabla de la base de datos en la que se almacenará todos los artículos escritos. Estos serán insertados al sistema por los autores que participan en la red social, guardando con ellos su estado, el comentario realizado por su autor acerca de dicho artículo y la nota puesta por el revisor. El estado puede tener un valor comprendido entre los siguientes valores: sent, halfreviewed, reviewed, sentVIP y halfreviewedVIP
- ArticleReview: Relaciona un revisor con el artículo a revisar. Esta tabla posee dos atributos que se podrían confundir con atributos de la tabla artículo: *state_review*, no es idéntico al estado del artículo, si no que corresponde al estado de la revisión de dicho artículo (resolved, unresolved); y *comment*, se refiere a los comentarios que hace el revisor acerca del artículo, esto no es lo mismo que los comentarios que realiza el propio autor de su artículo
- Follows: La red social permite que un usuario pueda seguir a otro usuario dentro del sistema sin obligar a que ese seguimiento sea mutuo
- Message: Los usuarios pueden comunicarse entre ellos a través de mensajes que podrán ser enviados sabiendo solamente su e-mail

- Notification: Esta tabla guardará el historial de notificaciones de los usuarios en los casos de revisiones, mensajes y seguidores
- Tag: Tabla de las etiquetas que puede tener un artículo o un revisor dentro del sistema. Estos tags pueden tomar valores entre los siguientes: noticias, deportes, ciencias, ocio y opinión
- ArticleTag: Tabla de relación en la que describe que a un artículo le corresponde un tag
- ReviewerTag: Tabla de relación entre revisores y tags. A cada revisor le corresponde un tag dentro del sistema

El diagrama de la base de datos se puede encontrar en secciones posteriores.

Más información sobre MySQL: [Micb]

5.6. Virtualizaciones

5.6.1. Virtualización de la base de datos

A la hora de trabajar con la base de datos se ha elegido virtualizarla para poder desarrollar y administrar esta parte del proyecto de forma mucho más sencilla. Del mismo modo nos ha servido para desacoplar la capa de persistencia de la aplicación (donde se encuentra el almacenamiento de los datos) del resto de capas de la aplicación lo cual ayudará a que en un futuro puedan ampliarse los recursos del sistema permitiendo una mayor escalabilidad.

Ventajas de la virtualización:

- Una mayor flexibilidad y agilidad de la infraestructura informática existente
- Puesta en común y compartición de los recursos
- Simplificación de la administración y gestión
- Aumento de la tolerancia a fallos

5.6.2. Software usado para la virtualización

El programa de anfitrión elegido es *Virtualbox* de *Sun*, por ser gratuito bien conocido por todos los integrantes del equipo de desarrollo. Sobre éste, como sistema invitado, se instaló un *Ubuntu server* debidamente configurado con `mysql-server`. Se eligió éste y no otro por ser un entorno muy básico y ligero (sin ni siquiera interfaz gráfica) para intentar compensar la pérdida de rendimiento por estar el sistema virtualizado.

Más información sobre Virtual Box: [Ora]

5.6.3. Método de trabajo

Durante el trabajo individual de cada integrante del equipo de desarrollo virtualizaba su propia base de datos sobre la que iba a trabajar. Por otro lado en las reuniones de trabajo conjunto un miembro del equipo virtualizaba la base de datos y el resto se conecta a esa base de datos. Es una manera de simular una carga de trabajo más realista así como de hacer más dinámico el trabajo en grupo y evitar problemas de versiones o refinamientos de la base de datos.

Capítulo 6

Diseño de la Herramienta

Para abordar la implementación del proyecto de la red social se ha decidido que el diseño de la herramienta conste de tres capas (Modelo – Vista – Controlador), patrón MVC, en el que está basado Struts 2 (framework de desarrollo web que usamos). Al mismo tiempo utilizaremos una serie de patrones que se explicarán más adelante junto con los problemas y arreglos que se han debido efectuar en la segunda versión del producto.

6.1. Struts 2

Como se ha nombrado anteriormente, el desarrollo e implementación del producto se ha resuelto con el uso de Struts2. Struts 2 es, como el nombre sugiere, la nueva versión del popular framework de desarrollo web en Java Apache Struts. Struts 2 está basado en el patrón MVC (Modelo-Vista-Controlador), una arquitectura que busca reducir el acoplamiento dividiendo las responsabilidades en 3 capas claramente diferenciadas:

- **El modelo**, que hace referencia a los datos que maneja la aplicación y las reglas de negocio que operan sobre ellos y que se traducen en Struts 2 mediante acciones. En nuestra implementación se corresponde con la capa de *Persistencia*, que concierne con las clases utilizadas a lo largo de toda la implementación.
- **La vista**, encargada de generar la interfaz con la que la aplicación interacciona con el usuario. Esto corresponde a la capa *SocialNetwork.actions* y *WebContent* de nuestro código; estas conciernen las acciones que se realizan con su iteración y la presentación que debe mostrarse al usuario.
- **El controlador**, que comunica la vista y el modelo respondiendo a eventos generados por el usuario en la vista, invocando cambios en el modelo, y devolviendo a la vista la información del modelo necesaria para que pueda generar la respuesta adecuada para el usuario. Esta última capa trata de nuestra *Dominio*, en la que se desarrollan las comunicaciones siguiendo una serie de patrones de diseño.

Se ha implementado una acción de Struts 2 para cada caso de uso que se quiere implementar. De esta manera identificamos una parte específica de la web con una acción concreta. En algunas acciones además del método *execute()* utilizamos también métodos auxiliares debi-

do a decisiones de diseño: damos prioridad a la semántica del modelo englobando en una acción un conjunto de operaciones que estén relacionadas.

6.2. Patrones

En el proyecto se ha hecho uso de tres patrones de diseño:

- **Singleton**, con pool de conexiones
- **DAO**
- **Abstract Factory**

Todos ellos se pueden consultar en [GHJV96]

6.2.1. Patrón *Singleton* con pool de conexiones

Diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Su intención consiste en garantizar que una clase solo tenga una instancia y proporcionar un punto de acceso global a ella. En nuestro caso esa clase da acceso a la base de datos, no nos interesa más de un punto de acceso a la base de datos y este patrón nos facilita este cometido de forma sencilla. Al utilizar la conexión se comprueba si ya existe una instancia, si existe se devuelve, y si no es así se crea una nueva que sería la que retorne. Sin embargo, en la primera versión del producto no tratamos correctamente el pool de conexiones de la aplicación a nuestra base de datos. Un pool de conexiones es un conjunto de conexiones abiertas disponibles para ser utilizadas al momento de hacer una consulta. Si tenemos una clase que al instanciarse se conecta a una base de datos en un sistema multitarea podemos llegar a ocupar todo el pool haciendo que otras peticiones no puedan acceder a la base de datos, además estaremos ocupando memoria en la aplicación por cada instancia creada y memoria en el servidor por cada conexión utilizada. Por lo tanto, una buen solución y muy común a esta problemática es que en el patrón de diseño Singleton le añadamos un bróker para nuestro pool de conexiones. Esto corresponde las nuevas clases Conexión.java y Broker.java.

6.2.2. Patrón DAO

DAO: Data Access Object (Objeto de Acceso a Datos) es un componente software que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos, tales como una Base de datos o un archivo. En nuestro caso disponemos de una interfaz DAO (como por ejemplo, UserDAO) para cada clase del dominio que utilice acceso a base de datos mediante el broker. Todas las clases DAOs de la persistencia se basan en la estructura y naturaleza de CRUD y heredan de ella. Esta estructura es la que muestra la clase abstracta CrudDAO.java de nuestra implementación (este fragmento de código se mostrará

en la finalización del próximo punto). En la versión anterior del producto, teníamos un solo agente que se conectaba a la base de datos el cual heredaba de las clases DAOs abstractas e implementaba todos los métodos. Por lo tanto, no se seguirá un ciclo de vida CRUD, no pudiendo aprovechar toda la potencia de este patrón. Esto conlleva a que en la nueva versión de la red social, se ha tenido en cuenta la naturaleza CRUD y por lo tanto, se ha reestructurado todas las conexiones a la base de datos para que así teniendo cada conexión con la clase DAO correspondiente.

6.2.3. Patrón *Abstract Factory*

Se crean diferentes familias de objetos, todos pertenecientes a un mismo perfil. Disponemos así de una clase abstracta (CrudDAO) la cual tiene todos los métodos a implementar por las subclases (distintas clases DAO), de forma que todas las clases DAO se comporten de igual manera y se interactúe con ellos de manera transparente a su naturaleza. Esto da la opción de variar la arquitectura, en este caso, la base de datos (actualmente implementada en MySQL).

```
package Persistence;

import java.sql.SQLException;
import java.util.LinkedList;

/**
 * DAO (J2EE Pattern)
 *
 * @param object type
 */
public abstract class CrudDAO<T> {

    public abstract T read(T obj) throws SQLException;

    public abstract void create(T obj) throws SQLException;

    public abstract void update(T obj) throws SQLException;

    public abstract void delete(T obj) throws SQLException;

    public abstract LinkedList<T> readAll(T obj) throws SQLException;
}
```

Listado 6.1: Implementación de CrudDAO.java

6.3. Diseño y diagramas

A continuación, se despliega parte de la información utilizada durante la implementación de la red social; principalmente diagramas de Casos de Uso y Clases, así como un modelo de la base de datos utilizada. A continuación, se despliega parte de la información utilizada durante la implementación de la red social, los diagramas de Casos de Uso y de Clases resultantes a las iteraciones descritas anteriormente, que abarcan el funcionamiento y comportamiento del sistema de la red social. Asimismo se mostrará seguidamente el esquema de la base de datos usada en el sistema.

Para mayor claridad, se incluyen los diagramas en alta definición en un anexo aparte a este documento, reflejando aquí algunos.

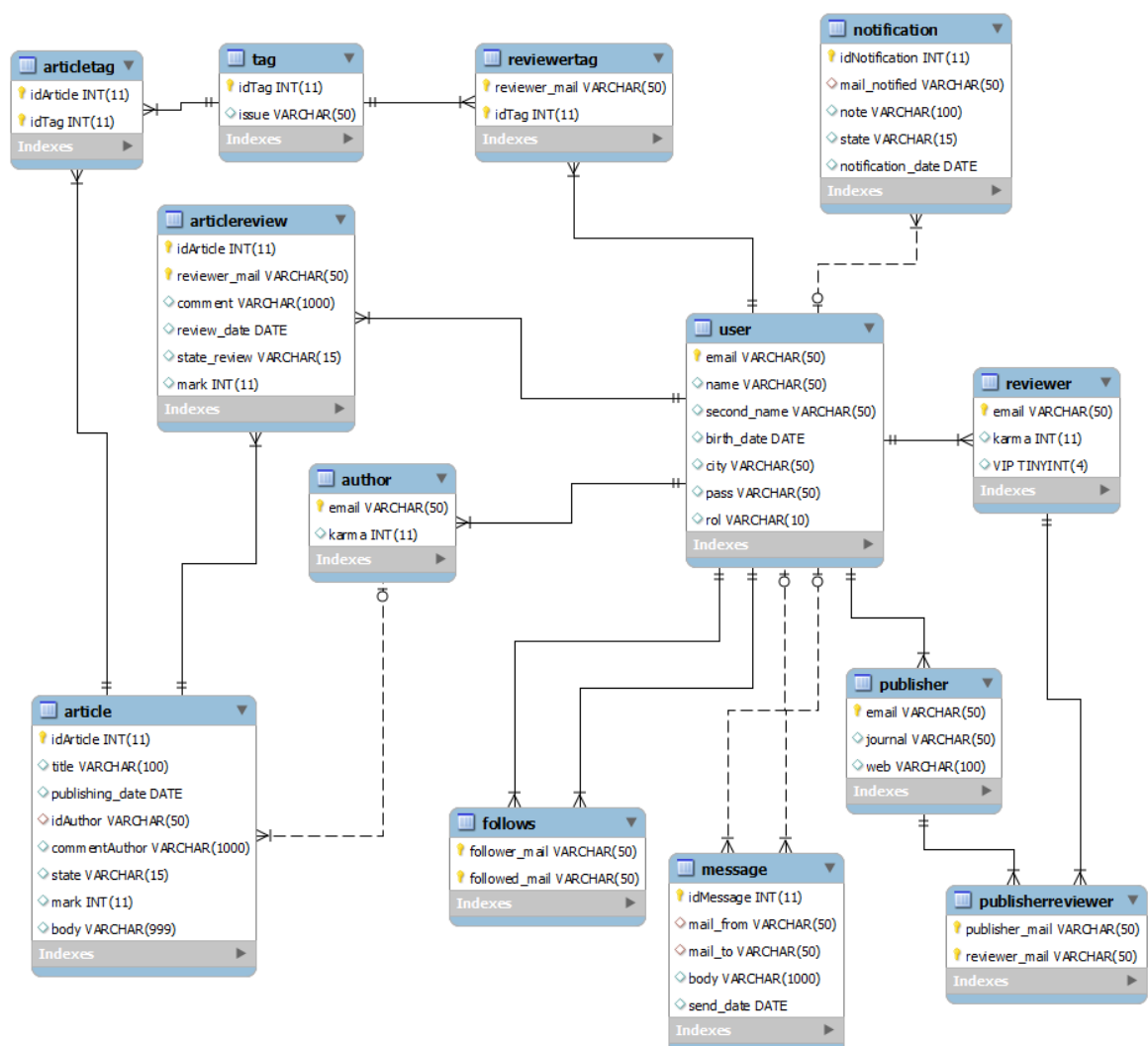


Figura 6.1: Modelo de la Base de Datos

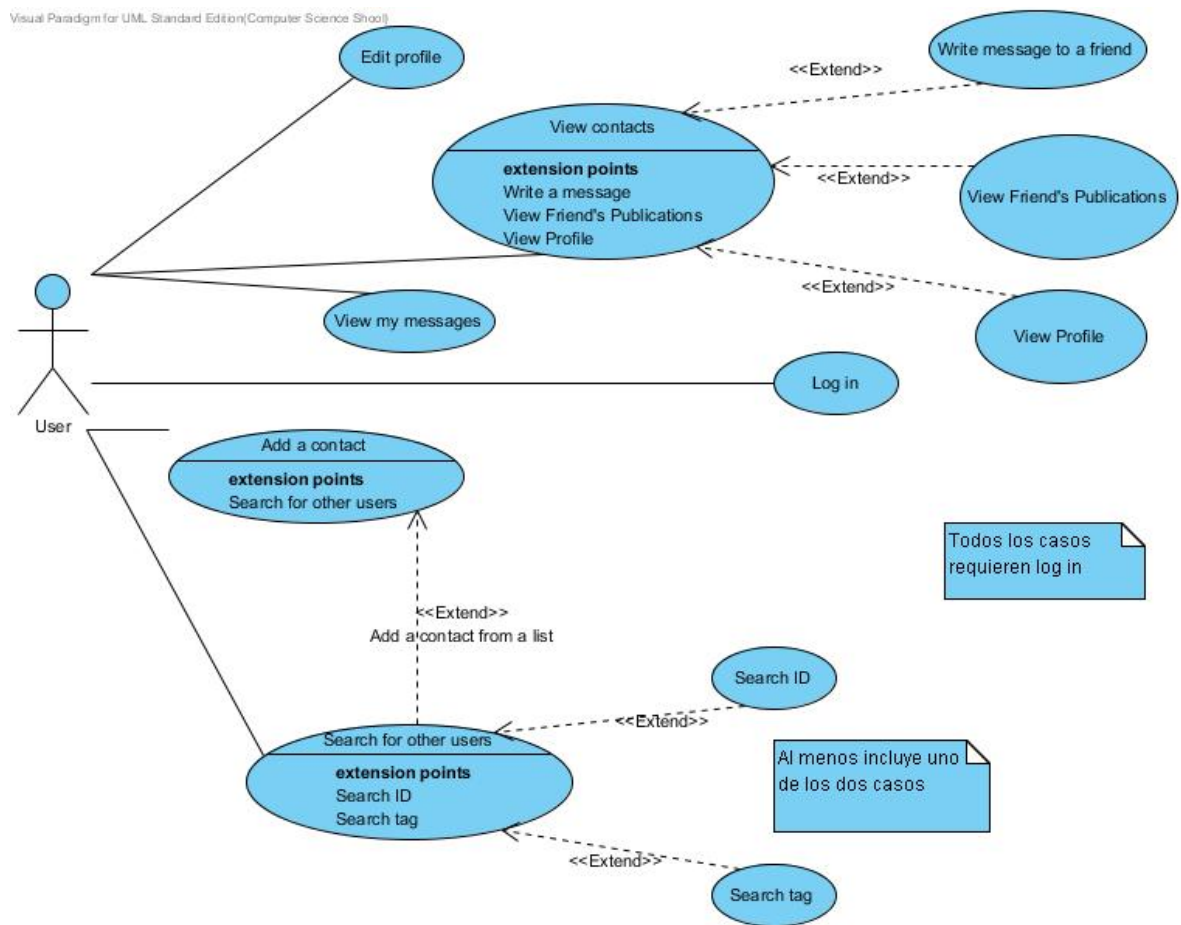


Figura 6.2: Caso de Uso: Usuario

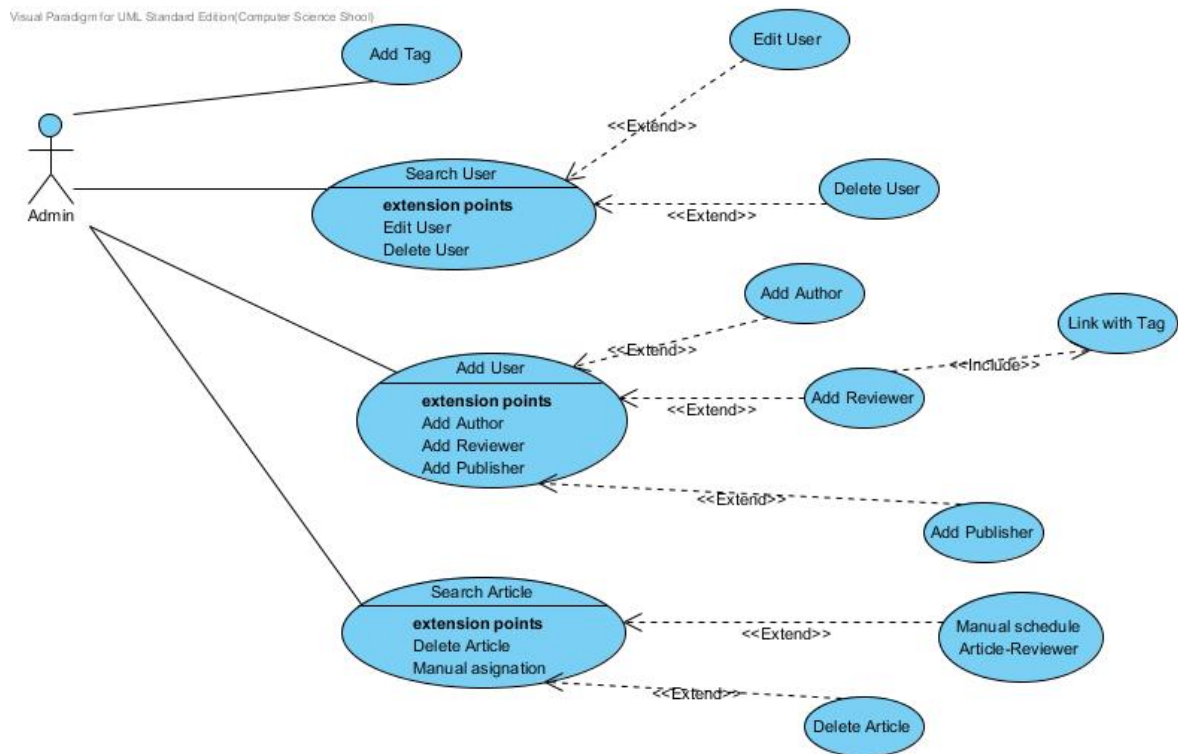


Figura 6.3: Caso de Uso: Admin

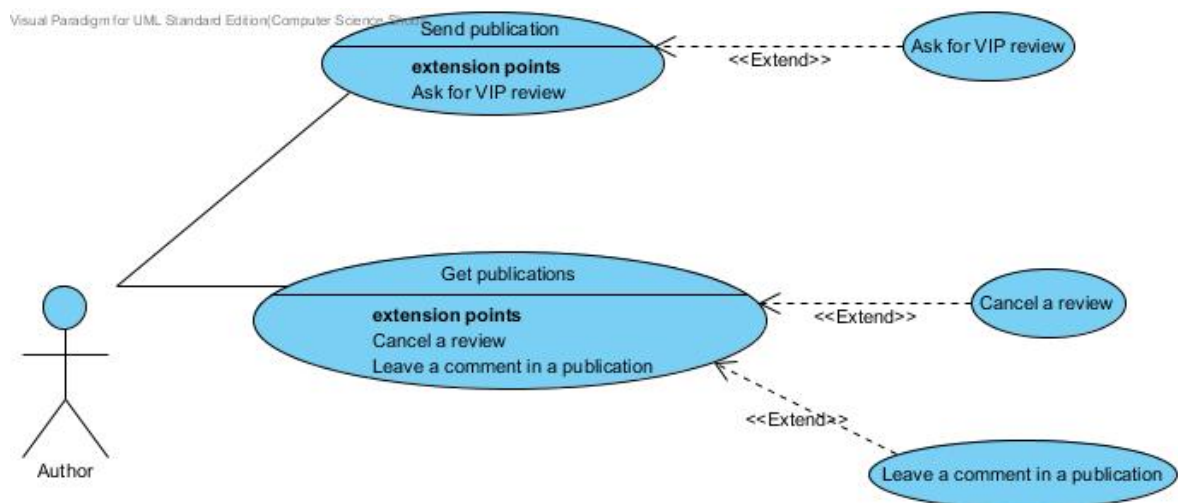


Figura 6.4: Caso de Uso: Autor

Visual Paradigm for UML Standard Edition(Computer Science School)

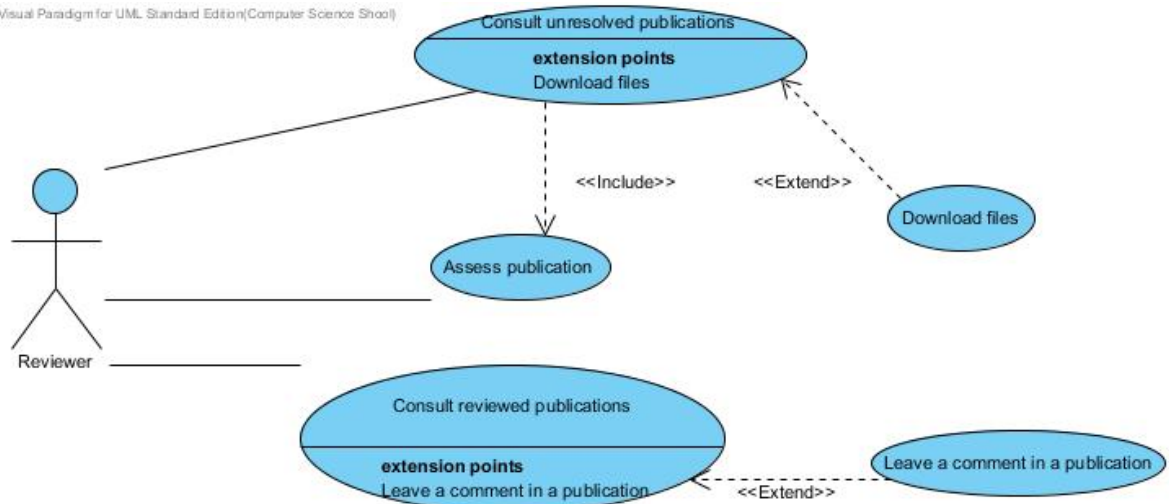


Figura 6.5: Caso de Uso: Revisor

Visual Paradigm for UML Standard Edition(Computer Science School)

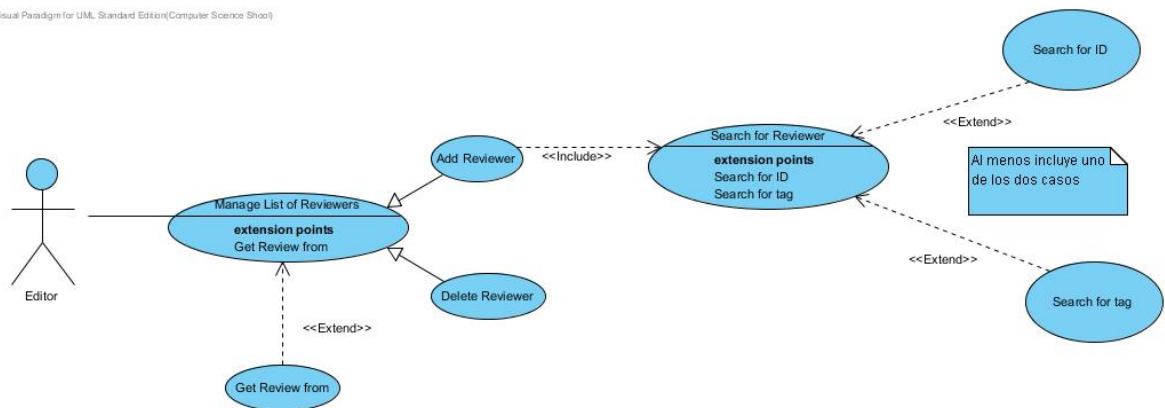


Figura 6.6: Caso de Uso: Editor

Visual Paradigm for UML Standard Edition(Computer Science School)



Figura 6.7: Diagrama de Clases: Árbol de herencia

Visual Paradigm for UML Standard Edition(Computer Science School)

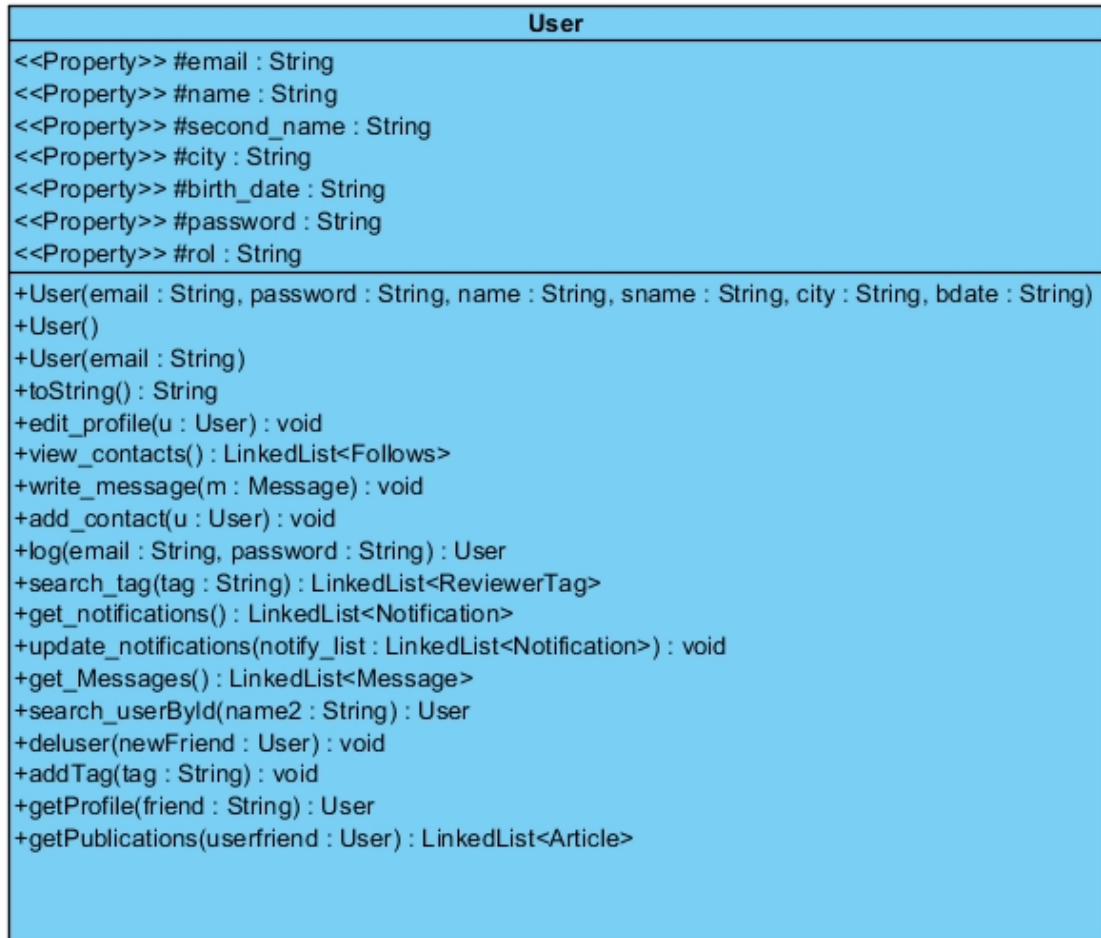


Figura 6.8: Diagrama de Clases: Usuario

Visual Paradigm for UML Standard Edition(Computer Science Shool)

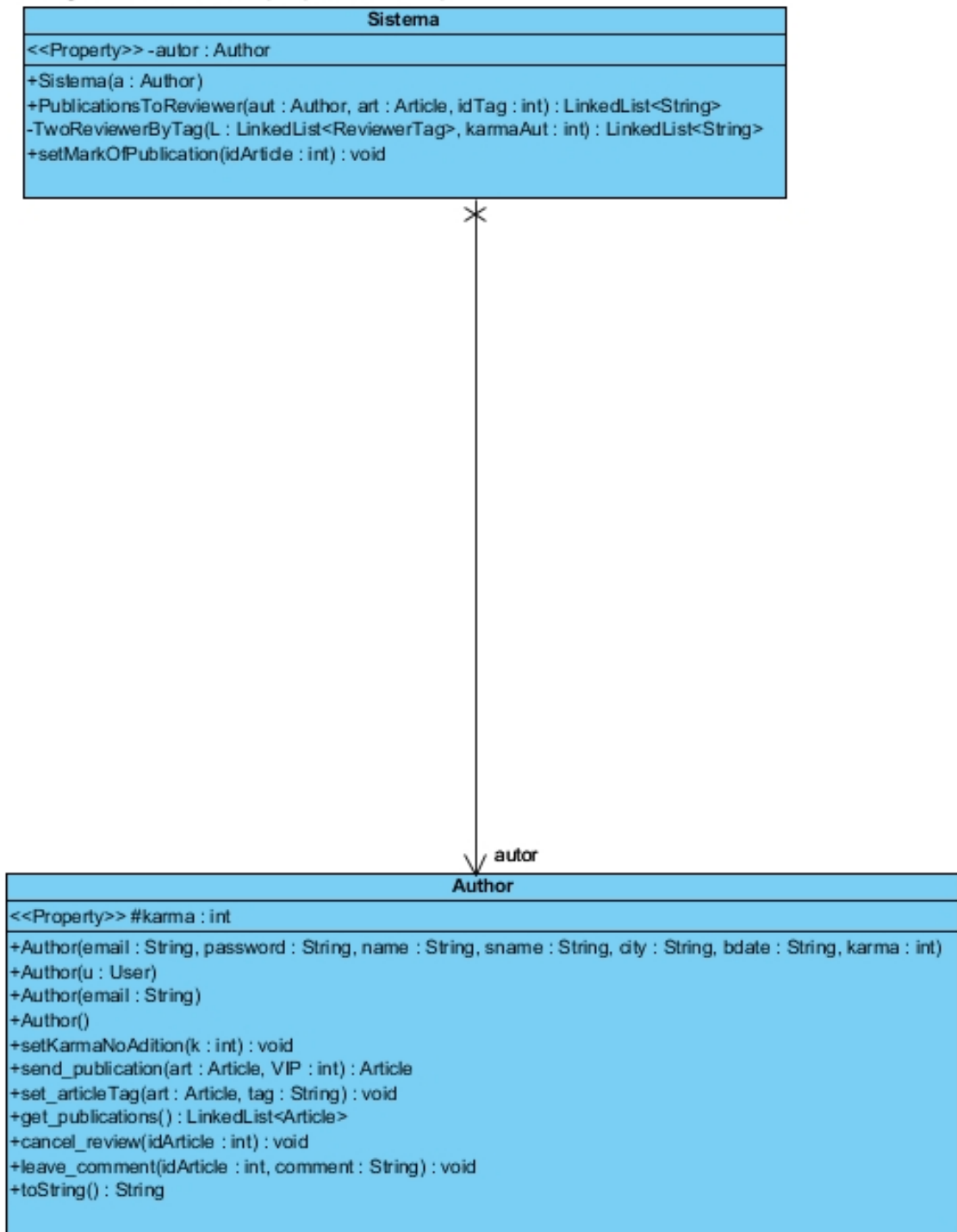


Figura 6.9: Diagrama de Clases: Autor y Sistema

Visual Paradigm for UML Standard Edition(Computer Science Shool)

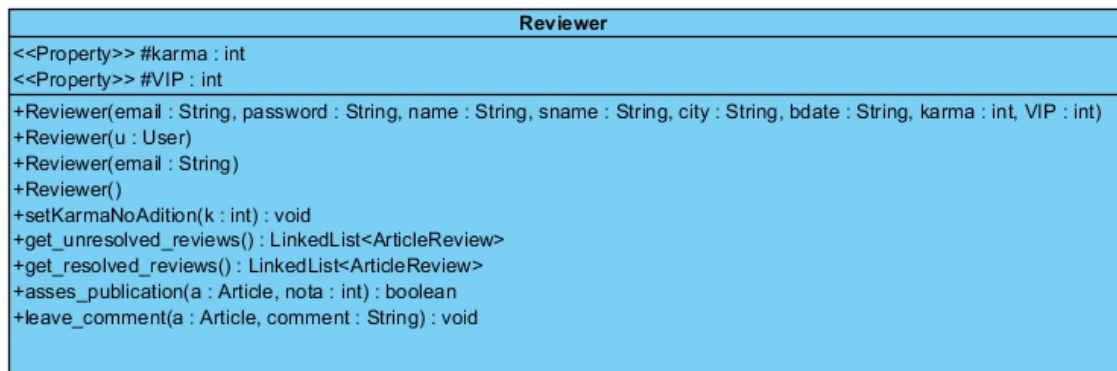


Figura 6.10: Diagrama de Clases: Revisor

Visual Paradigm for UML Standard Edition(Computer Science Shool)

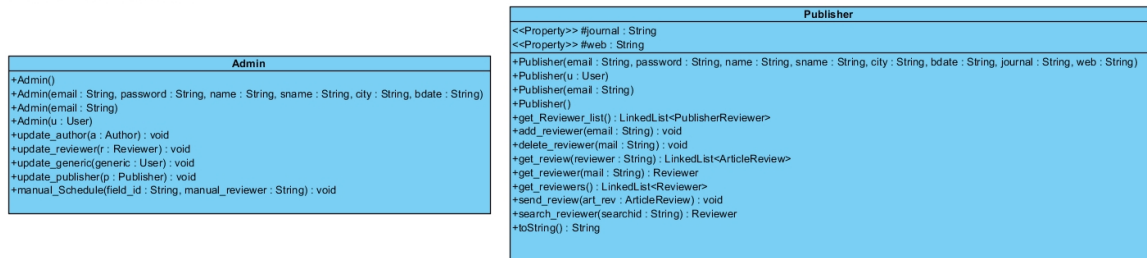


Figura 6.11: Diagrama de Clases: Admin y Editor

Visual Paradigm for UML Standard Edition(Computer Science Shool)

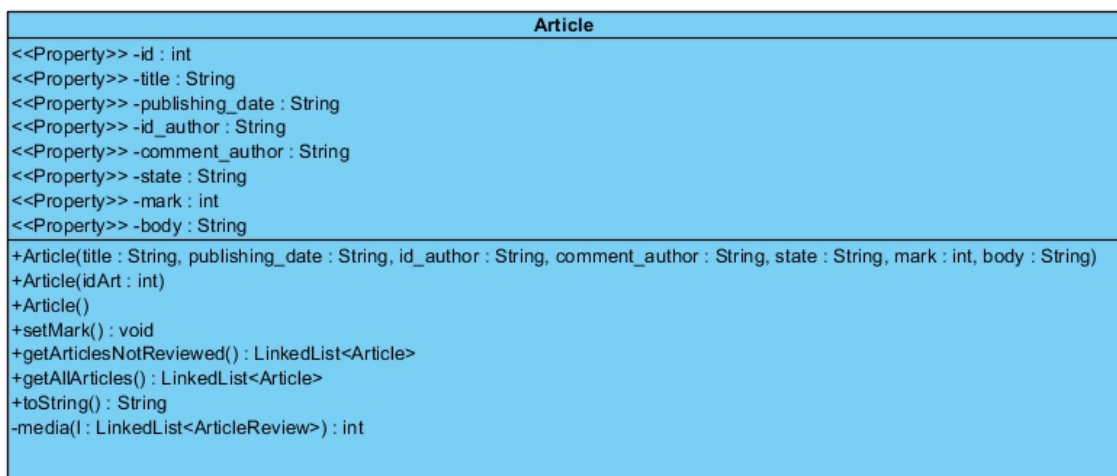


Figura 6.12: Diagrama de Clases: Artículo

Visual Paradigm for UML Standard Edition(Computer Science School)

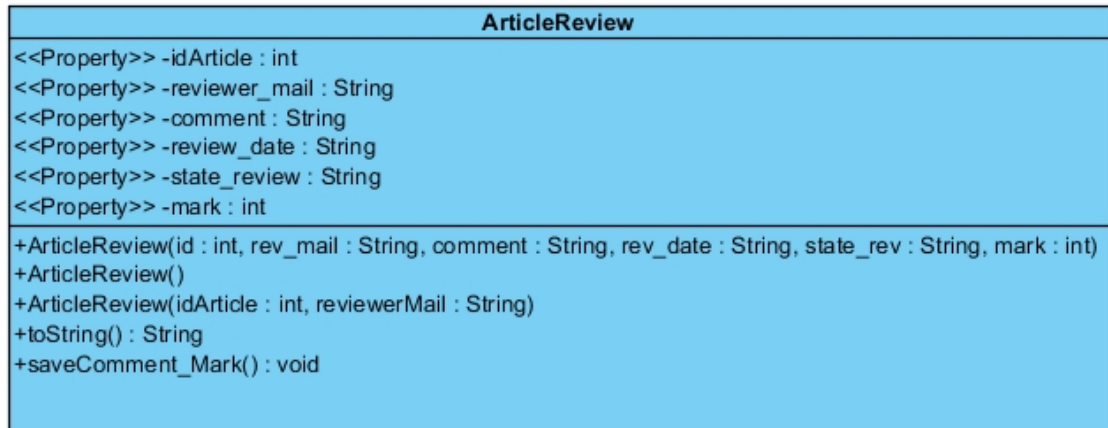


Figura 6.13: Diagrama de Clases: Revisión de Artículo

Visual Paradigm for UML Standard Edition(Computer Science School)

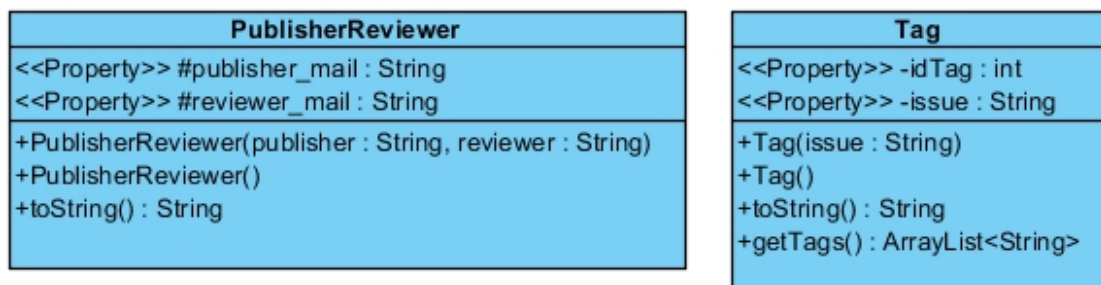


Figura 6.14: Diagrama de Clases: Editor con Revisor y Tag

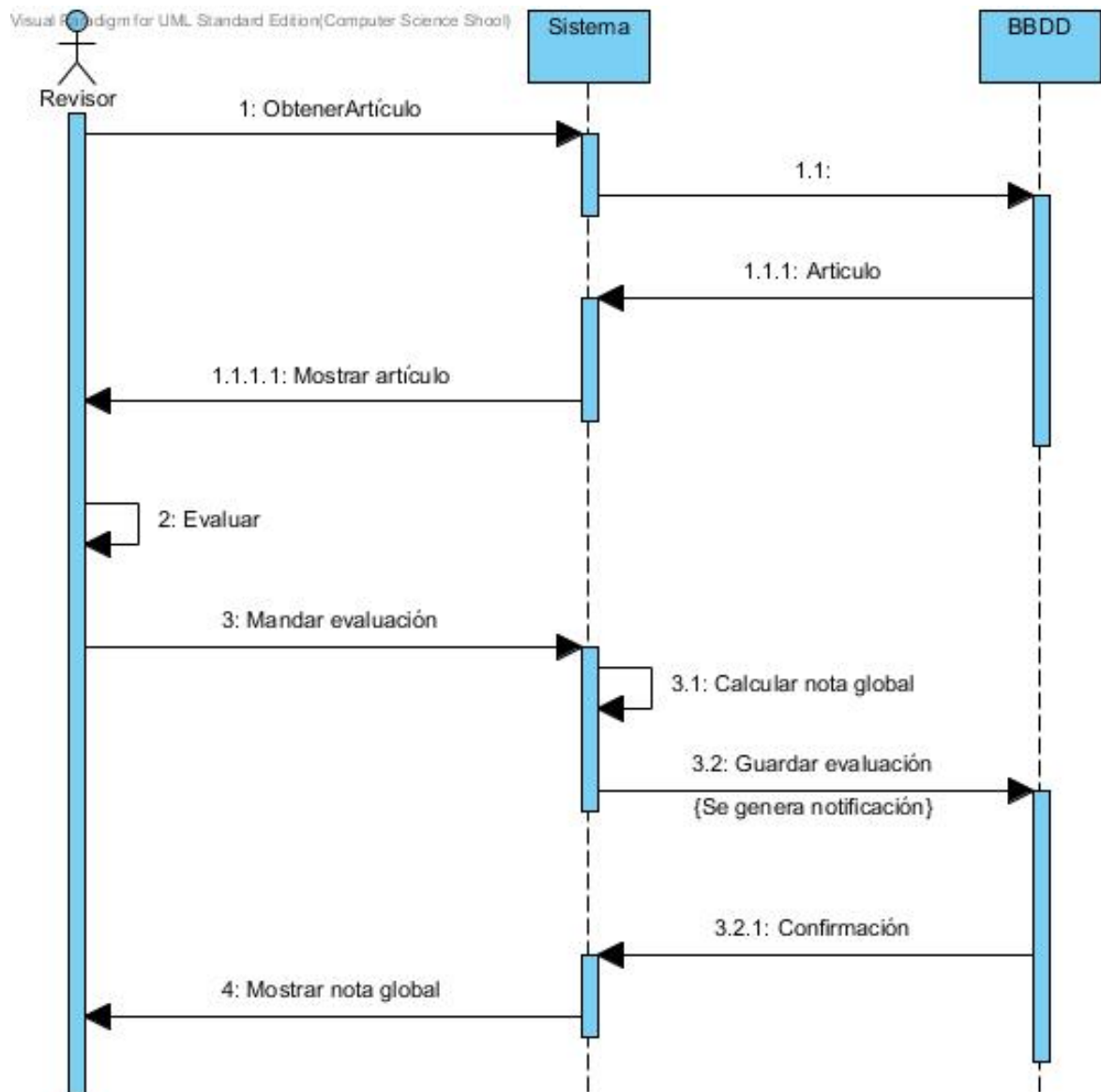


Figura 6.15: Diagrama de Secuencia: Realizar una Evaluación

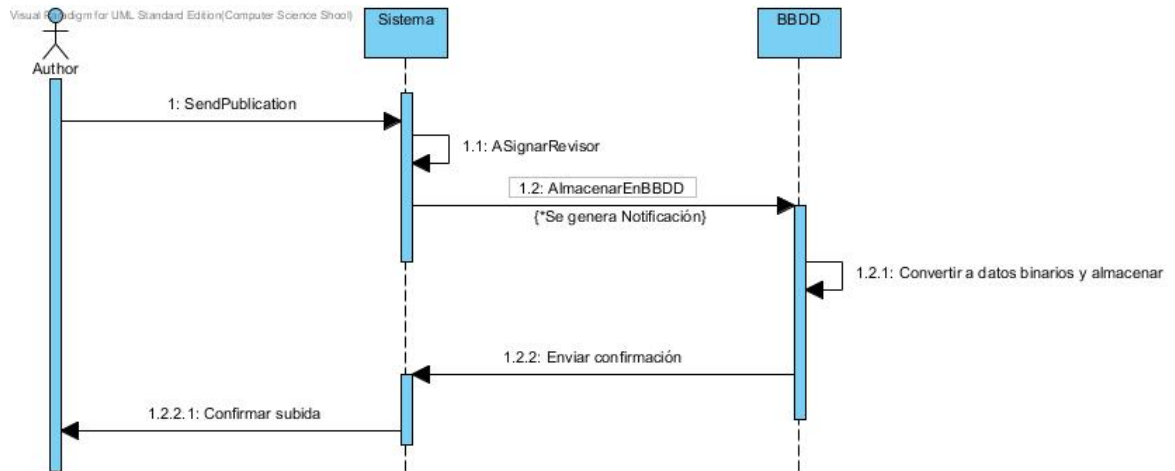


Figura 6.16: Diagrama de Secuencia: Enviar publicación

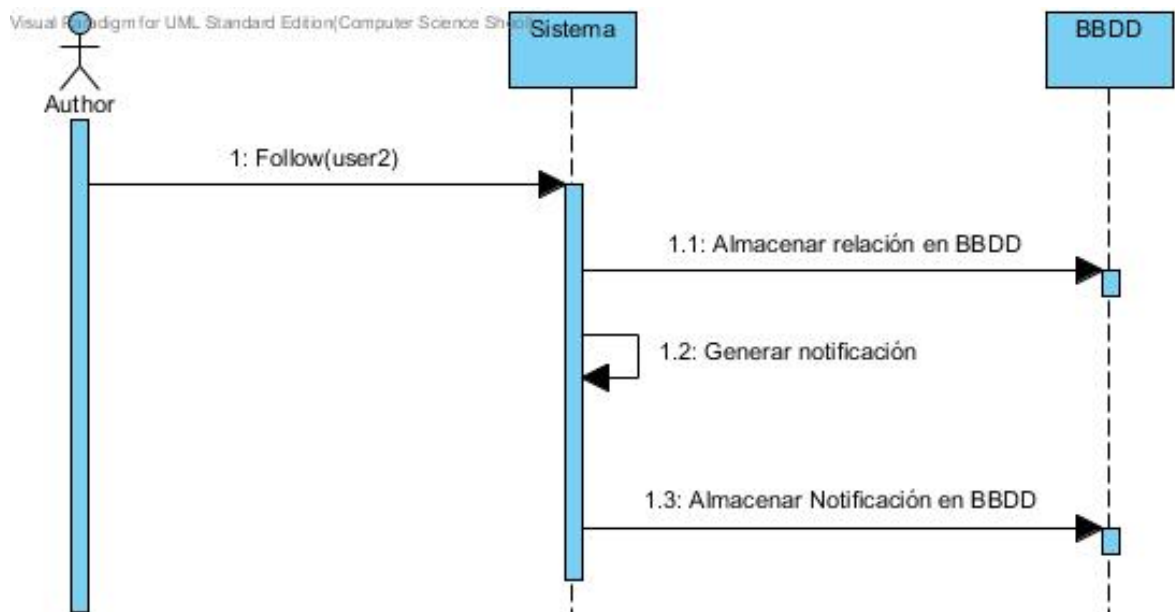


Figura 6.17: Diagrama de Secuencia: Notificar seguidor (I)

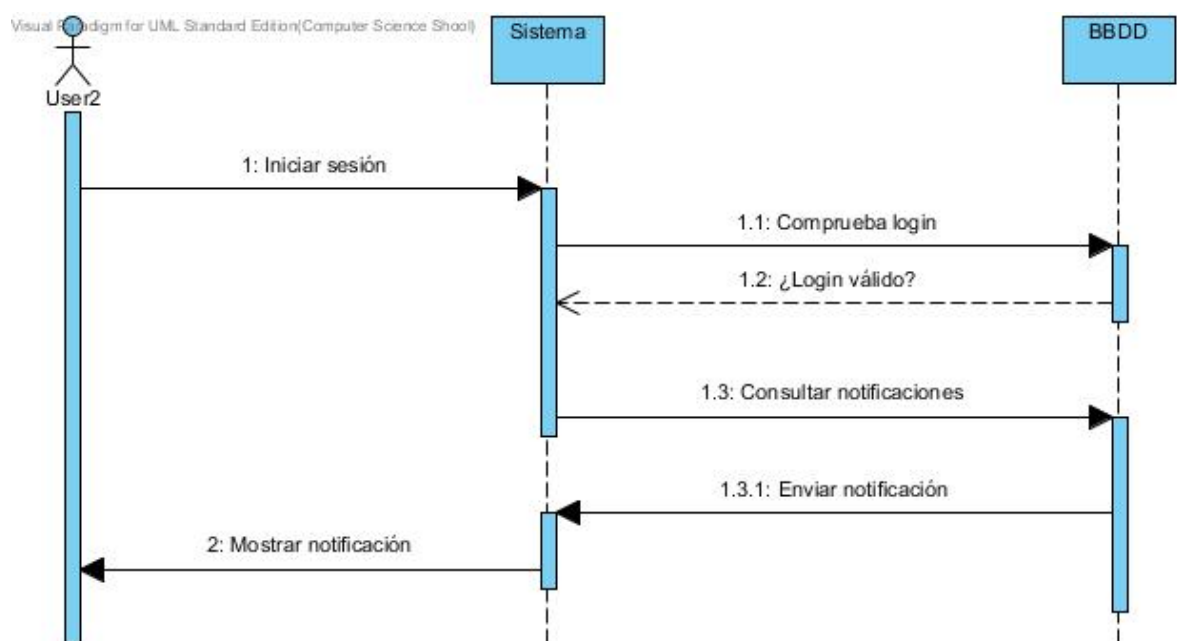


Figura 6.18: Diagrama de Secuencia: Notificar seguidor (y II)

Capítulo 7

Pruebas

7.1. Pruebas con JUnit

JUnit es un framework para el testeo de aplicaciones Java. Permite ejecutar clases Java para evaluar si el funcionamiento de esa clase se comporta de manera esperada. Del mismo modo también puede usarse para comprobar que después de una reestructuración de código o alguna modificación de una aplicación la funcionalidad no se ha visto alterada y los requerimientos del software no han variado.

La estructura de todas las clases del paquete `test_package` es:

- Método etiquetado con `@BeforeClass`: Se ejecutará antes que ninguno de esa clase, sirve para preparar el entorno de ejecución de esa clase. En nuestro caso imprimiremos un aviso como que la clase va a empezar a ejecutarse.
- Método etiquetado con `@AfterClass`: Se ejecutará el último después de todos los métodos de esa clase, normalmente se usa para revertir todas las modificaciones hechas por los test de la clase. En nuestro caso lo usamos para mostrar un aviso de que la clase ya ha terminado de ejecutarse.
- Método etiquetado con `@Before`: se ejecutará antes de cada uno de los test. Método etiquetado con `@After`: se ejecutará después de cada uno los tes.
- Métodos etiquetados con `@Test`: contiene el test propiamente dicho y es donde realizaremos las operaciones que queremos verificar.

De este modo el orden de ejecución de los métodos de una clase etiquetados de esa manera sería:

`@BeforeClass`

`@Before`

`@Test`

`@After`

`@Before`

`@Test`

@After

...

@Before

@Test

@After

@AfterClass

La primera clase `TestLogin.java` es la mas extensa puesto que verifica una funcionalidad tan comprometida como puede ser el login seguro de los usuarios. Hay un test para logear con contraseña correcta, incorrecta, mayusculas, minusculas incluso sin usar contraseña. Los test dejan claro cuales son los errores que subyacen a un simple login.

El siguiente test que se ejecuta comprueba si realmente se cambia en password y coincide con en nuevo. En la clase `TestAuthor.java` comprobamos que el que autor cuando envía un artículo realmente se almacena como suyo y que si cancela un artículo el estado de ese articulo ha sido modificado y que el estado de las revisiones también ha sido cambiado.

En la clase `TestSistema` probamos que aunque cada articulo ha de ser revisado por dos revisores puede darse el caso de que solo exista uno y el sistema tiene que asignarlo igualmente. Para ejecutar todos los test simplemente hay que ejecutar la clase `FullTest.java` que lanza todos los test. Los casos de prueba se han centrado en intentar proteger las partes mas cruciales del sistema. Por un lado los datos. Hemos considerado como parte fundamental he irremplazable del sistema los artículos escritos por los autores y dejando en segundo lugar los comentarios, revisiones, notas, listas de amigos, etc. No podemos arriesgarnos a que el sistema guarde un artículo y realmente no sea así, puesto que sera una perdida importante para el usuario. Tambien el acceso a esos datos comprobando las posibles fallas en el login de los usuarios.

Por otro lado la lógica más básica del funcionamiento de la pagina web: la asignacion automática de artículos a revisores para comentarlos. El objetivo fundamental de la página es poner en contacto a los autores con los editores a través de los revisores por lo que ningún artículo puede quedar sin comentar, puesto que los editores relegan ese trabajo en los revisores y si ese trabajo no se realiza los artículos nunca llegarán a los editores.

Las partes mas importantes para la controlar la calidad de la primera versión alpha de nuestro software es el almacenamiento seguro de los artículos y el funcionamiento mínimo de la asignación de revisiones.

Esta primera parte se centra en automatizar las pruebas críticas sobre nuestra aplicación. En la segunda parte hemos incluido otras pruebas manuales que representan los casos de uso. Su objetivo no es comprobar que el sistema funciona correctamente (que indirectamente lo hace) si no ser capaz de ver hasta qué punto el usuario final puede hacer uso de la aplicación.

User	To	Message	Received?	Notified?	Notes
clartKent .com	@autor loisLane .com	Do you love me?	Yes	Yes	Bad from/to for- matting
janeDoe .com	@autor johnSmith @revi- sor .com	Meeting at 09:00	Yes	Yes	
peterSmith @edi- tor .com	johnDoe @revisor .com	Where are my reviews?	Yes	Yes	Spanish and En- glish words are mixed

Cuadro 7.1: Ejemplo de informe de pruebas

Para ello hacemos uso de la aplicación de forma normal y apuntamos en formularios como el que se muestra los resultados de las pruebas.

Estos formularios se convertirán en informes de pruebas para comprobar el correcto funcionamiento de la aplicación cuando es utilizada de manera habitual por un usuario delante de un monitor, lo cual también es muy útil para descubrir errores visualización y comprobar sistema cumple todos los requisitos, tanto desde el punto de vista del sistema como del usuario final.

Conclusiones

8.1. Conclusiones generales

Tras haber batallado contra toda la tecnología, hemos descubierto un modo de trabajo altamente eficiente, comprensible y organizado. No poco esfuerzo hemos invertido en la elaboración de todo el proyecto, pero sí tenemos la sensación, contrariamente a otras veces, que ha sido esfuerzo bien aprovechado, que ha dado frutos y que las largas horas delante del monitor no se traducían en pocos o ningún avance, sino que pronto se podían ver resultados, lo que alienta el trabajo.

Cuando abordamos el proyecto mirábamos con reticencia los requisitos que habíamos modelado. En ocasiones pensábamos que nuestras expectativas habían ido quizá algo más allá de lo que se esperaba, pues una red social mínimamente funcional (y digna de llamarse red social) requiere de un conjunto grande de operaciones y servicios que deban implementarse, pero con Struts2 implementar dichas acciones era sencillo si previamente se había definido bien la capa de dominio.

Lidiar con la conjunción de toda la arquitectura partiendo prácticamente desde cero pudo ser lo más tedioso: elaborar el esqueleto de la aplicación, pudiendo tener operativa la base de datos, el proyecto web dinámico, las pruebas y el repositorio fue quizá lo más ingrato al ser noveles en esta materia, pero aprendimos que una vez estaba todo listo, replicarlo para otros fines era sencillo y trabajar sobre todo ello, inquietantemente cómodo.

Por otro lado, la familiaridad de Java y el modo en que se pueden utilizar los objetos, la flexibilidad para el despliegue de sus aplicaciones y en definitiva, todo el colectivo de sus ventajas, no deja de ser un factor importante a la hora de decidir sobre la arquitectura de un sistema. Tanto la propia implementación del dominio, el despliegue con el versátil Tomcat y la facilidad para incluir un entorno de pruebas tal como JUnit hacen de este conjunto de tecnologías una opción considerable a la hora de abordar un proyecto software.

En el particular, las elecciones tomadas así como el seguimiento del Proceso Unificado, han hecho que *todos hagamos de todo*, lo que se presenta como un requisito imprescindible a la hora de tratar con un personal reducido un proyecto considerable. Como conclusiones podemos sacar que como marco de trabajo, Struts ofrece una serie de ventajas considerables

que facilitan enormemente la elaboración de un producto software basado en web y que induce a un trabajo orientado al orden y la estructuración de sus contenidos, acoplándose perfectamente a cualquier metodología.

8.2. Trabajo futuro

Existen muchas tareas que se pueden realizar con el fin de mejorar la aplicación aquí desarrollada. Como se ha visto, únicamente se han tratado los requisitos más básicos, pero otorgándoles una alta funcionalidad y resistencia al error.

En un hipotético despliegue público, sería vital tener implementado el sistema de cobros (vía Paypal o similar), sobre todo en lo que refiere a toda la temática *VIP*.

Actualizar los perfiles de una manera más vistosa sería un retoque igualmente necesario para mantener al público en activo.

Quizá añadir nuevas funcionalidades más propias de redes sociales de ámbito más comercial y en definitiva, acoplar esta aplicación, hoy en una fase muy temprana de madurez, a un marco de negocio actualizado.

8.3. Notas sobre la segunda entrega

Una vez visto el temario del segundo cuatrimestre, nos hemos dado cuenta de la cantidad de cosas que se podrían corregir y mejorar en este aspecto.

Utilizar ORM sería primordial para acometer este proyecto así como escoger quizá otros sistemas gestores de bases de datos.

Trabajar con un framework de redes sociales desde luego facilitaría mucho las cosas así como mejoraría y potenciaría los aspectos relacionados con este área.

En suma, enlazando con lo anterior, se puede ver que en este mundo todo son continuos cambios y aprendizaje. Estamos satisfechos con la aplicación desarrollada pues haberla creado partiendo desde cero ha resultado gratificante, no obstante comprendemos que hay formas mejores y más rápidas de llevarlo a cabo.

ANEXOS

Anexo A

Definición de la empresa

A.1. Componente organizacional

A.1.1. Organigrama

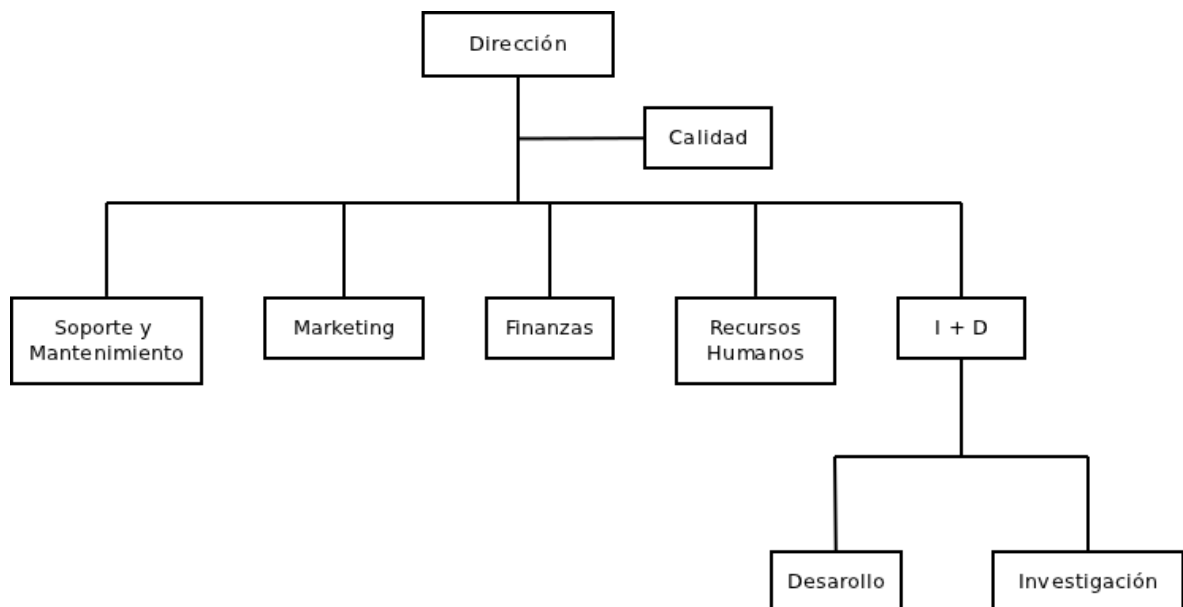


Figura A.1: Organigrama de la empresa

Distinguiéndose los siguientes cargos:

- **Director:** Persona encargada de la organización general y control de trabajo de los diferentes departamentos de la empresa. Raúl Reguillo
- **Recursos Humanos:** Se trata del departamento encargado de que en la empresa se encuentre en cada momento no sólo con la gente necesaria en cuantía sino también de forma cualitativa. Su trabajo también es conseguir que el trabajador se sienta gratificado, valorado y seguro en su puesto de trabajo. Raúl Reguillo
- **Marketing:** Es el área que hace que nuestra empresa tenga una imagen externa. Se encarga de las campañas publicitarias y el soporte a la publicidad de empresas externas como medio de financiación de la nuestra. Javier García

- **Finanzas:** Se ocupa de los estudios de adquisiciones de material, rentabilidad, inversiones, liquidez, reinversiones, de estar al día de las fluctuaciones en el mercado, de conocer las tendencias económicas y su implicaciones. Una función básica es la de planeación, pronóstico, cálculo para mantener el equilibrio económico de la empresa. Cristian del Cerro
- **I+D:** Es la sección empresarial que destina sus fuerzas al desarrollo de los servicios o productos que ofrece la empresa. El departamento de investigación es la cabeza pensante de nuestra empresa, y trata de mejorar nuestro producto estudiando nuevas tecnologías y analizando los requerimientos de los usuarios. Raúl Reguillo, Laura del Río, Javier García y Cristian del Cerro
- **Calidad:** Este departamento se encarga de que los desarrollos funcionen de forma correcta y continua. La comunicación con el usuario es continua, de tal forma que se atiende a las necesidades y sugerencias del usuario en todo momento. Javier García y Laura del Río

A.1.2. Acerca de la empresa Buggin' & Debuggin'

Descripción

B&D es una pyme multidisciplinar dedicada a ofrecer soluciones software en cualquier ámbito relacionado con las TIs. El objetivo es desarrollar y mantener servicios y aplicaciones de carácter general, ofrecer asistencia personalizada, cobertura y mantenimiento.

Target

Nos dirigimos a:

- PyMEs que requieren de aplicaciones concretas y/o servicio técnico
- EMpresas que requieren aplicaciones para una cadena de productos (como pueda ser el caso de restaurantes, hoteleras, oficinas, etc.) y/o servicio técnico
- Particulares que requieran de servicio técnico
- Desarrollo de aplicaciones para nuevas tecnologías como smartphones, tablets, etc

Horarios de trabajo

- De lunes a jueves:
 - Entrada: desde las 7 a las 9 de la mañana
 - Núcleo central de trabajo: desde las 9 a las 14 y de las 16 a las 18 (7 horas)
 - Salida: desde las 18 a las 20 de la noche
- Viernes:
 - Salida de 14 a 16

A.1.3. Plan de inversión

Gastos recurrentes

Todos los precios están en Euros salvo que se indique lo contrario:

Ver Anexos I, II y III

Costes sociales de los trabajadores

Salario bruto de 1,500 euros al mes con las pagas extras prorrateadas, y una base de cotización de 1,450 euros al mes. El coste por seguridad social para la empresa asciende a 483,33 euros mensuales, y el coste del trabajador sube a 1,983 euros mensuales.

Subtotal

- Costes sociales: 1983
- Alquiler de dominios: 12
- Apps for Business Google: 16
- Subcontratas:
 - Línea de Internet + teléfono: 20
 - Gastos variables de luz, agua y gas: 150 aproximadamente
 - Servicio de prensa: 30
- Alquiler de la oficina: 650
- Otros: 300
- **Total:** 3.161

Gastos no recurrentes

Todos los precios están en Euros salvo que se indique lo contrario:

Ver Anexos I, II y III

Subtotal

- Hardware:
 - 4 Equipos Dell latitude: $4 * 871,00 = 3484,00$
 - Servidor Dell Poweredge T110 II: 1623,00
 - Monitor Panorámico Dell E2011H: 111,20
 - Pizarra interactiva: 846,00
 - Video Proyector: 317,23
 - Multifunción BROTHER: 109,90

- Software y licencias:
 - Licencia Visual Paradigm: 699,00 (U.S. Dollars)
- Material de oficina: 402,56
- Decoración y mobiliario: 5267,68
- Otros: 692,53
- **Total:** 13.392,33

A.1.4. Estrategia de marketing y publicidad

- Presencia en las principales redes sociales: twitter, facebook
- Página web de empresa

Anexo B

Informes de Microsoft Project

Se exponen a continuación algunos fragmentos de los informes de MS Project generados durante la planificación del proyecto.

Dates			
Start:	Thu 25/10/12	Finish:	Sun 03/02/13
Baseline Start:	NA	Baseline Finish:	NA
Actual Start:	Thu 25/10/12	Actual Finish:	NA
Start Variance:	0 days	Finish Variance:	0 days
Duration			
Scheduled:	69,88 days	Remaining:	0 days
Baseline:	0 days	Actual:	69,88 days
Variance:	69,88 days	Percent Complete:	99%
Work			
Scheduled:	206,45 hours	Remaining:	57,98 hours
Baseline:	0 hours	Actual:	148,47 hours
Variance:	206,45 hours	Percent Complete:	72%
Costs			
Scheduled:	5.427,60 €	Remaining:	1.543,50 €
Baseline:	0,00 €	Actual:	3.884,10 €
Variance:	5.427,60 €		
Task Status		Resource Status	
Tasks not yet started:	0	Work Resources:	9
Tasks in progress:	8	Overallocated Work Resources	0
Tasks completed:	42	Material Resources:	0
Total Tasks:	50	Total Resources:	9

Figura B.1: Informe MS Project: General




ID	Indicators	Task Mode	Nombre de tarea	Duration	Start	Finish	Predecessors
18		Auto Scheduled	Iteraciones PUD	69,88 days	Thu 25/10/12	Fri 01/02/13	
13		Auto Scheduled	Hini - Hito Componente organizacional	0 mins	Tue 30/10/12	Tue 30/10/12	1
19		Auto Scheduled	Especificación de cada iteración	10,88 days	Fri 02/11/12	Fri 16/11/12	
50		Auto Scheduled	Hfin - Hito Entrega Proyecto/ prototipo	30 mins	Sun 03/02/13	Sun 03/02/13	13;40

Figura B.2: Informe MS Project: Hitos

	Total
Creación de la Empresa "Bugging and Debugging"	
Hini - Hito Componente organizacional	
Instalación del sistema	1.470,00 €
Iteraciones PUD	
Especificación de cada iteración	
Iteraciones creación de empresa	
Iteraciones de Intalación de sistema	
Iteraciones PUD 1	
Iteraciones PUD 2	
Iteraciones PUD 3	
Iteraciones PUD 4	
Iteraciones PUD 5	
Iteraciones PUD 6	
Iteraciones PUD 7	
Hfin - Hito Entrega Proyecto/ prototipo	52,50 €
Total	5.932,50 €

Figura B.3: Informe MS Project: Flujo de caja

Raul Reguillo			53,23 hours			
ID	Task Name	Units	Work	Delay	Start	Finish
12	Creación de documentos acreditativos	100%	2 hours	0 days	Mon 29/10/12	Mon 29/10/12
25	Identificación de Clases asociadas a un Caso de Uso	100%	5 hours	0 days	Tue 06/11/12	Wed 07/11/12
30	Identificación de Asociaciones y Agregaciones	100%	5 hours	0 days	Fri 09/11/12	Mon 12/11/12
34	Diseño e Identificación para la realización de las clases	100%	4,9 hours	0 days	Mon 12/11/12	Tue 13/11/12
14	Instalación del sistema	100%	14 hours	0 days	Wed 31/10/12	Thu 01/11/12
35	Generación del código de los componentes y procedimientos	100%	6 hours	0 days	Thu 15/11/12	Fri 16/11/12
39	Ejecución de las pruebas de integración	100%	2 hours	0 days	Fri 16/11/12	Fri 16/11/12
50	Hfin - Hito Entrega Proyecto/ prototipo	100%	0,5 hours	0 days	Sun 03/02/13	Sun 03/02/13
13	Hini - Hito Componente organizacional	100%	0 hours	0 days	Tue 30/10/12	Tue 30/10/12
26	Descripción de la Interacción de Objetos	100%	3 hours	0 days	Wed 07/11/12	Wed 07/11/12
21	Obtención y revisión de requisitos	100%	0,38 hours	0 days	Fri 02/11/12	Fri 02/11/12
22	Especificación de Casos de Uso	100%	4,68 hours	0 days	Fri 02/11/12	Mon 05/11/12
32	Especificación de interfaz	100%	0 hours	0 days	Mon 12/11/12	Mon 12/11/12
2	Acerca de la empresa	100%	5,77 hours	0 days	Thu 25/10/12	Thu 25/10/12

Figura B.4: Informe MS Project: Reparto de tareas (I)

Javier García			58,47 hours			
ID	Task Name	Units	Work	Delay	Start	Finish
7	Gastos recurrentes	100%	6 hours	0 days	Fri 26/10/12	Fri 26/10/12
25	Identificación de Clases asociadas a un Caso de Uso	100%	5 hours	0 days	Tue 06/11/12	Wed 07/11/12
26	Descripción de la Interacción de Objetos	100%	3 hours	0 days	Wed 07/11/12	Wed 07/11/12
29	Identificación de responsabilidades y atributos	100%	6 hours	0 days	Thu 08/11/12	Fri 09/11/12
32	Especificación de interfaz	100%	5 hours	0 days	Mon 12/11/12	Mon 12/11/12
34	Diseño e Identificación para la realización de las clases	100%	4,9 hours	0 days	Mon 12/11/12	Tue 13/11/12
14	Instalación del sistema	100%	14 hours	0 days	Wed 31/10/12	Thu 01/11/12
35	Generación del código de los componentes y procedimientos	100%	6 hours	0 days	Thu 15/11/12	Fri 16/11/12
38	Ejecución de las pruebas unitarias	100%	3 hours	0 days	Fri 16/11/12	Fri 16/11/12
50	Hfin - Hito Entrega Proyecto/ prototipo	100%	0,5 hours	0 days	Sun 03/02/13	Sun 03/02/13
13	Hini - Hito Componente organizacional	100%	0 hours	0 days	Tue 30/10/12	Tue 30/10/12
12	Creación de documentos acreditativos	100%	0 hours	0 days	Mon 29/10/12	Mon 29/10/12
21	Obtención y revisión de requisitos	100%	0,38 hours	0 days	Fri 02/11/12	Fri 02/11/12
22	Especificación de Casos de Uso	100%	4,68 hours	0 days	Fri 02/11/12	Mon 05/11/12

Figura B.5: Informe MS Project: Reparto de tareas (II)

Cristian Del Cerro			46,18 hours			
ID	Task Name	Units	Work	Delay	Start	Finish
25	Identificación de Clases asociadas a un Caso de Uso	100%	5 hours	0 days	Tue 06/11/12	Wed 07/11/12
26	Descripción de la Interacción de Objetos	100%	3 hours	0 days	Wed 07/11/12	Wed 07/11/12
29	Identificación de responsabilidades y atributos	100%	4 hours	0 days	Thu 08/11/12	Thu 08/11/12
34	Diseño e Identificación para la realización de las clases	100%	4,9 hours	0 days	Mon 12/11/12	Tue 13/11/12
14	Instalación del sistema	100%	14 hours	0 days	Wed 31/10/12	Thu 01/11/12
35	Generación del código de los componentes y procedimientos	100%	6 hours	0 days	Thu 15/11/12	Fri 16/11/12
21	Obtención y revisión de requisitos	100%	0,38 hours	0 days	Fri 02/11/12	Fri 02/11/12
22	Especificación de Casos de Uso	100%	4,68 hours	0 days	Fri 02/11/12	Mon 05/11/12
50	Hfin - Hito Entrega Proyecto/ prototipo	100%	0,5 hours	0 days	Sun 03/02/13	Sun 03/02/13
13	Hini - Hito Componente organizacional	100%	0 hours	0 days	Tue 30/10/12	Tue 30/10/12
38	Ejecución de las pruebas unitarias	100%	3 hours	0 days	Fri 16/11/12	Fri 16/11/12
9	Definición de estrategia	100%	0,72 hours	0 days	Mon 29/10/12	Mon 29/10/12

Page 1



Who Does What as of Sun 19/05/13
Proyecto1.mpp

Indicators Nombre del recurso Work

Cerro" continued

ID	Task Name	Units	Work	Delay	Start	Finish
12	Creación de documentos acreditativos	100%	0 hours	0 days	Mon 29/10/12	Mon 29/10/12
32	Especificación de interfaz	100%	0 hours	0 days	Mon 12/11/12	Mon 12/11/12

Figura B.6: Informe MS Project: Reparto de tareas (III)

Laura Del Rio			48,58 hours			
ID	Task Name	Units	Work	Delay	Start	Finish
8	Gastos no recurrentes	100%	1,37 hours	0 days	Fri 26/10/12	Fri 26/10/12
25	Identificación de Clases asociadas a un Caso de Uso	100%	5 hours	0 days	Tue 06/11/12	Wed 07/11/12
26	Descripción de la Interacción de Objetos	100%	3 hours	0 days	Wed 07/11/12	Wed 07/11/12
30	Identificación de Asociaciones y Agregaciones	100%	3,75 hours	0 days	Fri 09/11/12	Mon 12/11/12
32	Especificación de interfaz	100%	3 hours	0 days	Mon 12/11/12	Mon 12/11/12
34	Diseño e Identificación para la realización de las clases	100%	4,9 hours	0 days	Mon 12/11/12	Tue 13/11/12
14	Instalación del sistema	100%	14 hours	0 days	Wed 31/10/12	Thu 01/11/12
35	Generación del código de los componentes y procedimientos	100%	6 hours	0 days	Thu 15/11/12	Fri 16/11/12
39	Ejecución de las pruebas de integración	100%	2 hours	0 days	Fri 16/11/12	Fri 16/11/12
50	Hfin - Hito Entrega Proyecto/ prototipo	100%	0,5 hours	0 days	Sun 03/02/13	Sun 03/02/13
13	Hini - Hito Componente organizacional	100%	0 hours	0 days	Tue 30/10/12	Tue 30/10/12
12	Creación de documentos acreditativos	100%	0 hours	0 days	Mon 29/10/12	Mon 29/10/12
21	Obtención y revisión de requisitos	100%	0,38 hours	0 days	Fri 02/11/12	Fri 02/11/12
22	Especificación de Casos de Uso	100%	4,68 hours	0 days	Fri 02/11/12	Mon 05/11/12
	MySQL			0 hours		
	Eclipse			0 hours		
	Apache Tomcat			0 hours		
	Herramienta Dia			0 hours		
	Latex			0 hours		

Figura B.7: Informe MS Project: Reparto de tareas (y IV)

Anexo C

Informes de pruebas

Informe de Pruebas de SocialNetwork

Descripción: Write a message

Fecha: 24-03-2013

Versión: 0.1

From	To	Message	Received?	Notified?	Notes
Author: clarkKent@autor.com	Author: loisLane@autor.com	Test message 1	YES	YES	
Author: clarkKent@autor.com	Reviewer: johnDoe@revisor.com	Test message 2	YES	YES	Bad notification formatting
Author: clarkKent@autor.com	Publisher: janeDoe@editor.com	Test message 3	YES	YES	
Reviewer: johnDoe@revisor.com	Author: loisLane@autor.com	Dummy message 1	YES	YES	
Reviewer: johnDoe@revisor.com	Reviewer: peter@revisor.com	Dummy message 2	YES	NO	loisLane@autor.com did not received any notification
Reviewer: johnDoe@revisor.com	Publisher: janeDoe@editor.com	Dummy message 3	YES	YES	
Publisher: janeDoe@editor.com	Author: loisLane@autor.com	#%&' "¿ç \$ñ@	YES	YES	Some letters are missed (ñ)
Publisher: janeDoe@editor.com	Reviewer: johnDoe@revisor.com	¿?! <>{}()""\	YES	YES	Bad notification formatting
Publisher: janeDoe@editor.com	Publisher: anne@editor.com	Last message	YES	YES	

Figura C.1: Informe de pruebas (I)

Informe de Pruebas de SocialNetwork

Descripción: Register and modify data (autor)

Fecha: 24-03-2013 **Versión:** 0.1

	Registered Data	Profile	Modified	Profile
E-mail	peterPetreli@autor.com	peterPetreli@autor.com		peterPetreli@autor.com
Name	Peter	Peter	Pedrito	Pedrito
Second name	Petreli	Petreli		Petreli
City	New York	New York	Las Pedroñeras	Las Pedroñeras
Date of Birth	1985-02-21	1985-02-21		1985-02-21
Password	heroes	heroes		heroes

Figura C.2: Informe de pruebas (II)

Bibliografía

- [Foua] Apache Foundation. *Apache Struts 2*. <http://struts.apache.org/development/2.x/>.
- [Foub] Apache Foundation. *Apache Tomcat*. <http://tomcat.apache.org>.
- [Fouc] Eclipse Foundation. *Eclipse*. <http://www.eclipse.org>.
- [GHJV96] E. Gamma, R. Helm, R. Johnson, y J. Vlissides. *Design Patterns*. Addison-Wesley, 1996.
- [Mica] Sun Microsystems. *Java*. <http://www.java.com>.
- [Micb] Sun Microsystems. *MySQL*. <http://www.mysql.com>.
- [Ora] Oracle. *Oracle Virtual Box*. <http://www.virtualbox.org>.
- [Wik] Wikipedia. *Wikipedia*. http://es.wikipedia.org/wiki/Red_social.

Este documento fue editado y tipografiado con \LaTeX
empleando la clase **arco-pfc** que se puede encontrar en:
https://bitbucket.org/arco_group/arco-pfc

[Respetar esta atribución al autor]