Proyecto fin de carrera. Facultad de informática.
Universidad de Las Palmas de Gran Canaria

Desarrollo de una red social destinada a la pequeña y mediana empresa

Jonay Santana García
Las Palmas de Gran Canaria, 20/05/2010

Proyecto fin de carrera de la Facultad de Informática de la Universidad de Las Palmas de Gran Canaria presentado por el alumno:

Jonay Santana García

Título del Proyecto: Desarrollo de una red social destinada a la pequeña y mediana empresa

Tutor: Agustín J. Sánchez Medina

Alexis Quesada Arencibia

DEDICATORIA

A mis Padres y hermanos.

AGRADECIMIENTOS

Me gustaría tener unas palabras de agradecimiento a todos aquellos que han ayudado a que éste proyecto se llevara a cabo.

En primer lugar a los tutores por su dirección y colaboración en el mismo.

Gracias ante todo a mi familia, sin las cual nada de esto hubiera sido posible.

Por supuesto mil gracias a todos mis amigos, en especial a mis compañeros en el IUCTC Javier Domínguez Montes y Roberto González Suárez por aguantarme durante la realización del proyecto y por su ayuda.

Con cariño y admiración gracias.

ÍNDICE

Introducción.

PyMesNET. ¿Una red social para PyMes?

Para empezar extraeremos de diferentes fuentes la definición que toma una Red Social.

Si extraemos la definición de Wikipedia una red social es "una estructura social que se puede representar en forma de uno o varios grafos en el cual los nodos representan individuos (a veces denominados actores) y las aristas relaciones entre ellos. Las relaciones pueden ser de
distinto tipo, como intercambios financieros, amistad, relaciones sexuales, o rutas aéreas. También es el medio de interacción de distintas personas como por ejemplo juegos on líne, chats, foros, spaces, etc."

Esta sería una definición más generalizada, pero si nos centramos en el punto de vista de Internet una red social "trata de portales en donde las personas se inscriben y alimentan el sistema con sus datos personales, fotografías, videos, gustos específicos, etcétera. La idea, en el fondo, es hacer una especie de conglomerado de amistades, algunas que ya conocemos en la vida real, así como otras que son nuevas y que, incluso, no son de nuestro país."

Y el término Pymes acrónimo de pequeñas y medianas empresas, viene a significar aquellas empresas con características distintivas, y tienen dimensiones con ciertos límites ocupacionales y financieros prefijados por los Estados o Regiones. Son agentes con lógicas, culturas, intereses y un espíritu emprendedor específicos. Así, este tipo de empresas, mayoritario en nuestra región y también a nivel nacional, es el que más empleo aporta.

Uno se puede preguntar, ¿qué diferencias tiene una red social hecha por un estudiante a las que circulan por la red?

Primero que nuestra red social esta orientada a las Pymes y la segunda, y para nosotros la más importante, es que se ha implementado en Ruby on Rails.

La explicación del porque está destinada a las Pymes, es porque uno de los factores donde se sustenta el éxito de las empresas se encuentra en las redes, tanto formales como informales que puedan establecer. En este sentido, las pequeñas empresas se encuentran en una posición de desventaja frente a las grandes corporaciones. Por ello, el primer objetivo de este proyecto es diseñar una herramienta que permita mejorar las redes que poseen las Pymes. Por otra parte, este tipo de herramienta también permitirá los consumidores finales puedan conocer las diferentes Pymes asociadas a la red y que estas tengan la oportunidad de promocionarse por esta vía.

Por todo ello, se requiere desarrollar una Red Social que permita a las Pymes aprovechar la oportunidad de incrementar su negocio captando más clientes mediante el uso de dicha Red Social y fomentar el contacto con otras empresas de forma que puedan compartir, experiencias, resolución de problemas, inquietudes, etc. De este modo, entre otras cuestiones, la red que se pretende crear permitirá mejorar la competitividad de las empresas a través de la colaboración con proveedores, clientes y otras empresas.

La siguiente pregunta que nos hemos de realizar es, ¿qué funcionalidades nos permitirá hacer la aplicación?

Para comenzar registrarnos y crearnos un perfil de nuestra empresa. A partir de aquí, subir fotos, comentarios, blogs, grupos, conocer a otras empresas así como clientes y otras muchas acciones que se irán explicando al largo de esta documentación. Nosotros, con la ayuda de los tutores, realizaremos una aplicación, en la que podamos llegar a decir que forma parte de la generación Web 2.0.

Esperamos que al finalizar esta aplicación estar satisfecho con el resultado final y haber aprendido el lenguaje Ruby on Rails, adaptar este lenguaje con Ajax y otro tipo de tecnologías y/o lenguajes y por último, y no menos importante, tratar con el cliente, entendiendo sus puntos de vista pero a la vez mostrarle los nuestros.

Estado actual del tema.

Redes Sociales

Una red social es una estructura social que se puede representar a través de uno o varios grafos, en la cual los nodos representan individuos u otras redes, denominados *actores*, y las aristas, relaciones entre ellos. Las relaciones pueden ser de distinto tipo, pero están principalmente destinadas a la realización de operaciones de intercambio (Casson, 2007).

Las redes permiten el desarrollo de un capital social (Adler & Kwon, 2002; Burt, 2000), debido a que hacen posible la interacción repetida mediante encuentros de carácter formal o informal. Esta reiteración del contacto consolida lazos de confianza y estrecha las relaciones de cooperación que permiten a un emprendedor obtener numerosos beneficios, al exponerse a diferentes ideas, descubrir nuevas oportunidades, recibir consejo, identificar, localizar y recolectar recursos y acceder a información valiosa.

Muchos autores han sugerido que las redes sociales son de particular valor para los emprendedores, principalmente debido a que les permiten acceso a recursos (Premaratne, 2001), proveen información relevante (Bygrave y Minniti, 2000), son una fuente de competitividad (Malecki y Veldhoen, 1993), favorecen el crecimiento y desarrollo de los emprendimientos (Johannisson, 2000; Hansen, 2000), permiten la entrada a mercados internacionales (Phelan, Dalgic, Li y Sethi, 2006), son una fuente de legitimidad (Elfring y Hulsink, 2003) y han sido reconocidas como espacios para la innovación y el reconocimiento de oportunidades (Singh, Hills, Hybels y Lumpkin, 1999).

Las redes sociales se distinguen de las físicas, en el sentido de que las conexiones entre estas últimas se dan entre objetos, tales como edificios, equipos, sistemas de información, unidos a través de canales que permiten el flujo de intercambios. Dado lo anterior, es evidente la importancia económica de las redes físicas, puesto que ellas incluyen infraestructura de transporte (vías aéreas, terrestres y marítimas), infraestructura de comunicaciones (telecomunicaciones e Internet) o sistemas de distribución Casson (2007).

Casson (2007) menciona que si bien las redes sociales y las físicas son diferentes, están relacionadas de manera cercana. Por ejemplo, una red para exportación de los productos de un emprendimiento a mercados internacionales es una red física, pero las redes de comunicación entre los comerciantes de un país y otro son sociales. Las redes físicas consisten en el flujo de bienes y servicios, mientras que las redes sociales consisten, entre otros aspectos, en el flujo de necesidades, acuerdos de precios y pactos de colaboración (Phelan, Dalgic, Li y Sethi, 2006). Así, una red física compromete inversión en capital financiero, representada, por ejemplo, en la infraestructura usada para el transporte o en la compra de una carta de crédito; una red social representa la existencia de capital social definido por relaciones de compromiso y confianza entre los actores. La inversión en capital financiero supone el flujo de bienes tangibles; por su parte, el capital social supone el flujo de bienes intangibles (Casson, 2007).

Las redes sociales también varían en duración, dependiendo del tipo de relaciones de intercambio o transacciones que realicen sus actores. Existen redes sociales que son solamente transitorias, en las que grupos o uniones temporales se relacionan para aprovechar oportunidades específicas de mercado y luego los actores se dispersan. Ese tipo de redes pueden generar una diversidad de contactos que un emprendedor puede cruzar posteriormente, pero su importancia tiende a ser menor que la de las redes que persisten en el tiempo (Casson, 2007). Las redes sociales de largo plazo se autorrenuevan, retirando antiguos actores y seleccionando e incorporando nuevos miembros entre los cuales se generan altos niveles de confianza y compromiso (Granovetter, 1985). Las redes de largo plazo proporcionan cierto tipo de estabilidad, a través de la creación de fuertes lazos de relación, ayudando a reducir la incertidumbre asociada con la actividad emprendedora.

Hung (2006) resalta la existencia de dos tipos de redes importantes en el desarrollo de los nuevos emprendimientos: las redes interpersonales y las redes interorganizacionales. Según él, las redes interpersonales se refieren a redes personales del emprendedor, mientras que las interorganizacionales se refieren a redes extendidas. Una red interpersonal denota cómo a nivel personal los emprendedores están embebidos en varios sistemas sociales. Una red interorganizacional muestra cómo un *new entrepreneurial venture* ya establecido, se relaciona con otras organizaciones, después de que ha sido

creado, formado y desarrollado por su fundador. Así, los emprendedores usarán sus redes interpersonales para relacionarse con otros individuos o grupos, buscando acceso a recursos y fuentes de información relevante, en su propósito de convertir una idea en una entidad de negocio. Las relaciones interpersonales del emprendedor están constituidas básicamente por los sistemas sociales en los cuales está inmerso, tales como su familia, amigos, colegas, sus contactos de negocios y las afiliaciones a clubes o asociaciones profesionales.

De otro lado, una red interorganizacional es un mecanismo estratégico para mejorar la ventaja competitiva de una empresa a través de la minimización de los costos de transacción, mientras se mantiene la flexibilidad que permite acelerar la innovación tecnológica. Hansen (1995) sugiere que estructuralmente las redes interorganizacionales corresponden a redes de colaboración de tipo informal, y que son consideradas como una forma alternativa de coordinación interorganizacional frente a los mercados y jerarquías. Cuando las relaciones son de largo plazo, las transacciones de carácter reiterativo y existe algún tipo de especificidad de activos (Williamson, 1991), se genera interdependencia, la cual puede ser también una explicación de la formación de redes de cooperación interorganizacional, debido a que en este caso cumplen con tres funciones específicas: regular las transacciones inter-firmas, desarrollar una visión de futuro compartida y proveer una infraestructura de coordinación (Hung, 2006).

Casson (2007) menciona que aunque el crecimiento de un emprendimiento es logrado a través del fortalecimiento de las redes interorganizacionales, el aspecto interpersonal de las redes sociales es siempre fundamental, debido a que las redes interorganizacionales son mantenidas en la práctica por las comunicaciones interpersonales entre los representantes de las organizaciones; un cambio en las personas puede minar las relaciones entre las empresas, si los sucesores que se hacen cargo de ellas no son apropiadamente inducidos acerca del manejo de estas relaciones. De esta forma, muchas relaciones interorganizacionales no son más que formalizaciones impersonales de antiguas relaciones personales, las cuales tienden a disolverse tan pronto como la gente clave se retira, muere o es promovida.

Otra tipología de redes corresponde a las categorías *formal* e *informal*. Las redes formales son redes estructuradas a través de organizaciones establecidas para tal fin. Estas organizaciones están conformadas por actores de diversos tipos y tienen como objetivo la realización de actividades estructuradas para promover el incremento de sus miembros, generar interacción entre los emprendedores y proveer servicios, capital de inversión y contactos con proveedores de recursos. Las redes de emprendimiento conformadas por estas organizaciones son estructuras formales, debido a que existen exclusivamente para el propósito de facilitar la conectividad y el intercambio de información entre sus miembros y son parte de una organización formalmente conformada (Malewicki, 2005). Sin embargo, no cabe duda de que en el desarrollo de las actividades formales de estas redes, surgen contactos de carácter informal, producto de las interacciones personales entre los miembros, que ocurren por fuera de las actividades y esfuerzos realizados en la red. Hite y Hesterly (2001) indican que las organizaciones de redes de emprendimiento se convierten en un vehículo para acelerar el desarrollo de redes personales y pueden actuar como un puente entre redes basadas en la identidad y redes basadas en la racionalidad económica. Las redes basadas en la identidad están caracterizadas por fuertes lazos personales de afinidad, que pueden ser más útiles para los emprendedores en sus etapas de inicio. Las redes basadas en la racionalidad económica, buscan primordialmente el lucro y son importantes una vez conformado el emprendimiento y en sus etapas tempranas de crecimiento.

Impacto de las Redes Sociales 2.0 en las organizaciones

Introducción

Con el advenimiento de las tecnologías Web 2.0 la nueva generación de usuarios de Internet está reescribiendo las reglas de interacción social, y la forma en que el negocio es llevado a cabo. Por la utilización de medios electrónicos y las herramientas Web 2.0 como los blogs Wiki, los nuevos e ingeniosos métodos de la interacción social atraviesan las fronteras geográficas y los silos de la industria que están siendo creados (Fu et al., 2007; IBM, 2007).

En tan sólo cinco años, esta innovadora aplicación social electrónica se ha introducido en el dominio del negocio. Muchas razones han sido citadas por la popularidad de la electrónica de redes sociales entre los trabajadores de oficina, las más notables son: la disponibilidad de los ordenadores portátiles de bajo coste de acceso a internet, trabajar desde casa, y la erosión creciente de los conceptos tradicionales de las horas de oficina (Shirky, 2008; andWilliams Tapscott, 2006). Durante 2007 ClearSwift encargó una investigación para determinar el grado en que los medios de comunicación social están siendo utilizados (Clearswift, 2007a). Ellos encontraron que:

- El 83 por ciento de los trabajadores de oficinas en EE.UU., utiliza recursos de la oficina para acceder a los medios de comunicación social;
- El 30 por ciento de los trabajadores de oficinas en los EE.UU. y el 42 por ciento de los oficinistas del Reino Unido discutieron cuestiones relacionadas con el trabajo a través de aplicaciones de medios sociales;
- El 40,8 por ciento de las TI y los tomadores de decisiones en los negocios creían que los medios de comunicación social son relevantes en el entorno corporativo de hoy,
- Sólo el 11,1 por ciento de las TI y los líderes empresariales ya estaban haciendo uso de medios de comunicación sociales en sus negocios.

El objetivo de este estudio es identificar los beneficios y riesgos asociados de redes sociales en las organizaciones, que permitirá a los líderes de organización y de las TI comprender el alcance e impacto de las redes sociales. Y de esta forma fundamentar el porqué la implementación de este proyecto.

Objetivo y metodología del estudio

La incorporación de las tecnologías Web 2.0 como las redes sociales han sido acreditadas con la capacidad de ampliar los contactos sociales, acelerar los procesos de negocio, la mejora de relaciones con los clientes, la contratación costo-efectiva de personal de alto calibre, así como la mejora de la moral, la motivación y satisfacción laboral entre el personal. En el lado negativo de esta forma de redes sociales se ha

ganado la reputación de forma negativa al efectuar la productividad del personal, y de muchas empresas por temor a daños en la productividad y reputación (de MessageLabs, 2007a).

Este estudio tiene como objetivo conocer el impacto de las redes sociales 2.0 en las organizaciones. Con el fin de lograr este objetivo llevamos a cabo un estudio de los recursos disponibles, que incluyen los académicos como la literatura, artículos de revistas, libros blancos, los medios de comunicación populares y libros.

En la siguiente sección, la literatura actual disponible se utilizará para crear una definición de las redes sociales 2.0. Esta definición se utilizará para crear un conjunto de criterios con el propósito de determinar si una aplicación de red social cumple con la definición. Posteriormente veremos las ventajas evidentes de la utilización de las redes sociales electrónicas, siguiendo con un enfoque por las razones en contra de la aplicación de redes electrónicas sociales y una identificación de los riesgos claves.

Definición de la próxima generación Web basada en redes sociales electrónicas

La interacción humana y la colaboración por lo general se llevan a cabo dentro de los grupos. Estos grupos se forman alrededor de una relación compartida, meta o proyecto. Los grupos requieren la capacidad de interactuar con otros grupos para compartir su conocimiento y habilidad para que el grupo tenga éxito y sea innovador. El advenimiento de las redes informáticas y el Internet ha hecho posible que la interacción del grupo que tendrá lugar con independencia de la ubicación geográfica o la zona horaria, y la posterior incorporación de las tecnologías Web 2.0 ha hecho de esta interacción y cooperación más fluida, rentable y de fácil mantenimiento. (Anria Sophia van Zyl, 2008).

El término utilizado para describir lo nuevo a partir del uso de Internet es Web 2.0. Aunque las partes interesadas no han podido llegar a un acuerdo sobre la definición de Web 2.0, se podría definir como la segunda generación que percibe plataformas basadas en Web. Estas plataformas a cabo consisten en aplicaciones específicamente diseñadas para ayudar a la colaboración en línea y compartir contenidos generados por los usuarios (Clearswift, 2007a; Matuszak, 2007; O'Reilly, 2005).

En la Tabla I se resumen las tecnologías más populares que son adecuadas al desarrollo de redes sociales.

Tecnología	Descripción
Blogging (web blog)	Los blogs son una herramienta de auto-publicación que se asemeja a las revistas en línea donde periódicamente el propietario postea mensajes. Los lectores pueden suscribirse a un blog, un enlace a la misma, compartir enlaces, comentarios, en un formato interactivo e indicar su relación social con otros bloggers que leen el blog particular.
Wikis	Un wiki es un sitio web que permite la colaboración online, permitiendo a varios usuarios agregar, quitar o modificar el contenido y el contenido de cambio. También permite la vinculación entre cualquier número de páginas.
Social bookmarking (marcadores sociales)	Los libros sociales de marca permite a los usuarios publicar sus listas de favoritos o sitios web favoritos para que otros usuarios busquen y vean.
Tagging (etiquetado)	El etiquetado es el uso de palabras clave para rastrear el contenido en sitios web. Se puede utilizar como una forma de social de bookmarking, donde el usuario puede tener acceso a todos los contenidos determinados por otros usuarios y vinculados a la palabra clave específica.

Really simple syndication (RSS)	Un formato de fuentes web utilizada para publicar contenido actualizado con frecuencia. Lo que permite a los usuarios suscribirse a sus favoritos, "alimenta" la recepción automática de actualizaciones.
Collaborative real time editor	Una aplicación que permite la edición simultánea de un texto o archivo de medios por los diferentes participantes en una red.

Tabla 1. Tecnologías Web 2.0 Fuentes: ClearSwift (2007a, b); Godwin-Jones (2006); Matuszak (2007)

Aunque el cuadro no puede ser visto como una lista exhaustiva de todas las tecnologías Web 2.0 actualmente disponibles, enfatiza la naturaleza de las tecnologías con un enfoque en la colaboración en línea y compartiendo todo el contenido generado por el usuario. Gracias a que la naturaleza de las tecnologías Web 2.0 es fácilmente adoptada por los usuarios como una herramienta de ayuda para la creación de redes sociales en el mundo virtual.

Hay muchos términos utilizados por el público y los académicos para describir esta nueva ola de experiencias Web y la creación de redes sociales. Estos son algunos de los términos que se incluyen, pero no están limitados: Redes sociales, Web 2.0, comunidades virtuales, e- comunidades, comunidades en línea, software de redes sociales, software de colaboración y Servicios de Red Social (Boyd y Ellison, 2007; Rosen, 2007; Shirky et al., 2008). Cuando estas nuevas tecnologías y aplicaciones encuentran su camino en el dominio de los negocios, se refieren a menudo como Enterprise 2.0, EnterpriseWeb 2.0 o Enterprise Social Software (Matuszak, 2007; McAfee, 2006a, 2006b).

El problema con estos términos es que no están bien definidos y por lo tanto pueden significar diferentes cosas en diferentes contextos y por diferentes usuarios (Boyd y Ellison, 2007; Clearswift, 2007b). Por ejemplo las redes sociales electrónicas pueden incorporar las tecnologías Web 2.0 o pueden ser vistas como una forma de tecnología Web 2.0. La diferencia es que las primeras permiten a sus usuarios interactuar con otros

usuarios o cambiar contenido del sitio web, en contraste a las segudas donde los usuarios se limitan a la visualización pasiva de información que se les proporciona. (Clearswift, 2007b; Matuszak, 2007; O'Reilly, 2005).

Esta sindicación de las redes sociales electrónicas y las tecnologías Web 2.0 se conoce como red social 2.0, con el fin de diferenciarla de las redes sociales más tradicionales y las formas anteriores basadas en la web de redes sociales electrónicas (Clearswift, 2007b; Matuszak, 2007; O'Reilly, 2005.

Las aplicaciones de las Redes sociales 2.0 deben crear y administrar una expresión digital de las relaciones personales de la gente o los enlaces, al ofrecer actualizaciones automáticas del libro de direcciones y los perfiles que desee ver. Estas aplicaciones también deben ayudar en la identificación y la conversión de los vínculos potenciales en vínculos débiles o fuertes, proporcionando "introducción en los servicios "y permitir a los usuarios mostrar sus conocimientos y experiencia en un formato de búsqueda (Boyd, 2006; Clearswift, 2007a).

Los componentes que deben estar presentes para dar cumplimiento a estos criterios pueden resumirse en los siguientes:

- La aplicación debe construir una expresión digital de las relaciones personales y enlaces (Boyd, 2006);
- Se debe ayudar al descubrimiento de los lazos potenciales (Granovetter, 1973), y
- Ayudar a la conversión de los vínculos potenciales en vínculos débiles y fuertes (Granovetter, 1983).

Las redes sociales 2.0 por lo tanto se pueden definir como aplicaciones o sitios web que sostienen el mantenimiento de las relaciones personales, el descubrimiento del potencial de las relaciones y deben ayudar en la conversión de los vínculos potenciales en vínculos débiles y fuertes, mediante la utilización de las tecnologías Web 2.0 emergentes (Anria Sophia van Zyl, 2008).

Para que un individuo determine si desea crear una conexión con otra persona, necesitará algún tipo de retroalimentación social. La retroalimentación social es esencial

en la formación de una reputación digital (también conocida como karma o Whuffie) y permite que los usuarios valoren las contribuciones de los demás (Boyd, 2006; Brown y Duguid, 2000). La reputación digital ayuda a los usuarios para determinar si una persona posee más conocimiento, experiencia y pericia que dice tener, y si la creación de un vínculo débil o fuerte con esa persona sería ventajosa.

Los métodos tradicionales de comunicación empleados en Internet utilizan canales de comunicación donde la información se comunica de arriba hacia abajo o en una dirección. El énfasis de las aplicaciones y webs de redes sociales 2.0 se encuentra con conversaciones de dos vías donde todos los participantes tienen la oportunidad de participar y compartir opiniones y conocimientos (MessageLabs, 2007a).

Para calificar como redes sociales 2.0, dos o más de las siguientes modalidades de comunicación mediada por ordenador deben ser utilizadas según Boyd y Ellison (2007):

1. uno a uno (por ejemplo, correo electrónico o mensajería instantánea para uso privado y comunicaciones confidenciales);
2. de uno a muchos o uno-a-pocos (por ejemplo, páginas web y blogs), y
3. muchos-a-muchos o pocos a pocos (por ejemplo wikis y whiteboards).

La definición y componentes de redes sociales 2.0 se pueden resumir en el Tabla 2.

Criterios	Componentes
Creación de redes de apoyo social (debe contener los tres componentes)	Construir una expresión digital de las relaciones personales y vínculos Ayuda en el descubrimiento de los vínculos potenciales Ayuda en la conversión de los vínculos potenciales en vínculos débiles o fuertes
Apoyo a dos o más modos de comunicación mediada por ordenador (debe contener al menos dos	Uno-a-uno Uno-a muchos/ uno-a-pocos Muchos-a-muchos/pocos-a-pocos

componentes)	
Permitir realimentación social	Las contribuciones de un usuario son valoradas por los demás usuarios

Tabla 2. Requisitos Redes sociales 2.0 (Clearswift)

Beneficios asociados con las redes sociales 2.0

Según John Brown y Paul Duguid el conocimiento puede ser definido por tres criterios, a saber: conocimiento está asociado a un conocedor, el conocimiento está incrustado en el conocedor, y llegar a ser un conocedor una persona que debe tener el compromiso de entender la información presentada a él (Brown y Duguid, 2000, pp. 119-120). En las organizaciones este conocimiento está conformado por la experiencia, conocimientos especializados y el conocimiento práctico de cómo operar los sistemas de organización (Orlikowski, 2002).

Las Redes sociales 2.0 proporcionan a los usuarios la posibilidad de crear una lista global de los datos de contacto (ya sea en un formato gráfico o basada en texto) de las personas con las que tienen fuertes lazos profesionales, compañeros de trabajo, colegas y personas que hacen negocios, quiénes tienen la confianza necesaria para participar e incluso de recomendar a los demás (Gorge, 2007). Esta lista de contactos es diferente de otros directorios electrónicos en los que la información está vinculada directamente a los perfiles creados y mantenidos por el mismo contacto, lo que permite actualizaciones automáticas de los cambios en los datos de contacto, las actividades en curso, interés y conocimientos especializados y experiencia, en un formato de búsqueda (Boyd, 2006; Clearswift, 2007a).

Estas expresiones gráficas de las relaciones personales que pueden ser adquiridas a lo largo del transcurso de toda una carrera, permiten a los usuarios identificar las relaciones mutuas que pueden ser explotadas para introducciones o recomendaciones (Boyd, 2006; Gorge, 2007; Granovetter, 2004).

Una función importante del sistema del servicio social es la prestación de una colaboración aprendiendo del ambiente, en el que los problemas encontrados son

colectivamente resueltos y las soluciones son compartidas entre pares, reduciendo la brecha entre los procedimientos y la práctica (Boshoff y du Plessis, 2008; Brown y Duguid, 2000; Cairncross, 2001, p. 132; Davenport, de 2001, Orlikowski, 2002).

Este flujo natural de los conocimientos se ve gravemente alterado en las empresas distribuidas geográficamente, expandiéndose a través de diversas líneas de servicios, departamentos, regiones geográficas y zonas de tiempo (Brown y Duguid, 2000, p. 78). Don Tapscott y Anthony Williams señalaron que el conocimiento es cada vez más visto como un producto de las personas en red y organizaciones que están buscando nuevas soluciones a problemas específicos (Tapscott y Williams, 2006, p. 153).

En las organizaciones jerárquicas, donde los trabajadores del conocimiento se agrupan en especialistas en líneas de servicio o de procedimientos, los vínculos débiles se vuelven más importantes, con el fin de ser capaces de acceder a conocimientos especializados y presentar la información de otras organizaciones de redes sociales (líneas de servicios) (Granovetter, 1973, 1983, 2004). Los recursos organizativos a menudo se pierden cuando los empleados tienen que reinventar soluciones a los problemas, que ya han sido creados por otra persona dentro de la organización (Brown y Duguid, 2000, p. 112, IBM, 2007). En un sistema de gestión del conocimiento perfecto, todo conocimiento no es rival y sólo debería ser producido una vez. Los recursos adicionales efectuados deben aumentar su valor y la precisión para eliminar errores y deficiencias encontradas en el pasado (Benkler, 2006, pp. 36, 37, 373).

El conocimiento y la información suelen extenderse a lo largo de muchos tipos de herramientas de comunicación, formatos de documentos, aplicaciones de escritorio, y las fuentes dentro y fuera del cortafuegos, y pueden incluir el correo electrónico, faxes, mensajes instantáneos, manuales, hojas de cálculo y presentaciones. La integración de los diferentes modos de comunicaciones mediadas por ordenador en una sola aplicación permite a los trabajadores del conocimiento global información de forma eficiente, permitiendo a los usuarios añadir etiquetas (a través de enlaces, etiquetas y los marcadores sociales), para hacer material más persistente que facilita la consulta y la participación (Brown y Duguid, 2000, p. 200; Cairncross, 2001, p. 132; IBM, 2007).

Productividad y flujo de trabajo son a menudo obstaculizados por el uso del correo electrónico, mensajes instantáneos y llamadas telefónicas. La comunicación síncrona o en tiempo real (por ejemplo, llamadas telefónicas y reuniones) puede llevar mucho tiempo, que interrumpen y producen una disminución en la productividad, mientras que las comunicaciones asíncronas o retardadas (como el correo electrónico) a menudo se usan mal y en exceso (Burger y Rensleigh, 2007; Richtel, 2008).

Las Redes sociales 2.0 pueden ayudar a las organizaciones para crear un recurso en línea que contiene la sabiduría acumulada de la organización, permitiendo al conocimiento ser codificado, buscado y compartido (Cairncross, 2001, pp. 131, 134, IBM, 2007). Al disminuir el uso de correos electrónicos y otros métodos de comunicación destructivos, el uso de métodos de comunicación asíncrona, como los blogs y wikis, puede aumentar la productividad y la eficiencia del flujo de trabajo.

Otros ejemplos incluyen:

- Etiquetado y marcado de libros sociales permiten colegas para la búsqueda y localización de expertos y "mirar por encima del hombro" en la industria de artículos, blogs, manuales, wiki y otra información que el experto considera útil, y descubrir así respuestas y soluciones, sin interrumpir con correo electrónico, mensajes instantáneos o llamadas telefónicas (Godwin-Jones, 2006; IBM, 2007).

- Permitir a los usuarios participar en las discusiones, la planificación y toma de decisiones, cuando tienen el tiempo para hacerlo, en un foro abierto, sin la necesidad de enviar y enviar correos electrónicos a todos los participantes (Ariyur, 2008).

- Permitir a los usuarios tener siempre acceso a la última versión de un documento y contribuir a la comprensión de este fenómeno mediante la adición de anotaciones y enlaces a fuentes externas (Godwin-Jones, 2006).

La lista siguiente muestra ejemplos de lo que constituiría el uso eficaz y apropiado de algunas de las herramientas de comunicación mediada por ordenador incluido en las Redes sociales 2.0:

(1) Herramientas de comunicación mediada por ordenador: uno-a-uno (ejemplo: E-mail):
- Tiempo crítico de comunicaciones (Andreson et al., 2006);
- Las comunicaciones privadas y personales (Andreson et al., 2006);
- La información confidencial o sensible (Andreson et al., 2006).

(2) Herramientas de comunicación mediada por ordenador: uno-a-muchos (por ejemplo, blogs):
- "Recomendar" ideas a un público amplio y compartir el conocimiento en un formato narrativo (Brown y Duguid, 2000, p. 106; IBM, 2007);
- Comunicaciones tradicionales, tales como boletines de noticias (Clearswift, 2007b);
- Foros informales para tratar cuestiones con el personal, clientes y socios, respondiendo a las preguntas (Clearswift, 2007b; Godwin-Jones, 2006).

(3) Herramientas de comunicación mediada por ordenador: Muchos-a-muchos (por ejemplo: wikis):
- Preguntas y respuestas (Matuszak, 2007);
- La planificación colaborativa, toma de decisiones conjuntas (Ariyur, 2008);
- Conocimiento de captura y clasificación (Clearswift, 2007b).

Mantener la moral del personal y la satisfacción laboral, mientras que mantener la disciplina y la productividad se ha convertido en uno de los mayores retos de los gerentes. Los defensores de la Redes sociales 2.0 y herramientas de colaboración sostienen que estas plataformas abiertas pueden tomar la fricción de la colaboración (Tapscott y Williams, 2006, pp. 94-6), crear una cultura de compartir (IBM, 2007) y aumentar la satisfacción en el trabajo y con ello aumentar la productividad.

Peter Kollock argumentó que hay cuatro motivaciones de la gente para contribuir conocimiento, experiencia y el tiempo sin la expectativa de recibir un beneficio directo (de monitoreo o de otro tipo) a cambio (Smith y Kollock, 1999, pp. 227-9). Estos hallazgos se pueden resumir de la siguiente manera: Una persona puede estar motivada para contribuir valiosa información al grupo, por esperar para recibir ayuda e información útil a cambio (Graham y Hall, 2004; Smith y Kollock, 1999, p. 227). Esto puede llevar a una cultura de compartir conocimientos y experiencia (IBM, 2007).

Las contribuciones de las Redes sociales 2.0 a través de puntuaciones, comentarios, y la creación de un siguiente (personas que enlazan con, o suscribirse a su trabajo). Esta reputación digital sirve para reconocer las contribuciones de una persona y más allá al grupo inmediato, y pone un valor al conocimiento de la persona y las habilidades de creación de conocimiento (Brown y Duguid, 2000, p. 112; IBM, 2007; Smith y Kollock, 1999, p. 228).

Esta mayor visibilidad satisface el deseo de la mayoría de los individuos por el prestigio y el reconocimiento y aumenta su satisfacción en el trabajo (IBM, 2007; Smith y Kollock, 1999, p. 228). Las personas pueden estar motivadas a participar en grupos debido a un deseo de tener un efecto en su medio ambiente por hacer las cosas bien (Shirky, 2008, pp. 131-133; Smith y Kollock de 1999, p. 228). Clay Shirky (2008) señaló que mucha gente se siente motivada a contribuir a las malas contribuciones, que por el deseo de iniciar una nuevo artículo a partir de cero. Los individuos también pueden estar motivados a compartir una innovación con la esperanza de que la comunidad va a mejorar, por lo que las innovaciones serían más útiles a sí mismos. Esto se ve muy a menudo en el movimiento de código abierto (Benkler, 2006, p. 42, Smith y Kollock, 1999, p. 228).

Estos procesos transparentes (donde todas las contribuciones son vistas y respondidas por la comunidad) puede ayudar a las comunidades en la co-creación de soluciones donde las no "Buy-in" son necesarias, porque los equipos están emocionalmente comprometidos con una solución o plan acordado (Ariyur, 2008).

Una de las áreas donde las redes sociales 2.0 tendrán el mayor impacto en las organizaciones es en la comunicación continua con los clientes y el público, apoyadas por las redes sociales 2.0. Esta comunicación abierta puede tener un impacto en la imagen de las organizaciones percibidas o marca, y su imagen de ser innovadoras y líderes del mercado (Anria Sophia van Zyl, 2008).

Las relaciones con los clientes se han mejorado permitiendo a los clientes el acceso directo a la información, para los que anteriormente habrían tenido un teléfono, o e-mail. Esto elimina la frustración causada por los retrasos (Brown y Duguid, 2000, p. 77; Cairncross, 2001, p. 132; Clearswift, 2007b). Se estima que tres cuartas partes de las redes sociales de Reino Unido han visitado ya perfiles creados por las empresas, en sitios como MySpace y Facebook, para promover determinadas marcas (IBM, 2007; MessageLabs, 2007a).

Las Redes sociales 2.0 también se pueden utilizar como una herramienta de marketing viral, donde la gente está animada a transmitir mensajes de marketing en forma voluntaria a través de la palabra (IBM, 2007). Las promociones virales pueden incluir secuencias de vídeo, juegos en Flash, e-books, software libre, imágenes y mensajes de texto.

La innovación puede ser estimulada mediante una supervisión de las comunicaciones del cliente, retroalimentación y opiniones (Matuszak, 2007; Tapscott y Williams, 2006, pp. 93-94). Esta comunicación continua con los clientes se puede utilizar para el desarrollo de soluciones por la utilización de opiniones de los clientes en la toma de decisiones clave de un producto (IBM, 2007).

Los efectos negativos y los riesgos asociados con las redes sociales 2.0

Muchas organizaciones ya utilizan alguna forma de directorio electrónico que contiene la información de contacto del personal, clientes, proveedores y otros agentes, y podría ser sostenido en otro directorio que no es necesario (Cairncross, 2001, p. 133). Estas listas pueden mantenerse en la aplicación de contactos Microsoft Outlook o en aplicaciones e-mail similares, o mantenerse como una hoja de cálculo de un miembro de personal responsable y tiene que ser actualizada permanentemente cuando los contactos cambian

de oficina, cambian de números de teléfono y direcciones de correo electrónico; y un cierto grado de descomposición del enlace (cuando la información del contacto no está al día) puede tener lugar (Brown y Duguid, 2000, p. 201).

Los nuevos servicios de directorio abierto utilizados por las redes sociales 2.0 permiten a las personas acceder a un gran volumen de información, que puede ser utilizado en un ataque de ingeniería social (KasperskyLab, 2008; Leitch y Warren, 2006). Los spammers y escritores de virus pueden configurar perfiles falsos y de rastreo a través de redes sociales (incluyendo blogs) reunir información sobre cargos, números de teléfono, e-mail direcciones, etc (MessageLabs, 2007a).

Perfiles falsos, blogs y otras herramientas de creación de redes, pueden contener enlaces a otros sitios web que descargan el software espía no deseado, o el desplazamiento en sí mismo puede contener un archivo flash con un virus o un gusano incrustado (Clearswift, 2007d; MessageLabs, 2007a). El objetivo de la mayoría del malware es causar una fuga de datos.

Una de las mayores preocupaciones con respecto a las plataformas de redes sociales, es que la productividad se efectuará negativamente ya que los empleados pueden dedicar demasiado tiempo a la creación de redes y a la publicación de entradas en los blogs y las wikis. También existe el riesgo de que los empleados lo utilizen más con fines sociales y no sobre el trabajo de anuncios relacionados (Ariyur, 2008; Clearswift, 2007b; MessageLabs, 2007a; Shirky, 2008, pp. 120-1). Esto puede tener graves implicaciones en cuanto a la capacidad y la utilización de servidores y redes, con ancho de banda que se ha congestionado con contenidos multimedia a menudo no relacionados con el trabajo (Clearswift, 2007d; MessageLabs, 2007a).

El conocimiento de las aplicaciones de las redes sociales 2.0 no se crea en el control de grupos jerárquicos. La información generada por el usuario usando herramientas de colaboración, tales como los blogs y las wikis, permiten a cualquiera añadir y modificar contenidos, incluidos los actores imprevistos que no son expertos en la materia (Ariyur, 2008; Clearswift, 2007b). Este interlocutor produce conocimiento que no puede ser tan fiable como los procedimientos y manuales generados por personal especializado y se

comunicará a la cadena de mando. El vandalismo y la desinformación causada por los empleados pueden dejar abiertos a los trabajadores a la acción legal (en el principio de la responsabilidad civil subsidiaria), por el cual los trabajadores son responsables de la negligencia, actos u omisiones de sus empleados en el curso de su trabajo, incluso si estos actos son accidentales (Clearswift, 2007c).

La posibilidad de vincular, marcar y los marcadores sociales son algunas de las características principales de las Redes sociales 2.0, por lo que es fácil de compartir, etiquetar, y encontrar información. Muchos trabajadores están preocupados por la posible pérdida de información confidencial por un descuido (o malicioso) comentario o vínculo creado por un empleado, lo que podría traducirse en una vergüenza de empresa, perjuicio financiero, la responsabilidad legal o posibles riesgos de seguridad (Clearswift, 2007b; MessageLabs, 2007a; NETconsent Limited, 2004).

El daño a la reputación de la organización también puede ser causado por los artículos que aparecen en la prensa sobre los empleados al ser despedidos por una organización por la utilización inadecuada de los recursos de la oficina (NETconsent Limited, 2004). Al publicar comentarios personales negativos acerca de su organización, clientes y colegas en línea puede llegar a ser fácil de encontrar a través de una búsqueda en línea y pueden estar disponibles por un tiempo ilimitado (Clearswift, 2007b; MessageLabs, 2007a, 2007b). Otra preocupación seria son las herramientas sociales del foro creadas, en las que los clientes insatisfechos pueden criticar y quejarse de la organización en la creación de una imagen pública de la organización que están fuera del control de la organización (Shirky, 2008, p. 179).

Los impactos (tanto negativos como positivos) de las redes sociales 2.0 se pueden resumir en la Tabla 3.

Percepción positiva	**Percepción negativa**
Información actualizada del contacto vinculada al usuario manteniendo los perfiles	Fuente potencial de información que puede ser utilizada en los ataques de ingeniería social

Identificación de expertos, oportunidades y posibles socios comerciales	Los spammers y creadores de virus pueden crear perfiles falsos
Aumento de la productividad y eficiencia del flujo de trabajo. Aumento de la motivación del personal y sentido de comunidad a través de la acumulación de una reputación digital	Disminución de la productividad causada por excesivo tiempo y fijación de entradas en los blogs y wikis por parte de los empleados
Mantenimiento del conocimiento acumulativo organizacional y experiencia en un formato de búsqueda completa	El contenido generado por el usuario puede ser poco fiable. Posible pérdida de información confidencial o sensible
Uso más eficaz, adecuado y eficiente de las tecnologías de comunicación mediada por ordenador	Desperdicio de recursos con respecto al ancho de banda, servidor y utilización de la red
La capacidad de influir en la percepción de la organización y / o marcas a través de la mejora en las relaciones con los clientes, el marketing viral e innovación	Daño a la reputación de la organización ya sea a través de actos intencionales de vandalismo y desinformación o por un acto negligente u omisiones

Tabla 3. Impacto social del Networking 2.0 en organizaciones
(Clearswift, 2007b; MessageLabs, 2007a, 2007b).

Conclusión

El objetivo de esta documentación es identificar los beneficios y riesgos asociados de las redes sociales en las organizaciones, lo que nos permitirá comprender el alcance y el impacto social del Networking 2.0 en las organizaciones.

Las redes sociales 2.0 se pueden definir como la utilización de las tecnologías Web 2.0 por sitios web o aplicaciones de apoyo al mantenimiento de las relaciones personales, el descubrimiento de posibles relaciones y para la ayuda en la conversión de los vínculos potenciales en vínculos débiles y fuertes.

Las razones a favor y en contra de la implementación de las redes sociales electrónicas como herramienta de gestión del conocimiento, se identificaron anteriormenye. Los motivos para la aplicación de las Redes sociales 2.0 mostraron que las plataformas de redes sociales aumentan la productividad, la eficiencia del flujo de trabajo, la motivación del personal y la innovación al permitir:

- A los usuarios utilizar las tecnologías de la comunicación mediada por ordenador con mayor eficacia y apropiada para colaborar con los compañeros de trabajo;
- la identificación de los expertos, las oportunidades y posibles colaboradores fuera de los trabajadores del conocimiento tradicional de los canales de la organización;
- la retención de la acumulación del conocimiento organizacional y experiencia en un formato de búsqueda.

Algunas de las razones claves en contra de la aplicación de las redes sociales 2.0 son:

- las ventajas evidentes de la actual estructura de la organización jerárquica donde los trabajadores del conocimiento se agrupan en canales y la información es comunicada en una dirección, frente a la estrategia de plataforma abierta propugnada por las plataformas emergentes basadas en la Web;

- existe el temor a que las plataformas de redes sociales tendrán un efecto negativo en la productividad;

- la posible pérdida de datos confidenciales o sensibles mediante actos negligentes o maliciosos por los empleados o por medio de la ingeniería social o ataques malware.

Esta documentación se centró en el impacto de las redes sociales 2.0 en las organizaciones haciendo especial hincapié en los beneficios percibidos y los efectos negativos para las empresas. Sólo se puede plantear la hipótesis de si la identificación y aplicación de procedimientos de reducción del riesgo conducirá a los beneficios de

permitir la creación de redes sociales en las organizaciones para salir por la percepción negativa que los líderes de la organización tienen actualmente.

Redes sociales destinadas a Pymes

Actualmente hay dos redes sociales destinadas a las Pymes y negocios. Por un lado nos encontramos con Linked In sitio web orientado a negocios, fue fundado en diciembre de 2002 y lanzado en mayo de 2003 principalmente para redes profesionales.

Y por otro lado tenemos RedSocialPymes.com es una comunidad que incluye, involucra y conecta a cientos de miles de PyMes de todo Iberoamérica. El concepto del sitio es que, a través de él, podamos acceder a otras PyMes y por ello hacer nuevos negocios, ampliando principalmente la red de contactos.

Objetivos.

El objetivo fundamental del proyecto es desarrollar una red social para PyMes de Gran Canaria. Las tareas que se van a abordar para la realización de dicha red social son las siguientes:

a) Promover las PyMes de Gran Canaria
b) Posibilidad de creación de cuentas tanto para PyMes como usuarios anónimos.
c) Gestión y control del número de Pymes y del resto de usuarios, así como de las cuentas y uso de las mismas.
d) Encontrar oportunidades de negocios y socios potenciales
e) Publicar ofertas de trabajo y encontrar a los mejores talentos para cada PyMe.
f) Chat y Foro para hacer que la comunicación sea mejor.
g) Mantenimiento actualizado de la red social.

Otros objetivos que se quieren conseguir con la realización de este proyecto son:

- Realizar una página intuitiva, moderna, ágil, dinámica y funcional.
- Utilizar y conocer nuevos lenguajes y tecnologías. Como por ejemplo Ruby on Rails, Ajax.
- Adaptarla de una manera que forme parte de la nueva generación de aplicaciones Web 2.0
- Poder llevar a la práctica gran parte de la teoría vista en las diferentes asignaturas de la carrera, especialmente las tareas de análisis, diseño e implementación de un proyecto, que van desde que se tiene una idea inicial hasta que esa idea es una realidad, y todos los pasos y planificaciones llevados a cabo para realizarlo.

Dotar a la aplicación de una gran reusabilidad, de tal forma que se puedan implementar mejoras cambios con relativa facilidad y sin afectar al diseño original.

Requerimientos

Los principales puntos en los que se basará esta aplicación serán:

- Simple de usar.
- Intuitiva.
- Moderna.
- Completa.
- El foco principal sea la Pyme.

Que sea simple de usar e intuitiva porque irá dirigida a personas que no saben manejarse por la red, moderna quiere decir una página que no se vea desfasada, antigua, cuando nos referimos a completa, quiere decir que sea una página donde hayan perfil de la empresa, fotos, se puedan crear posts, etc. y el último punto, ya se ha comentado bastante el porqué.

Otro punto importante es el perfil de la empresa estará marcado por una serie de campos, como la descripción o información de la misma, foro, comentarios y otros campos que aún estarían por determinar. Este perfil también estará formado por fotos y videos.

Estos requerimientos se irán ampliando y modificando al largo del proyecto, por ejemplo la creación de eventos.

Motivaciones

Para hacer frente a este proyecto y no a otros, hemos tenido en cuenta a parte de otros puntos, estos dos factores:

1. Ruby en Rails.
2. Ajax.

Realizarla en Ruby on Rails nos proporcionará el aprendizaje de un nuevo lenguaje, y en este caso, este lenguaje supone la nueva generación en desarrollo de aplicaciones Web y cada día va ganando más adeptos ya que fue diseñado para un desarrollo rápido y sencillo.

El segundo punto va enlazado con el primero, ya que Ajax también pertenece a esta nueva generación de desarrollo de aplicaciones Web. Ajax no es una tecnología, es realmente muchas tecnologías como por ejemplo, XML y JavaScript.

Ajax proporcionará entre el usuario y nuestra aplicación una interacción asíncrona, independientemente de la comunicación con el servidor, de esta manera el usuario no estará mirando la página en blanco del navegador y un icono del reloj de arena esperando a que el servidor haga algo.

Tanto Ruby on Rails como Ajax forman parte de Web 2.0, que representa la evolución de las aplicaciones tradicionales hacia aplicaciones Web enfocadas al usuario final.

Metodología.

Vamos a describir la metodología a utilizar para la realización del proyecto haciendo especial énfasis en las técnicas de Ingeniería del Software que se utilizarán, así como en las técnicas para planificar y coordinar las actividades del proyecto. Así como el mantenimiento del proyecto.

Para las etapas de análisis y desarrollo se hará uso de las herramientas aprendidas en ingeniería del software. En cuanto al análisis, lo enfocaremos en un entorno orientado a objetos con UML, el cual comprende las etapas de análisis de requisitos de usuario y análisis de requisitos de software.

Para el diseño, se generarán el diseño de la base de datos y el diseño de la aplicación Web (Red Social). Mientras que en la etapa de desarrollo nos basaremos en un ciclo de vida en espiral en el que se definen cuatro actividades principales:

1. Determinación de objetivos, alternativas y restricciones,
2. Análisis de alternativas e identificación o resolución de riesgos,
3. Desarrollo del producto del siguiente nivel y
4. Planificación de la siguiente fase, cuyo ciclo repetiremos hasta que alcancemos los objetivos del proyecto.

Recursos necesarios.

Para la realización del trabajo será necesario disponer de un equipamiento informático básico: PC (Windows XP), impresora, etc.., así como tener acceso a otras herramientas básicas e imprescindibles para la consecución del proyecto: editor de texto, acceso a internet, etc....

Para la realización del software se seleccionará un lenguaje de programación en este caso RubyonRails. Usaremos software libre, en este caso usaremos Tog es una nueva plataforma para la creación de todo tipo de comunidades 2.0, desde blogs hasta completas redes sociales. Se trata de una plataforma en Ruby&Rails de código abierto, bajo licencia MIT, completamente extensible gracias a la incorporación de plugins. Además la base de datos estará implementada en SQLite3.

Además de estos recursos, hemos tenido la oportunidad de tener un puesto de trabajo en el Centro IUCTC (Instituto Universitario de Ciencias y Tecnologías Cibernéticas).

Plan de trabajo y temporización.

Desarrollo del plan de trabajo desglosado en etapas, con una estimación en cada etapa del tiempo de ejecución, basándonos en la propuesta del proyecto realizada conjuntamente con los profesores. Cada una de las actividades a desarrollar en las distintas etapas del proyecto conlleva la correspondiente tarea de generación de la documentación respecto a la tarea ejecutada.

Etapa 1: Análisis (260 horas)

Actividad 1.1: Documentación y herramientas.

- Estudio herramientas necesarias para el PFC.
- Búsqueda en internet de información herramientas
- Instalación del hardware y software necesario para el PFC

Actividad 1.2: Identificación del ámbito del sistema.

- Búsqueda de información sobre la funcionalidad del sistema y estudio del mismo
- Definición de objetivos y fines del sistema

Actividad 1.3: Identificación del ámbito del sistema.

- Análisis de requerimientos

Actividad 1.4: Prototipo de Interfaz. Actividad en la que desarrollaremos un prototipo de la interfaz.

- Desarrollo de prototipo de la interfaz

Etapa 2: Diseño (250 horas)

Actividad 2.1: Diseño Arquitectónico. Estructuraremos el sistema y modelamos el control del mismo.

- Estructuración del sistema
- Modelado de Control

- Descomposición Modular

Actividad 2.2: Modelado de la aplicación web. Definiremos el modelado de la aplicación y los diagramas de clases de la aplicación.

- Definición del modelado de la aplicación
- Definición de los diagramas de clases de la aplicación

Actividad 2.3: Modelado relacional de aplicación. Definiremos el modelado relacional de la aplicación.

- Definición del modelado relacional de la aplicación
- Generación documentación

Etapa 3: Implementación (250 horas)

Actividad 3.1: Módulo de cuadro de mando. Actividad en la que desarrollaremos la base de datos y la modulos de la misma.

- Desarrollo de la base de datos
- Desarrollo de módulo de bloques

Actividad 3.2: Interfaz Gráfica. Actividad encargada del desarrollo del estilo de la interfaz y casos de uso.

- Desarrollo del controlador
- Desarrollo del estilo de la interfaz
- Desarrollo de los casos de uso

Etapa 4: Validación y Publicidad del PFC (70 horas)

Actividad 4.1: Tests de validación. Definición y aplicación de los test para validar la red social.

- Definición de los test de validación
- Aplicación de los test de validación
- Análisis de resultados de los test de validación

Actividad 4.2: Publicidad. Publicidad del proyecto realizado, mediante la confección de un manual y promoción a través del una página web.

- Confección de manuales de usuario
- Realización página web publicidad PFC

Global de Horas: Estimamos un total de 850 horas.

Etapa 1 de Mi proyecto: Análisis

Desarrollo de la etapa 1 de mi proyecto, que tiene por nombre Análisis. En esta etapa realizaremos las siguientes actividades con su correspondiente documentación:

Actividad 1.1: Documentación y herramientas.

Actividad del proyecto en donde estudiaremos las herramientas necesarias para el desarrollo del mismo, información sobre esas herramientas y la generación de documentación de ellas. Así como la instalación del hardware y software necesario.

1. Estudio herramientas necesarias para el PFC.

Tarea donde haremos un estudio sobre las herramientas necesarias para la realización del proyecto.

1. Ruby on Rails

1.1 ¡Sobre raíles!

Ruby on Rails, también conocido como RoR o Rails es un framework de aplicaciones Web de código abierto escrito en el lenguaje de programación Ruby, siguiendo el paradigma de la arquitectura Modelo Vista Controlador (MVC). Trata de combinar la simplicidad con la posibilidad de desarrollar aplicaciones del mundo real escribiendo menos código que con otros frameworks y con un mínimo de configuración. El lenguaje de programación Ruby permite la metaprogramación, escribir programas que escriben o manipulan otros programas, de la cual Rails hace uso, lo que resulta una sintaxis que muchos usuarios encuentran muy legible. Rails se distribuye a través de RubyGems, que es el formato oficial de paquete y canal de distribución de librerías y aplicaciones Ruby.

1.2 Ruby

1.2.1 Historia

El lenguaje fue creado por Yukihiro "Matz" Matsumoto, quién empezó a trabajar el Ruby en 24 de febrero de 1993, y lo lanzó al público en el año 1995. El nombre "Ruby" fue ideado en base a la piedra correspondiente al mes de nacimiento de un amigo. A la fecha de Diciembre de 2007, el 26 de Diciembre de 2007 fue publicado Ruby 1.9.0, versión estable, en la que empezamos a desarrollar, pero que actualizamos cuando fue publicado Ruby 2.0.2, también una versión estable, que incorpora mejoras sustanciales en el rendimiento del lenguaje. Diferencias en rendimiento entre la implementación de Ruby (1.8.6) y otros lenguajes de programación más arraigados ha llevado al desarrollo de varias máquinas virtuales para Ruby. Entre esas se encuentra JRuby, un intento de llevar a la plataforma Java, y Rubinius, un intérprete modelado en base a las máquinas virtuales de Smalltalk.

1.2.2 Filosofía

El creador del lenguaje Yukihiro "Matz" Matsumoto, ha dicho que Ruby está diseñado para la productividad y la diversión del desarrollador, siguiendo los principios de un buen interface de usuario. Sostiene que el diseño de sistemas necesita enfatizar las necesidades humanas más que las de la máquina.

Ruby sigue el "principio de la menor sorpresa", lo que significa que el lenguaje debe comportarse de tal manera que minimice la confusión de los usuarios experimentados. Matz ha comentado que su principal objetivo era hacer un lenguaje que fuera capaz de minimizar el trabajo de programación y la posible confusión.

1.2.3 Sintaxis

La sintaxis de Ruby es similar a la de Perl o Pyton. La definición de clases y métodos está definida por palabras clave. Sin embargo, en Perl, las variables no llevan prefijos.

Cuando se usa, un prefijo indica el ámbito de las variables. La mayor diferencia con C y Perl es que las palabras clave son usadas para definir bloques de código sin llaves. Los saltos de línea son significativos y son interpretados como el final de una sentencia, el punto y coma tiene el mismo uso.

Ruby mantiene todas sus variables de instancia privadas dentro de las clases y sólo la expone a través de métodos de acceso (attr_writer, attr_reader, …). Los métodos de acceso en Ruby pueden ser escritos con una sola línea de código. Como la innovación de estos métodos no requiere el uso de paréntesis es trivial cambiar una variable de instancia en una función sin tocar una línea de código o refactorizar dicho código. En Ruby todas las variables de instancia son privadas y también proporciona una manera sencilla de declarar métodos set y get. Esto mantiene el principio de que en Ruby no se pueden acceder a los miembros internos de una clase desde fuera de esta, en lugar de esto se pasa un mensaje, se invoca un método, a la clase y recibe la respuesta.

1.2.4 Características

· Orientado a objetos. Esto quiere decir que todos los datos en Ruby son un objeto, en el sentido de Smalltalk, sin excepción. Por ejemplo: En Ruby, el número 1 es una instancia de la clase Fixnum.

· Existe diferencia entre mayúsculas y minúsculas *(case sensitive)*.

· **Comentarios**. Todo lo que siga al símbolo # hasta el final de la línea en que aparece, es ignorado por el intérprete. Para facilitar bloques de comentarios, el intérprete ignora también todo el texto que aparezca entre una línea que comience con =begin y una que termine con =end.

· **Límites de expresiones** Múltiples expresiones en una misma línea deben ser separadas por ";" (punto y coma) pero no se requieren al final de cada línea. Si una línea termina con \, el salto de línea es ignorado, lo que te permite dividir líneas muy largas en varias más pequeñas.

· **Keywords** – También conocidos como términos reservados (hay alrededor de 38) no

pueden ser usados para ningún otro propósito. Tal vez estés acostumbrado a pensar que un valor falso puede representarse con un cero, una cadena vacía, null o alguna otra

cosa, pero en Ruby, todos los anteriores son valores verdaderos. De hecho, **todo** es verdadero excepto los *keywords* false y nil.

· Potentes operaciones sobre cadenas de caracteres y expresiones regulares.

· La orientación a objetos de Ruby ha sido cuidadosamente diseñada para ser completa y abierta a nuevas mejoras al mismo tiempo. Por ejemplo: Ruby tiene la habilidad de añadir métodos a una clase, o incluso a una instancia, mientras se procesa. Por lo tanto, si hace falta, una instancia de una clase **puede** actuar de diferente manera que otras estancias de la misma clase.

· Rápido y sencillo.

· Son innecesarias las declaraciones de variables.

· No necesita declaraciones de variables. Utiliza una nomenclatura sencilla para definir el alcance de una variable. Por ejemplo: un simple var = variable local,@var = variable de instancia, $var = variable global.

· La sintaxis es simple y consistente.

· Recolección de basura automática. La gestión de memoria se realiza automáticamente. Trabaja con todos los objetos de Ruby. No tiene que preocuparse por mantener una relación de referencias en las librerías de extensiones.

· Cuatro niveles de ámbito de variable: global, clase, instancia y local. Las variables siempre son referencias a objetos, no los objetos mismos.

· Modelo de procesamiento de excepciones, para que sea sencillo el manejo de errores.

· Ruby dispone de bloques en su sintaxis (código rodeado por { } o do end). Estos bloques se pueden pasar a los métodos o convertirse en cierres.

· Los integrales en Ruby pueden (y de hecho deben) ser usados sin tener en cuenta sus representaciones internas. **Son** pequeños integrales (instancias de la clase Fixnum) y grandes integrales (Bignum), pero no se debe de preocupar de cual usar en cada momento. Si un valor es lo suficientemente pequeño, un integral es un Fixnum, y de otro modo un Bignum. La conversión se produce de forma automática.

· Altamente portable: Funciona en Linux, DOS, Windows (cualquiera), MacOS, etc.

· Dispone de hebras independientes del S.O. Por lo tanto, en cualquier plataforma en la que corra Ruby, usted dispone de multihebrado (multithreading), independientemente de que el S.O. lo soporte o no, incluído MS-DOS.

· Puede cargar librerías de extensiones dinámicamente si el Sistema Operativo lo permite.

· Amplia librería estándar.

· Ruby dispone únicamente de herencias simples, *a cosa hecha*. Pero Ruby conoce los conceptos de módulos (llamados Categorías en Objective-C). Los módulos son colecciones de métodos. Cada clase puede importar un módulo y al hacerlo obtiene todos sus métodos a cambio.

1.3 RoR

1.3.1 Historia de RoR

Ruby on Rails fue escrito por David Heinemeier Hansson a partir de su trabajo en Basecamp, una herramienta de gestión de proyectos, por 37signals. En gran parte gracias al
éxito de Basecamp, 37signals se ha especializado desde entonces en desarrollo de aplicación y producción.

Al principio RoR no fue creado como framework independiente. Fue extraído de una aplicación existente que ya estaba en uso, entonces 37signals tuvo en mente usar lo que ya existía para construir otras aplicaciones. Heinemeier Hansson vio más fácil el potencial para hacer su trabajo extrayendo la funcionalidad común como la abstracción de la base de datos y más tarde se hizo la primera publicación de Ruby on Rails.

Decidió publicar Rails como código libre por "fundamentalmente para que hayan nuevas versiones de las Webs". La primera versión beta fue publicada en Julio de 2004, con RoR 1.0 se publicó el 13 de diciembre de 2005. hasta la fecha, mas de 300000 copias de Rails se descargaron y este número va en aumento.

El hecho que el framework de Rails fuera extraído de Basecamp es considerado por la comunidad de Rails que este Framework es un punto fuerte al que quieren seguir apostando.

Rails resolvió verdaderos problemas cuando fue publicado, demostró ser un framework útil, coherente y completo.

Mientras Heinemeier Hansson promocionaba Rails y todavía sigue esforzándose con todo lo relacionado con la programación de Rails, el framework ha sido beneficiado enormemente por ser código libre. Con el tiempo, los desarrolladores de Rails han desarrollado miles de extensiones para el repositorio. Este repositorio esta bien almacenado por el equipo de Rails, compuesto por doce desarrolladores profesionales, escogidos entre todos los contribuyentes y llevado por Heinemeier Hansson.

A continuación se muestra todas las publicaciones de Ruby on Rails hasta el día de hoy.

· Ruby on Rails 1.0 fue publicado el 13 de diciembre de 2005.
· Ruby on Rails 1.1 fue publicado el 28 de marzo de 2006.
· Ruby on Rails 1.2 fue publicado el 18 de enero de 2007.
· Ruby on Rails 1.8 fue publicado el 9 de noviembre de 2009.
· Ruby on Rails 1.9 fue publicado el 2 de febrero de 2010.

1.3.2 Filosofía

· Basado en el patrón MVC (Modelo – Vista – Controlador)
· DRY (Don't repeat yourself, "No te repitas")
· COC (Convención sobre configuración)

1.3.2.1 Modelo Vista Controlador

· Patrón de arquitectura de software que separa los datos de una aplicación en tres componentes distintos:

· **Modelo**: representación específica de la información con la cual el sistema opera.

· **Vista**: responsable de presentar la interfaz y la información del usuario.

· **Controlador**: organiza la aplicación. Recibe eventos del exterior, interactúa con el modelo y actualiza la información de las vistas.

· Ventajas de utilizar MVC:

· Código limpio.

· DRY.

· Facilita el trabajo en equipo.

• **Modelo**

En las aplicaciones Web orientadas a objetos sobre base de datos, el Modelo consiste en las clases que representan a las tablas de la base de datos.

En Ruby on Rails, las clases del Modelo son gestionadas por ActiveRecord, nos permite manejar todos los aspectos de la base de datos de la aplicación (conexión a la base de datos, mapeo tabla-clase y manipulación de datos). Por lo general, lo único que tiene que hacer el programador es heredar de la clase *ActiveRecord::Base*, y el programa averiguará automáticamente qué tabla usar y qué columna tiene.

Las definiciones de las clases también detallan las relaciones entre clases con sentencias de mapeo objeto relacional.

Las rutinas de validación de datos y las rutinas relacionadas con la actualización también se especifican e implementan en la clase del modelo.

El modelo representa:

· Las tablas de la Base de Datos.

· Migraciones (Expresan cambios en la BD).

· Observadores.

El patrón ActiveRecord:

· Manejo de base de datos con orientación a objetos.

· Las tablas son clases.

· Las filas son objetos.

· Las columnas son atributos.

· Sin configuración: todo por convención:

· Por ejemplo:

· pyme.name columna *name* de una fila de la tabla *pyme*.

· Métodos find().

• **Vista**

En MVC, Vista es la lógica de visualización, o cómo se muestran los datos de las clases del Controlador. Con frecuencia en las aplicaciones Web la vista consiste en una cantidad mínima de código incluido el HTML.

El método que emplea Rails, para gestionar las vistas, por defecto es usar Ruby Embebido (archivos .rhtml), que son básicamente fragmentos de código HTML con algo de código en Ruby. También pueden construirse vistas en HTML y XML con Builder o usando el sistema de plantillas Liquid.

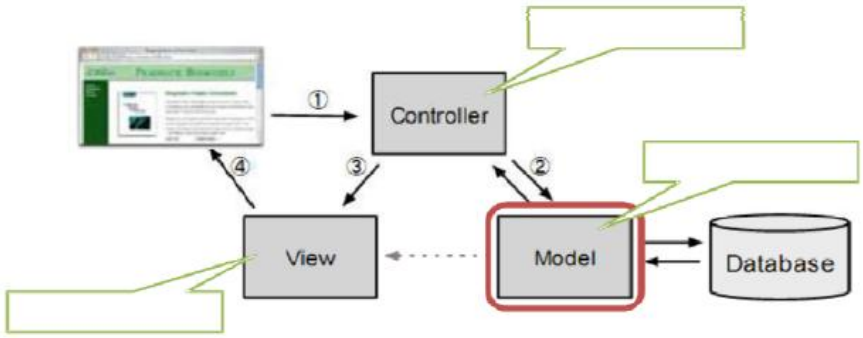
Es necesario escribir un pequeño fragmento de código en HTML para cada método del controlador que necesita información al usuario. El "maquetado" o distribución de los elementos de la página se describe separadamente de la acción del controlador y los fragmentos pueden invocarse unos a otros.

• **Controlador**

En MVC, las clases del Controlador responden a la interacción del usuario e invocan a la lógica de la aplicación, que a su vez manipula los datos de las clases del Modelo y muestra los resultados usando las Vistas. En las aplicaciones Web basadas en MVC, los métodos del controlador son invocados por el usuario usando navegador Web.

La implementación del Controlador es manejada por el ActionPack de Rails, que contiene la clase ApplicationController. Una aplicación Rails simplemente hereda de esta

clase y define las acciones necesarias como métodos, que pueden ser invocados desde la Web.



1. El navegador manda una petición
2. El controlador interactúa con el Modelo
3. El controlador invoca una Vista
4. La vista se encarga de formar toda la pagina para el navegador

Fig 1. Interacción MVC

1.3.2.2 DRY – No te repitas

· Las definiciones deberían hacerse sólo una vez.

· La repetición innecesaria de conocimiento es fuente de errores (incongruencias).

· En Ruby on Rails los componentes están integrados de manera que no hace falta establecer puentes entre ellos.

Java

```
public class Conferencia{
   private String nombre;
   private Date fecha;

   public void setNombre(String n){
      this.nombre=n;
   }

   public String getNombre(){
      return nombre;
   }

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

   public Date getFecha(){
      return fecha;
   }
}
```

Ruby

```
class Conferencia
   attr_accessor :nombre, :fecha
end
```

Fig 2. Mismo ejemplo en Java y en Ruby

Como podemos ver en la imagen en Java para mostrar el nombre y la fecha, se han de crear cuatro métodos, dos para recoger el valor y otros dos para mostrarlos.

En cambio en RoR lo hace automáticamente con la sintaxis attr_accessor, que hace que los atributos sean legibles y se puedan escribir.

1.3.2.3 COC – Convención sobre configuración

· El programador sólo necesita definir aquella configuración que no es convencional.

· En vez de requerir innumerables archivos de configuración ".xml", Rails propone defaults" razonables, que podemos cambiar.

· Algunos tipos de datos, campos sugeridos (id, created at y updated at, etc.)

· Cómo se llaman las tablas y las clases, así como las tablas de relación (y qué va en singular, qué va en plural | y cómo se determina cómo singularizar/pluralizar)· Cómo se organizan los archivos en nuestro árbol.

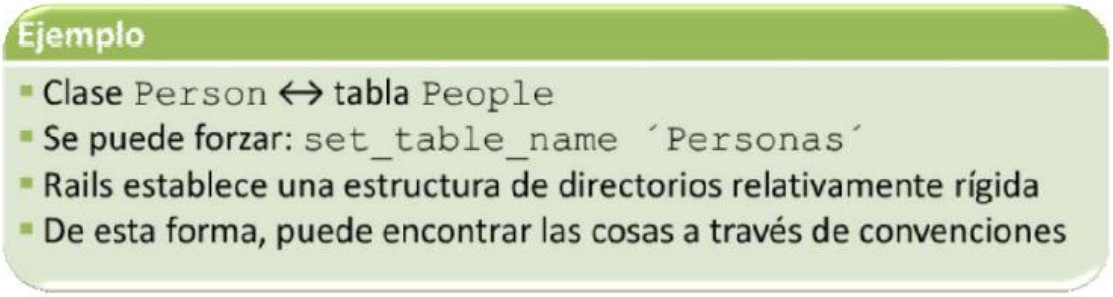


Fig 3. Ejemplo COC

1.4 Migraciones

· Clases de Ruby para crear, modificar y eliminar tablas.

· Permite independizar tu BBDD del motor de BBDD que utilices.

· Permite ejecutar cambios en tu modelo de datos sin esfuerzo.

· Rails soporta las principales bases de datos como:

· MySQL

· Sqlite3

· Oracle

· Postgre SQL

· SQL Server

· Sintaxis de la clase de migración:

· Las acciones disponibles son las siguientes:

· Sobre las tablas: create_table, drop_table, rename_table.

· Sobre las columnas: add_column, remove_column, rename_column, change_column.

· Sobre los indíces: add_index, remove_index.

· Se definen dos métodos: self.up y self.down: Todo lo que hagamos en nuestro método up lo debemos deshacer en el método down.

· Al crear un modelo, entre los archivos generados se encuentra una migración. Las migraciones son creadas en orden, por lo que los archivos serán numerados conforme se vayan creando, empezando por 001.

· Principal ventaja: actúan como si fueran versiones de nuestra aplicación. Si nos equivocamos en algo y queremos regresar a la versión anterior sólo tenemos que recrear nuestro esquema de tablas que teníamos para esa versión.

· Para "navegar" entre versiones tenemos algunas tareas rake:

· **db:migrate [VERSION=n]** Crea el esquema de tablas.

· **db:migrate:redo [STEP=n**] Regresa a la versión previa y vuelve a migrarla.

· **db:migrate:reset** Borra tu db, la vuelve a crear, y migra las tablas .

· **db:rollback [STEP=n]** Regresa a la versión previa.

1.5 El patrón ActionController

· Procesa la URL solicitada.

· Dirige la petición al controlador correspondiente.

· El controlador realiza la tarea correspondiente, solicitando al Modelo los datos que necesite.

· Renderiza la plantilla (la Vista).

1.6 El patrón ActionView

· Se encarga de convertir los datos que le pasa el controlador en el HTML que se servirá al navegador.

· Es necesario escribir un pequeño fragmento de código en HTML para cada método del controlador que necesita mostrar información al usuario.

· Existen muchas maneras de gestionar las vistas. En Rails se usa Ruby Embebido (archivos .rhtml), fragmentos de código HTML con algo de código en Ruby.

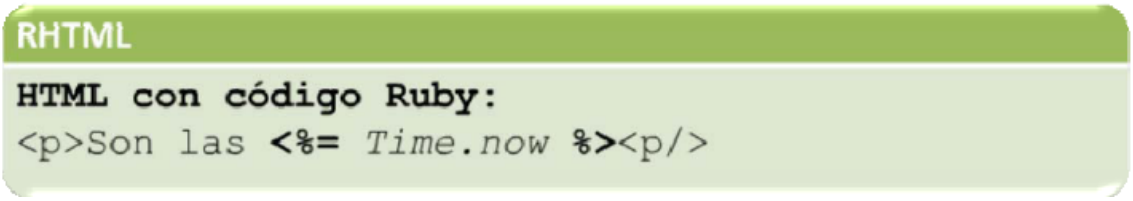


Fig 4. Ejemplo RHTML

1.7 Helpers

· Permiten minimizar el código utilizado en la presentación, ya que encapsulan grandes bloques de código en llamadas simples a funciones.

· Generación de URLs, de formularios.

· Formateo de datos.

· Paginación.

1.8 Plugin

Un plugin en Rails puede ser tanto una extensión como una modificación del framework. El plugin proporciona:

· un camino para los desarrolladores para compartir ideas sin modificar la base del código.

· una arquitectura segmentada dónde las unidades de código puedan ser fijadas o actualizadas sobre su propia lista.

· una salida para los desarrolladores que no tienen que incluir todos los rasgos nuevos.

· DRY: muchos plugins nacen por evitar la repetición sistemática de código en distintas aplicaciones y en una misma aplicación.

• **Ejemplos de plugins:**

· **tog_user:** plugin encargado de la gestión del sistema

· **tog_social:** extensión que añade soporte de base de red social a su sitio.

1.9 RubyGems

RubyGems es un gestor de paquetes para el lenguaje de programación Ruby que proporciona un formato estándar y autocontenido, llamado gem, para poder distribuir programas o librerías Ruby, una herramienta destinada a gestionar la instalación de estos, y un servidor para su distribución. Es, por tanto, equivalente al papel que juegan CPAN y PEAR en los lenguajes Perl y PHP, respectivamente.

1.9.1 Ejemplos de RubyGems

· **Ferret**: buscador de alta precisión basado en Apache Lucene.

· **Mongrel**: servidor http ligero creado para soportar aplicaciones en Ruby y muy extendido entre aplicaciones en producción. Este será usado por nuestra red social.

· **Rmagick**: es una interfaz entre el lenguaje de programación de Ruby y el ImageMagick y GraphicsMagick que son las librerías de procesamiento de imágenes. Nosotros usaremos esta gema en nuestra red social.

1.10 Helpers

Un Helper es un módulo que ayuda a tus vistas definiendo funciones que ayudan a que tus vistas sean más que nada HTML y no contengan demasiado código. En un

sistema de modelo MVC, la idea es que la vista (la "V" de MVC) sea tan simple como sea posible.

Rails viene con varios Helpers, y tu aplicación tiene un "application_helper" para que tengas ayudas específicas a tu aplicación. Normalmente los helpers producen contenido para el HTML or Javascript de tu aplicación.

2. Comparación otros lenguajes

2.1 PHP

PHP es un lenguaje de programación interpretado usado normalmente para la creación de páginas Web dinámicas, habitualmente es combinado con el motor MYSQL, aunque también cuenta con soporte nativo para otros motores, incluyendo el ODBC, lo que amplía en gran medida sus posibilidades de conexión.

Fig 5. ¿Cómo funciona PHP?

2.1.1 Como funciona PHP

PHP se ejecuta en el servidor, esto quiere decir que no es necesario que el navegador soporte este lenguaje, pero el servidor ha de soportar este lenguaje.

Al ser ejecutado en el servidor, este permite acceder a los recursos que tenga, como por ejemplo la base de datos, y el resultado es enviado al navegador.

2.1.2 Ventajas

· La sintaxis de PHP es similar a la de C, por esto cualquier con experiencia en lenguajes al estilo C, (Java, Javascript) podrá entender rápidamente PHP.

· PHP corre en casi todas las plataformas utilizando el mismo código fuente, pudiendo ser compilado y ejecutado en algo así como 25 plataformas, incluyendo diferentes versiones de Unix, Windows y Macs. Como en todos los sistemas se utiliza el mismo código base, los scripts pueden ser ejecutados de manera independiente al OS.

· Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad, destaca su conectividad con MYSQL.

· Capacidad de expandir su potencial utilizando la enorme cantidad de módulos. Cuando un programador en PHP necesite una interfaz para una librería en particular, fácilmente podrá crear una API para esta. Algunas de las que vienen implementadas permiten manejo de gráficos, archivos, calendarios, etc.

· Posee una amplia documentación.

· Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.

· Permite las técnicas de Programación Orientada a Objetos.

· Biblioteca nativa de funciones sumamente amplia e incluida.

· No requiere definición de tipos de variables.

· Tiene manejo de excepciones.

· Rapidez, PHP generalmente es utilizado como módulo de Apache, lo que lo hace extremadamente veloz. Esta completamente escrito en C, así que se ejecuta rápidamente utilizando poca memoria.

2.1.3 Desventajas

· No posee una abstracción de base de datos estándar, sino bibliotecas especializadas para cada motor (a veces más de una para el mismo motor).

· No posee adecuado manejo de unicode.

· Por su diseño dinámico no puede ser compilado y es muy difícil de optimizar.

· Por sus características promueve la creación de código desordenado y complejo de mantener.

· Está diseñado especialmente para un modo de hacer aplicaciones Web que es ampliamente considerado problemático y obsoleto.

2.2 Java

• Es un lenguaje de programación que ofrece la potencia del diseño orientado a objetos con una sintaxis fácilmente accesible y un entorno robusto y agradable.
• Proporciona un conjunto de clases potente y flexible.
• Pone al alcance de cualquiera la utilización de aplicaciones que se pueden incluir directamente en páginas Web (aplicaciones denominadas *applets*).

Java aporta a la Web una interactividad que se había buscado durante mucho tiempo entre usuario y aplicación

2.2.1 Características

• **SIMPLE**:

Java ofrece toda la funcionalidad de un lenguaje potente, pero sin las características menos usadas y más confusas de éstos. C++ es un lenguaje que adolece de falta de seguridad, pero C y C++ son lenguajes más difundidos, por ello Java se diseñó para ser parecido a C++ y así facilitar un rápido y fácil aprendizaje.

Java elimina muchas de las características de otros lenguajes como C++, para mantener reducidas las especificaciones del lenguaje y añadir características muy útiles como el *garbage collector* (reciclador de memoria dinámica). No es necesario preocuparse de liberar memoria, el reciclador se encarga de ello y como es un *thread* de baja prioridad, cuando entra en acción, permite liberar bloques de memoria muy grandes, lo que reduce la fragmentación de la memoria.

Java reduce en un 50% los errores más comunes de programación con lenguajes como C y C++ al eliminar muchas de las características de éstos, entre las que destacan:

- aritmética de punteros
- no existen referencias
- registros (struct)
- definición de tipos (typedef)
- macros (#define)
- necesidad de liberar memoria (free)

Aunque, en realidad, lo que hace es eliminar las palabras reservadas (struct, typedef),
ya que las clases son algo parecido.

Además, el intérprete completo de Java que hay en este momento es muy pequeño, solamente ocupa 215 Kb de RAM.

• **ORIENTADO A OBJETOS**:

Java implementa la tecnología básica de C++ con algunas mejoras y elimina algunas cosas para mantener el objetivo de la simplicidad del lenguaje. Java trabaja con sus datos como objetos y con interfaces a esos objetos. Soporta las tres características propias del paradigma de la orientación a objetos: encapsulación, herencia y polimorfismo. Las plantillas de objetos son llamadas, como en C++, *clases* y sus copias, *instancias*. Estas instancias, como en C++, necesitan ser construidas y destruidas en espacios de memoria.

Java incorpora funcionalidades inexistentes en C++ como por ejemplo, la resolución dinámica de métodos. Esta característica deriva del lenguaje Objective C, propietario del sistema operativo Next. En C++ se suele trabajar con librerías dinámicas (DLLs) que obligan a recompilar la aplicación cuando se retocan las funciones que se encuentran en su interior. Este inconveniente es resuelto por Java mediante una interfaz específica llamada RTTI (*RunTime Type Identification*) que define la interacción entre objetos excluyendo variables de instancias o implementación de métodos. Las clases en Java tienen una representación en el runtime que permite a los programadores interrogar por el tipo de clase y enlazar dinámicamente la clase con el resultado de la búsqueda.

• **DISTRIBUIDO**:

Java se ha construido con extensas capacidades de interconexión TCP/IP. Existen librerías de rutinas para acceder e interactuar con protocolos como *http* y *ftp* . Esto permite a los programadores acceder a la información a través de la red con tanta facilidad como a los ficheros locales.

La verdad es que Java en sí no es distribuido, sino que proporciona las librerías y herramientas para que los programas puedan ser distribuidos, es decir, que se corran en varias máquinas, interactuando.

• **ROBUSTO**:

Java realiza verificaciones en busca de problemas tanto en tiempo de compilación como en tiempo de ejecución. La comprobación de tipos en Java ayuda a detectar errores, lo antes posible, en el ciclo de desarrollo. Java obliga a la declaración explícita de métodos, reduciendo así las posibilidades de error. Maneja la memoria para eliminar las preocupaciones por parte del programador de la liberación o corrupción de memoria. También implementa los *arrays auténticos* , en vez de listas enlazadas de punteros, con comprobación de límites, para evitar la posibilidad de sobreescribir o corromper memoria resultado de punteros que señalan a zonas equivocadas. Estas características reducen drásticamente el tiempo de desarrollo de aplicaciones en Java.

Además, para asegurar el funcionamiento de la aplicación, realiza una verificación de los *byte-codes* , que son el resultado de la compilación de un programa Java. Es un código de máquina virtual que es interpretado por el intérprete Java. No es el código máquina directamente entendible por el hardware, pero ya ha pasado todas las fases del compilador: análisis de instrucciones, orden de operadores, etc., y ya tiene generada la pila de ejecución de órdenes.

Java proporciona, pues:

- Comprobación de punteros

- Comprobación de límites de arrays
- Excepciones
- Verificación de byte-codes

• **ARQUITECTURA NEUTRAL**:

Para establecer Java como parte integral de la red, el compilador Java compila su código a un fichero objeto de formato independiente de la arquitectura de la máquina en que se ejecutará. Cualquier máquina que tenga el sistema de ejecución (*run-time*) puede ejecutar ese código objeto, sin importar en modo alguno la máquina en que ha sido generado. Actualmente existen sistemas run-time para Solaris 2.x, SunOs 4.1.x, Windows 95, Windows NT, Linux, Irix, Aix, Mac, Apple y probablemente haya grupos de desarrollo trabajando en el porting a otras plataformas.

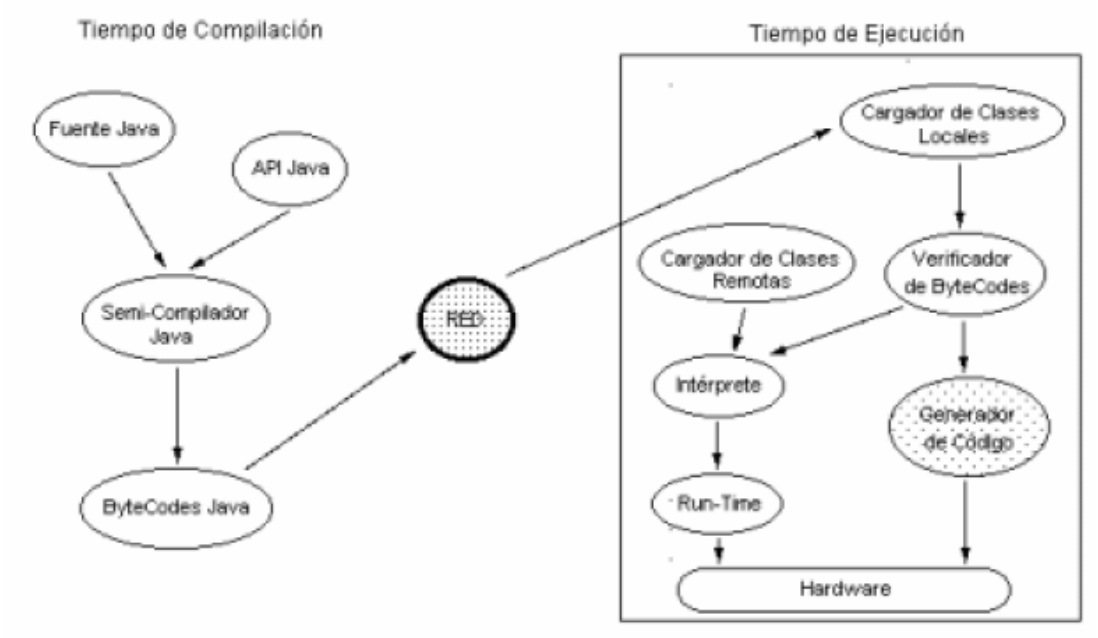


Fig 6. Explicación de Tiempo de compilación y Tiempo de Ejecución en Java

El código fuente Java se "compila" a un código de bytes de alto nivel independiente de la máquina. Este código (byte-codes) está diseñado para ejecutarse en una máquina hipotética que es implementada por un sistema run-time, que sí es dependiente de la máquina.

En una representación en que tuviésemos que indicar todos los elementos que forman parte
de la arquitectura de Java sobre una plataforma genérica, obtendríamos una figura como la siguiente:

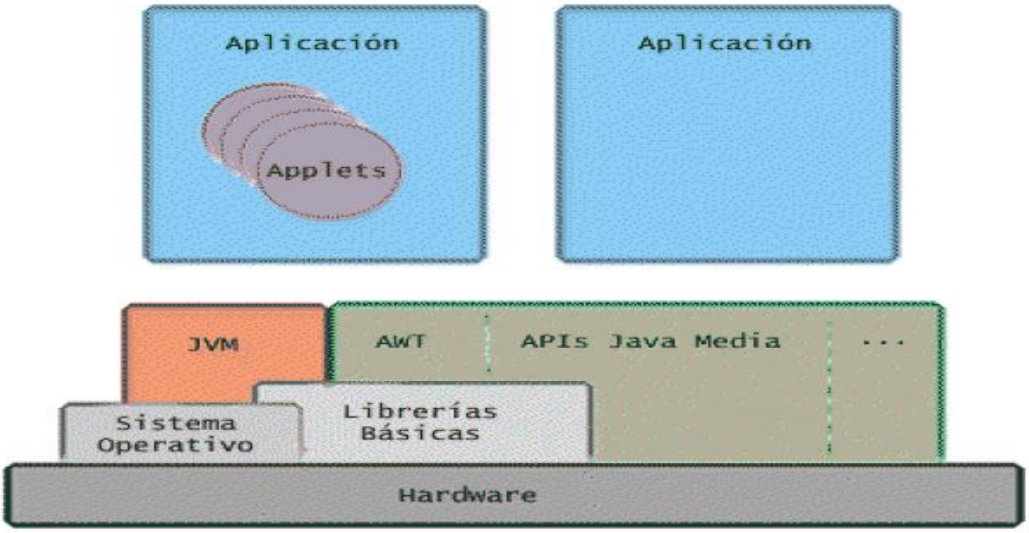


Fig 7. Arquitectura de Java

En ella podemos ver que lo verdaderamente dependiente del sistema es la *Máquina Virtual Java* (JVM) y las librerías fundamentales, que también nos permitirían acceder directamente al hardware de la máquina. Además, habrá APIs de Java que también entren en contacto directo con el hardware y serán dependientes de la máquina, como ejemplo de este tipo de APIs podemos citar:

- Java 2D: gráficos 2D y manipulación de imágenes
- Java Media Framework : Elementos críticos en el tiempo: audio, video...
- Java Animation: Animación de objetos en 2D
- Java Telephony: Integración con telefonía
- Java Share: Interacción entre aplicaciones multiusuario
- Java 3D: Gráficos 3D y su manipulación

• **SEGURO**:

La seguridad en Java tiene dos facetas. En el lenguaje, características como los punteros o el *casting* implícito que hacen los compiladores de C y C++ se eliminan para prevenir el acceso ilegal a la memoria. Cuando se usa Java para crear un navegador, se combinan las características del lenguaje con protecciones de sentido común aplicadas al propio navegador.

El lenguaje C, por ejemplo, tiene lagunas de seguridad importantes, como son los *errores de alineación* . Los programadores de C utilizan punteros en conjunción con operaciones aritméticas. Esto le permite al programador que un puntero referencie a un lugar conocido de la memoria y pueda sumar (o restar) algún valor, para referirse a otro lugar de la memoria. Si otros programadores conocen nuestras estructuras de datos pueden extraer información confidencial de nuestro sistema. Con un lenguaje como C, se pueden tomar números enteros aleatorios y convertirlos en punteros para luego acceder a la memoria:

```
printf( "Escribe un valor entero: " );
scanf( "%u",&puntero );
printf( "Cadena de memoria: %sn",puntero );
```

Otra laguna de seguridad u otro tipo de ataque, es el *Caballo de Troya* . Se presenta un programa como una utilidad, resultando tener una funcionalidad destructiva. Por ejemplo, en UNIX se visualiza el contenido de un directorio con el comando *ls* . Si un programador deja un comando destructivo bajo esta referencia, se puede correr el riesgo de ejecutar código malicioso, aunque el comando siga haciendo la funcionalidad que se le supone, después de lanzar su carga destructiva. Por ejemplo, después de que el caballo de Troya haya enviado por correo el /etc/shadow a su creador, ejecuta la funcionalidad de *ls* persentando el contenido del directorio. Se notará un retardo, pero nada inusual.

El código Java pasa muchos *tests* antes de ejecutarse en una máquina. El código se pasa a través de un verificador de byte-codes que comprueba el formato de los fragmentos de código y aplica un probador de teoremas para detectar fragmentos de

código ilegal - código que falsea punteros, viola derechos de acceso sobre objetos o intenta cambiar el tipo o clase de un objeto-.

Si los byte-codes pasan la verificación sin generar ningún mensaje de error, entonces sabemos que:

- El código no produce desbordamiento de operandos en la pila
- El tipo de los parámetros de todos los códigos de operación son conocidos y correctos
- No ha ocurrido ninguna conversión ilegal de datos, tal como convertir enteros en punteros
- El acceso a los campos de un objeto se sabe que es legal: public, private, protected
- No hay ningún intento de violar las reglas de acceso y seguridad establecidas

El Cargador de Clases también ayuda a Java a mantener su seguridad, separando el espacio de nombres del sistema de ficheros local, del de los recursos procedentes de la red. Esto limita cualquier aplicación del tipo *Caballo de Troya* , ya que las clases se buscan primero entre las locales y luego entre las procedentes del exterior.

Las clases importadas de la red se almacenan en un espacio de nombres privado, asociado con el origen. Cuando una clase del espacio de nombres privado accede a otra clase, primero se busca en las clases predefinidas (del sistema local) y luego en el espacio de nombres de la clase que hace la referencia. Esto imposibilita que una clase suplante a una predefinida.

En resumen, las aplicaciones de Java resultan extremadamente seguras, ya que no acceden a zonas delicadas de memoria o de sistema, con lo cual evitan la interacción de ciertos virus. Java no posee una semántica específica para modificar la pila de programa, la memoria libre o utilizar objetos y métodos de un programa sin los privilegios del kernel del sistema operativo. Además, para evitar modificaciones por parte de los crackers de la red, implementa un método ultraseguro de autentificación por clave pública. El Cargador de Clases puede verificar una firma digital antes de realizar una

instancia de un objeto. Por tanto, ningún objeto se crea y almacena en memoria, sin que se validen los privilegios de acceso. Es decir, la seguridad se integra en el momento de compilación, con el nivel de detalle y de privilegio que sea necesario.

Dada, pues la concepción del lenguaje y si todos los elementos se mantienen dentro del estándar marcado por Sun, no hay peligro. Java imposibilita, también, abrir ningún fichero de la máquina local (siempre que se realizan operaciones con archivos, éstas trabajan sobre el disco duro de la máquina de donde partió el applet), no permite ejecutar ninguna aplicación nativa de una plataforma e impide que se utilicen otros ordenadores como puente, es decir, nadie puede utilizar nuestra máquina para hacer peticiones o realizar operaciones con otra. Además, los intérpretes que incorporan los navegadores de la Web son aún más restrictivos. Bajo estas condiciones (y dentro de la filosofía de que el único ordenador seguro es el que está apagado, desenchufado, dentro de una cámara acorazada en un bunker y rodeado por mil soldados de los
cuerpos especiales del ejército), se puede considerar que Java es un lenguaje seguro y que los applets están libres de virus.

Respecto a la seguridad del código fuente, no ya del lenguaje, JDK proporciona un desensamblador de byte-code, que permite que cualquier programa pueda ser convertido a código fuente, lo que para el programador significa una vulnerabilidad total a su código. Utilizando *javap* no se obtiene el código fuente original, pero sí desmonta el programa mostrando el algoritmo que se utiliza, que es lo realmente interesante. La protección de los programadores ante esto es utilizar llamadas a programas nativos, externos (incluso en C o C++) de forma que no sea descompilable todo el código; aunque así se pierda portabilidad. Esta es otra de las cuestiones que Java tiene pendientes.

• **PORTABLE**:

Más allá de la portabilidad básica por ser de arquitectura independiente, Java implementa otros estándares de portabilidad para facilitar el desarrollo. Los enteros son siempre *enteros* y además, enteros de 32 bits en complemento a 2. Además, Java

construye sus interfaces de usuario a través de un sistema abstracto de ventanas de forma que las ventanas puedan ser implantadas en entornos Unix, Pc o Mac.

• **INTERPRETADO**:

El intérprete Java (sistema run-time) puede ejecutar directamente el código objeto. Enlazar (linkar) un programa, normalmente, consume menos recursos que compilarlo, por lo que los desarrolladores con Java pasarán más tiempo desarrollando y menos esperando por el ordenador. No obstante, el compilador actual del JDK es bastante lento. Por ahora, que todavía no hay compiladores específicos de Java para las diversas plataformas, Java es más lento que otros lenguajes de programación, como C++, ya que debe ser interpretado y no ejecutado como sucede en cualquier programa tradicional.

Se dice que Java es de 10 a 30 veces más lento que C, y que tampoco existen en Java proyectos de gran envergadura como en otros lenguajes. La verdad es que ya hay comparaciones ventajosas entre Java y el resto de los lenguajes de programación, y una ingente cantidad de folletos electrónicos que supuran fanatismo en favor y en contra de los distintos lenguajes contendientes con Java. Lo que se suele dejar de lado en todo esto, es que primero habría que decidir hasta que punto Java, un lenguaje en pleno desarrollo y todavía sin definición definitiva, está maduro como lenguaje de programación para ser comparado con otros; como por ejemplo con Smalltalk, que lleva más de 20 años en cancha.

La verdad es que Java para conseguir ser un lenguaje independiente del sistema operativo y del procesador que incorpore la máquina utilizada, es tanto interpretado como compilado. Y esto no es ningún contrasentido, me explico, el código fuente escrito con cualquier editor se compila generando el byte-code. Este código intermedio es de muy bajo nivel, pero sin alcanzar las instrucciones máquinas propias de cada plataforma y no tiene nada que ver con el p-code de Visual Basic. El byte-code corresponde al 80% de las instrucciones de la aplicación. Ese mismo código es el que se puede ejecutar sobre cualquier plataforma. Para ello hace falta el run-time, que sí es completamente dependiente de la máquina y del sistema operativo, que interpreta dinámicamente el byte-code y añade el 20% de instrucciones que faltaban para su ejecución. Con este sistema es

fácil crear aplicaciones multiplataforma, pero para ejecutarlas es necesario que exista el run-time correspondiente al sistema operativo utilizado.

• **MULTITHREADED**:

Al ser multithreaded, Java permite muchas actividades simultáneas en un programa. Los threads (a veces llamados, procesos ligeros), son básicamente pequeños procesos o piezas independientes de un gran proceso. Al estar los threads construidos en el lenguaje, son más fáciles de usar y más robustos que sus homólogos en C o C++.

El beneficio de ser multithreaded consiste en un mejor rendimiento interactivo y mejor comportamiento en tiempo real. Aunque el comportamiento en tiempo real está limitado a las capacidades del sistema operativo subyacente (Unix, Windows, etc.), aún supera a los entornos de flujo único de programa (single-threaded) tanto en facilidad de desarrollo como en rendimiento.

Cualquiera que haya utilizado la tecnología de navegación concurrente, sabe lo frustrante que puede ser esperar por una gran imagen que se está trayendo. En Java, las imágenes se pueden ir trayendo en un thread independiente, permitiendo que el usuario pueda acceder a la información en la página sin tener que esperar por el navegador.

• **DINÁMICO**:

Java se beneficia todo lo posible de la tecnología orientada a objetos. Java no intenta conectar todos los módulos que comprenden una aplicación hasta el tiempo de ejecución. Las librerías nuevas o actualizadas no paralizarán las aplicaciones actuales (siempre que mantengan el API anterior).

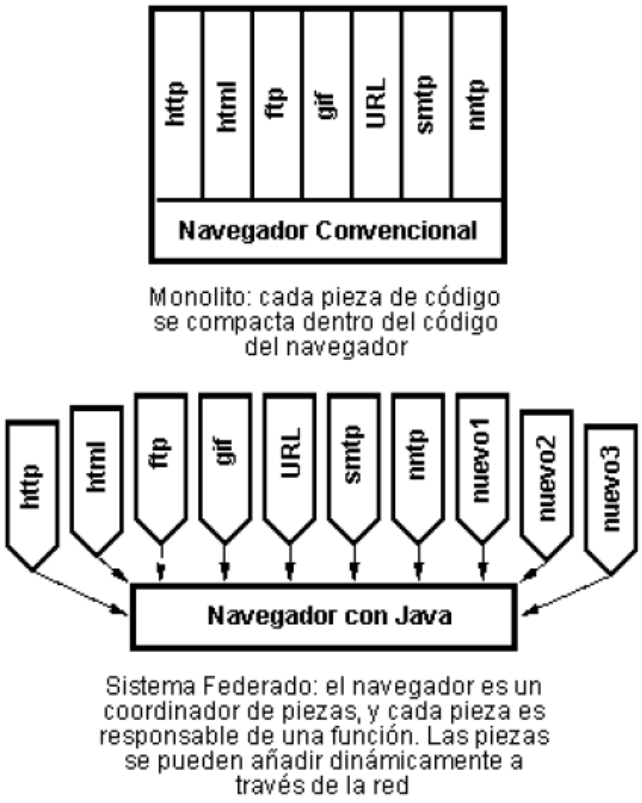


Fig 8. Diferencia entre Navegador Convencional y Navegador con Java

Java también simplifica el uso de protocolos nuevos o actualizados. Si su sistema ejecuta una aplicación Java sobre la red y encuentra una pieza de la aplicación que no sabe manejar, tal como se ha explicado en párrafos anteriores, Java es capaz de traer automáticamente cualquiera de esas piezas que el sistema necesita para funcionar.

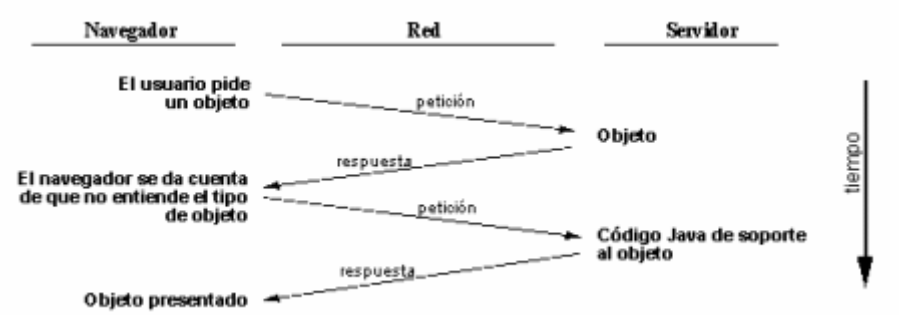


Fig 9. Ejecución de una aplicación Java sobre la Red

Java, para evitar que los módulos de byte-codes o los objetos o nuevas clases, haya que estar trayéndolos de la red cada vez que se necesiten, implementa las opciones de persistencia, para que no se eliminen cuando de limpie la caché de la máquina.

2.3 Ruby & Java

Fig 10. Libros de Java y Libros de Ruby on Rails

2.3.1 Similitudes

- La memoria es manejada por garbage collector (limpiador de basura)
- Los métodos son públicos, privados y protegidos.
- La documentación generada por rdoc es muy similar a la utilizada por javadoc.

2.3.2 Diferencias

A diferencia de Java, en Ruby:

· No se necesita compilar el código.

· Hay diferentes tipos de GUI. Ruby puedes usar WxRuby, FXRuby, Ruby-GNOME2.

· Se usa la palabra *end* después de definir las clases, en vez de utilizar llaves en bloques de código.

· Se usa *require* en vez de *import*.

· Todas las variables son privadas.

· Los paréntesis en las llamadas a métodos, normalmente son opcionales y otras omitidos.

· Todo es un objeto.

· No hay ninguna comprobación de tipo estática.

· Los nombres de las variables son sólo etiquetas, no hace falta que se les asocie ningún tipo.

· No hay ningún tipo de declaraciones, sólo asigna los nombres de las nuevas variables cuando lo necesita y ellas solas se crean.

· A la hora de crear un objeto hay pequeñas diferencias:

  · Es foo = Foo.new(“hi”) en vez de Foo foo= new foo(“hi”)

· El constructor siempre se llama “initialize”, en vez de cómo el nombre de la clase.

· Es nil en vez de nulo.

· == y equal() tienen un significado diferente en Ruby. Se usa == cuando se quiere comprobar la equivalencia en Ruby (en Java equals()). Se utiliza equal?() cuando quieres saber si dos objetos son los mismos (en Java ==)

2.4 Ruby & PHP

2.4.1 Similitudes

· Ruby es dinámico, como PHP, por lo tanto no necesitas preocuparte por la declaración de variables.

· Hay clases, y podemos controlar el acceso como en PHP 5 (públicos, privados y protegidos)

· Las variables se inicializan con $, igual que en PHP 5.

· Se utiliza eval en los dos lenguajes.

· Ruby tiene excepciones, igual que PHP 5.

· Tienen una amplia librería estándar.

2.4.2 Diferencias

A diferencia de PHP, en Ruby:

· Necesitas llamar to_s, to_i, etc, para convertir los strings o enteros, en vez de confiar en el lenguaje para hacerlo.

· En vez de abs(-1) se utiliza -1.abs.

· Los paréntesis son opcionales cuando llamas a los métodos, excepto cuando se clarifica que parámetros van al método que llaman.

· Las variables son referencias.

· No hay clases abstractas.

· Los hash y los arrays no son intercambiables.

3. Integración

3.1 Ajax

AJAX no es un lenguaje exactamente su nombre viene dado por el acrónimo de Asynchronous JavaScript And XML y es posiblemente la mayor novedad en cuanto a programación Web en estos últimos años.

El corazón de Ajax es el objeto XMLHttpRequest que nos permite realizar una conexión al servidor y al enviarle una petición y recibir la respuesta que procesaremos en nuestro código Javascript, estamos hablando del verdadero motor de Ajax, por ejemplo gracias a este objeto podemos desde una página HTML leer datos de una web o enviar datos de un formulario sin necesidad de recargar la página.

Puedes programar numerosas nuevas aplicaciones enfocadas desde una visión distinta como es el caso de este paginador AJAX, si esto no te convence a leer este artículo prueba a ver 10 razones para usar AJAX.

1. Basado en los estándares abiertos.
2. Usabilidad.
3. Válido en cualquier plataforma y navegador.
4. Beneficia las aplicaciones Web.
5. No es difícil su utilización.
6. Compatible con Flash.
7. Adoptado por los "gordos" de la tecnología Web.
8. Web 2.0.
9. Es independiente del tipo de tecnología de servidor que se utilice.
10. Mejora la estética de la Web.

3.1.1 Ajax on Rails

Para implementar RoR con Ajax es bastante simple, ya que RoR tiene un modelo que implementa las operaciones de Ajax.

Una vez el buscador muestra la página inicia, las diferentes acciones del usuario provoca que se abra una nueva página (tradicional) o accionar un trigger en Ajax:

· Una acción trigger ocurre. Esto se puede provocar porque el usuario ha clicado en botón o link,

· Los datos se asocian al trigger, estos se envían asincrónicamente en el servidor vía XMLHttpRequest.

· El servidor recoje la acción basada en los datos, y devuelve un fragmento HTML como respuesta.

· El cliente Javascript (creado automáticamente por Rails) recibe el HTML y lo utiliza para actualizar una parte especificada de las páginas HTML, a menudo el contenido de una etiqueta.

Rails tiene varios métodos de ayuda para implementar Ajax en nuestras plantillas. Un método sencillo es *link_to_remote()* y *javascript_include_tag()* es un método que utiliza las librerías de Javascript.

Otras características de Ajax en Rails son las siguientes:

· RoR utiliza una libreria javascript, Prototype.

· Su autor es Sam Stephenson.

· La abstracción con respecto al navegador que ejecuta el código es completa.

· RoR abstrae mucha funcionalidad de Prototype, por lo que pocas veces necesitaremos conocer Javascript.

· Script.aculo.us es un conjunto de objetos Javascript para escribir efectos visuales.

· Thomas Fuchs es el artífice de este proyecto.

· Está basado en Prototype y gracias a eso, aprovecha toda la potencia de AJAX.

· Algunos efectos están disponibles directamente desde Ruby on Rails, sin necesidad de escribir código Javascript.

4. Rails 2.0

4.1 Recursos

El primer cambio que encontramos con las versions anteriores a Rails es a la hora de crear los recursos.

./script/generate scaffold Post title:string body:text

En Rails 2.0 se genera con RESTful. La única diferencia es que ‘scaffold’ se comporta como el ‘scaffold_resource’ que teníamos antes y el antiguo ‘non_RESTful scaffold’ se ha eliminado. Tampoco se utiliza la clase ActionController, este dinámicamente rellenaba los controladores vacíos con acciones. Por lo tanto, todo ‘scaffold’ que hacemos ahora es RESTful.

Esto creará las típicas clases:

· Controladores.
· Helpers.
· Modelos.
· Migraciones.
· Test de unidad.
· Test de funcionalidad.

La principal diferencia está en la Migración.

```
# db/migrate/001_create_posts.rb

class CreatePosts < ActiveRecord::Migration
  def self.up
    create_table :posts do |t|
      t.string :title
      t.text :body
      t.timestamps
    end
  end

  def self.down
    drop_table :posts
  end
end
```

A esto se denomina ‘Migración Sexy’, primero proporcionado por ‘Err the Blog’ como plugin y posteriormente fue incorporado al núcleo.

Un ejemplo de una migración con Rails 1.2.

```
class CreatePosts < ActiveRecord::Migration
  def self.up
    create_table :posts do |t|
```

```
      t.column :title, :string
      t.column :body, :text
      t.column :created_at, :datetime
      t.column :updated_at, :datetime
    end
  end

  def self.down
    drop_table :posts
  end
end
```

Se deshace de la repetición de ‘t.column’ y ahora usa la forma ‘t.tipo_columna’ y la columna automática datetime está determinada en un simple ‘t.timestamps’. Esta forma de crear las columnas de la tabla, hace que el código quede más ‘sexy’.

Otra diferencia, es a la hora de deshacer una migración. Con Rails 1.2 se hacía de la siguiente manera.

· rake db:migrate VERSION=xxx

Ahora simplemente es:

· rake db:rollback

Mucho más bonito y elegante.

4.2 Rutas anidadas

Para ver las diferencias entre Rails 1.2 y Rails 2.0, la mejor manera es poniendo un ejemplo.

Creamos un recurso Comentario para el Post:

```
./script/generate scaffold Comment post:references body:text

rake db:mígrate
```

Lo mismo de antes, el recurso ‘scaffold’, configura los nombres de las columnas y el tipo en la línea de comando, y el archivo de migración se ejecuta inmediatamente. Otro pequeño cambio es la palabra ‘references’.

Comparemos lo mismo pero con código antiguo.

```
./script/generate scaffold Comment post_id:integer body:text
```

El archive con la nueva migración es el siguiente:

```
def self.up
    create_table :comments do |t|
      t.references :post
      t.text :body
      t.timestamps
    end
end
```

La palabra referente añade una columna nombre_id, más una correspondencia tipo_columna si la opción: polymorphic es suministrada.

Una vez ejecutado db:migrate se crea la tabla en la base de datos. Luego configuramos el modelo ActiveRecord para que se relacionen entre sí.

```
# app/models/post.rb

class Post < ActiveRecord::Base
    has_many :comments
```

```
end

# app/models/comment.rb

class Comment < ActiveRecord::Base
    belongs_to :post
end
```

En Rails 2.0, podemos indicarle que queremos unas URLs como las siguientes:

http://localhost:3000/posts/1/comments
http://localhost:3000/posts/1/comments/new
http://localhost:3000/posts/1/comments/3

El recurso scaffold sólo genera URLs del tipo:

http://localhost:3000/posts/1
http://localhost:3000/comments/new
http://localhost:3000/comments/3

Esto es debido a que en el archivo config/routes.rb tenemos lo siguiente:

```
# config/routes.rb

ActionController::Routing::Routes.draw do |map|
    map.resources :comments
    map.root :controller => 'posts'
    map.resources :posts
end
```

En los modelos, podemos crear una Ruta Anidada.

```
# config/routes.rb
```

```
ActionController::Routing::Routes.draw do |map|
    map.root :controller => 'posts'
    map.resources :posts, :has_many => :comments
end
```

De esta manera podemos crear URLs anidadas como las que se han mostrado anteriormente.

Hemos de entender que cuando nosotros escribimos la URL.:

http://localhost:3000/posts/1/comments

Rails lo interpreta de la siguiente manera:

· Carga el controlador Comments.
· Carga el parámetro [:post_id] = 1
· Y finalmente, llama a la acción *index*.

Ahora tendríamos que hacer que el Controlador Comment sea anidado. Para esto hemos de cambiar lo siguiente:

```
class CommentsController < ApplicationController
    before_filter :load_post
    ...

    def load_post
      @post = Post.find(params[:post_id])
    end
end
```

Esto hace que el @post se utilice en todas las acciones dentro del controlador Comment.

Podemos comparar dos archivos entre Rails 2.0 y Rails 1.2.

En Rails 2.0:

```
# edit.html.erb and new.html.erb
    form_for(@comment) do |f|
    ...
End
```

Y en Rails 1.2:

```
# new.rhtml
    for(:comment, :url => comments_url) do |f|
    ...
End

# edit.rhtml
    form_for(:comment, :url => comment_url(@comment),
      :html => { :method => :put }) do |f|
      ...
End
```

Hemos de fijarnos en que utilizamos la misma declaración tanto en edit como en el new. Esto es debido a que Rails puede deducir que hacer, basado en el nombre de la clase del modelo de instancia @comment.

4.3 HTTP Basic Authentication

Rails 2.0 tiene un nuevo módulo para trabajar con with HTTP Basic Authentication, que resulta un gran modo de hacer la autenticación API sobre SSL. Es muy fácil de usar. Por ejemplo:

```
class PostsController < ApplicationController
    USER_NAME, PASSWORD = "dhh", "secret"

     before_filter :authenticate, :except => [ :index ]

    def index
      render :text => "Everyone can see me!"
    end

    def edit
      render :text => "I'm only accessible if you know the password"
    end

    prívate

    def authenticate
      authenticate_or_request_with_http_basic do |user_name, password|
        user_name == USER_NAME && password == PASSWORD
      end
    end

end
```

Se ha hecho mucho más fácil para estructurar su Javascript y hojas de estilo en unidades lógicas sin que se destructure el http en miles de archivos. Usando javascript_include_tag(:all, :cache => true) hacemos que public/javascripts/.js se convierte en un archivo public/javascripts/all.js en producción, guardando los archivos separados en el desarrollo, para poder trabajar interativamente sin la necesidad de borrar la caché.

4.4 Query Cache

La idea es simple: mientras estas procesando una petición, puedes realizar la misma consulta SQL más de una vez. A veces tenemos condiciones más complejas y no resulta tan obvio en como trabaja la caché. A continuación explicaremos con más detalle que hace, en estos casos, la Query Cache.

```
#app/controllers/posts_controller.rb
    class PostsController < ApplicationController

    def index
      @posts = Post.find(:all)
      @posts2 = Post.find(:all)
      ...
    end
    ...
End
```

Cuando hacemos referencia a http://localhost:3000/posts y miramos log/development.log, esto es lo que veremos:

```
Parameters: {"action"=>"index", "controller"=>"posts"}
Post Load (0.000357) SELECT * FROM `posts`
CACHE (0.000000) SELECT * FROM `posts`
```

Lo primero que realiza a la base de datos es una sentencia, pero lo segundo, al ser idéntico, no vuelve a lanzarlo, si no que consigue los resultados de la caché interna.

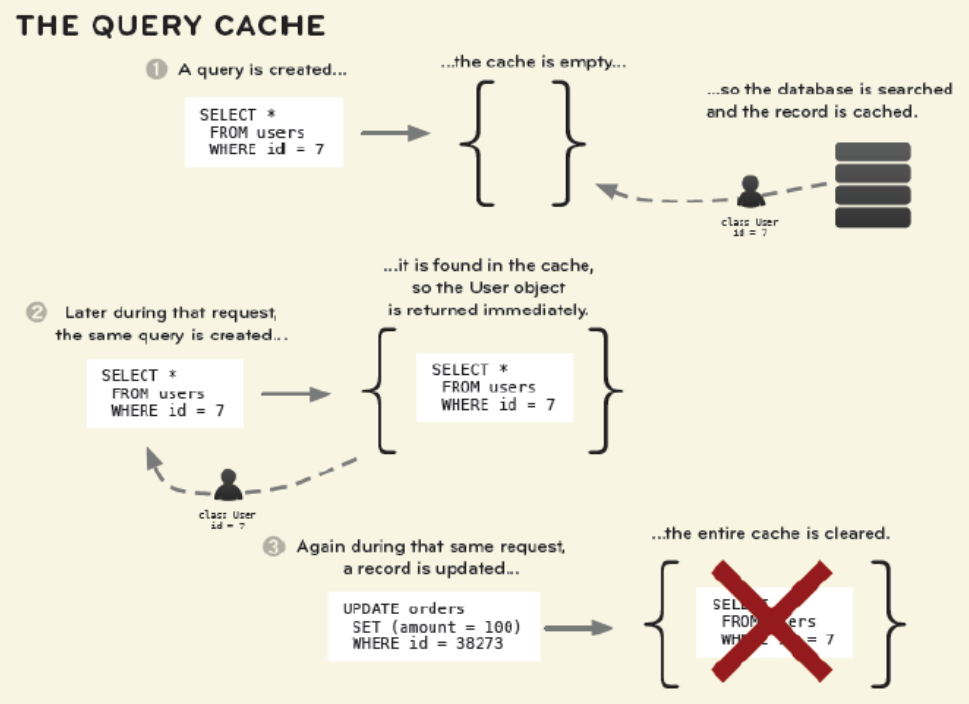


Fig 11. Esquema cómo funciona la Query Cache

Como podemos observar en la figura anterior los pasos son los siguientes:

1. Se hace una consulta en la Base de Datos, como podemos comprobar la caché se crea en ese momento y esta vacía, mientras que la base de datos busca el registro y lo almacena en la caché.
2. Mientras dura la petición, la consulta se crea en la base de datos, y si se hace la misma consulta, se devuelve inmediatamente el resultado.
3. La caché se limpia una vez se haya hecho un update.

4.5 Comandos adicionales para la base de datos

Uno de los comandos más útiles que proporciona Rails es *rake db:migrate*. Con Rails 2, sin embargo, tiene otros comandos adicionales que dan soporte a la base de datos, como por ejemplo *create* y *drop*.

o **DB:CREATE**

El comando *rake db:create* crea la base de datos específica en *config/databases.yml* para RAILS_ENV. Aplicar *rake db:create* en un entorno que no sea de desarrollo es fácil si se especifica en la línea de comando RAILS_ENV.

· rake db:create RAILS_ENV=test

Este comando no se sobrescribe si ya existe en la base de datos.

o **DB:CREATE ALL**

También existe un modo rápido para crear toda la base de datos en el archivo *databases.yml*, sólo es necesario poner *rake db:create:all*. Esto altera a la definición de la base de datos y la crea para cada definición, si esta en local. Eso es, solamente si el valor del host esta en blanco o es localhost, entonces la base de datos podrá ser creada. Con esto por lo menos te aseguras que no se hace ninguna catástrofe en ninguna base de datos remota.

o **DB:DROP and DB:DROP:ALL**

Los comandos *rake db:drop* y *rake db:drop:all* trabajan sobre todo en como borrar la base de datos para RAILS_ENV y borrar toda la base de datos local, respectivamente.

o **DB:RESET**

El comando *rake db:reset* es un camino rápido para resetear la base de datos. Esto borra la borra, la crea y hace la migración. Esto asegura más claridad y para asegurarse que las migraciones se ejecutarán en un nuevo entorno. Este comando sólo se ejecuta en bases de datos que exista RAILS_ENV y coge la misma VERSIÓN como *db:migrate* que le puedes especificar la versión que quieres que sea migrada.

· rake db:reset RAILS_ENV=test VERSION=23

o **DB:ROLLBACK**

Retroceder una version es bastante común ocurre cuando se hace un gran desarrollo y es cuando normalmente se retrocede una versión. Para poder realizar esto, podemos utilizar el comando rollback con un argumento opcional STEP que sirve para especificar cuantas versiones quieres retroceder (por defecto retrocede una)

· rake db:rollback STEP=2

o **DB:VERSION**

Con esta instrucción sabremos en que versión nos encontramos, y no hará falta ver el archivo schema_info.

· rake db:version
Current version: 53

4.6 Eliminación de la paginación

El plugin de paginación se ha eliminado ya que resultaba lento. Ahora podemos usar el plugin *will_paginate* que usa una sintaxis diferente que resulta mejor que el anterior plugin.

Para utilizarlo, sólo se ha de poner en el controlador la llamada al método *paginate*:

· @articles = Post.paginate(:page=>params[:page])

También podemos filtrar dinámicamente con condiciones.

Por ejemplo para que sólo pagine los libros donde el autor sea ‘Jim’, utilizaríamos la siguiente sintaxis.

· @author = Author.find_by_name 'jim'

· @articles = Post.paginate_by_author_id(@author.id, :page => params[:page])

En la vista, llamamos al helper:

· <%= will_paginate @articles %>

4.7 Plugin acts_as

Algunos de los plugins acts_as son los siguientes:

• acts_as_versioned: Almacena una copia de archivos de la base de datos cuando estos son editados.

• acts_as_paranoid: Ningún archivo es eliminado permanentemente.

• acts_as_state_machine: Ejecuta métodos de rellamada cuando el estado cambia.

4.8 Conclusiones

Rails 2.0 tiene buenas mejoras, y la mayoría de ellas no hace falta que se vuelva a aprender cómo funcionan.

La mayoría de las mejoras se centran en ActionPack, el paquete que incluye a ActionController (incluyendo el sistema de rutas) y ActionView (los templates). El ruteo basado en REST se consolida y algunas cosas se reestructuran. ActiveResource, el equivalente a ActiveRecord para trabajar con API's REST, es ahora parte del framework mientras que cosas como la paginación y los adaptadores de bases de datos comerciales salen y se transforman en gemas o plugins opcionales.

Los cambios son varios, pero brillan ahí donde dan énfasis a la filosofía de Rails de "Convención sobre Configuración", en un intento continuo por detectar patrones de uso por parte de los desarrolladores que pueden abstraerse en *macros* o métodos que hagan a esas tareas comunes lo más intuitivas posible. Un buen ejemplo es el nuevo manejo de

excepciones en los controladores. Hasta ahora quedaba en manos del desarrollador qué hacer con excepciones como ActiveRecord::RecordNotFound, desde interceptarlas en cada acción hasta capturarlas en un sólo punto con rescue_action_in_public. Ahora, ActionController provee una macro para definir métodos de controlador para hacerse cargo de cada excepción.

Otra novedad es la macro para habilitar autentificación HTTP en los controladores. Esto facilita las cosas considerablemente para implementar API's REST protegidas.

El conjunto de mejoras aumenta el rendimiento y velocidad del framework, especialmente en el sistema de rutas y templates que hacían de ActionPack el componente más lento del conjunto.

Por ahora, para pasar las aplicaciones existentes a 2.0 el equipo de Rails recomienda que primero instales la versión 1.2.3.

Las nuevas características que incorpora caen dentro de la "evolución", más que dentro de la "revolución".

4.9 Actualizar versión de Rails

Para poder utilizar Rails 2.0 se ha de hacer lo siguiente:

· gem install rails -y --source http://gems.rubyonrails.org

5. Implementaciones de Ruby

Una de las críticas más habituales que se hace a Ruby es el pobre rendimiento del intérprete Ruby frente a otros lenguajes rivales. Las cifras indican que por lo general para un mismo algoritmo el intérprete de Ruby ofrece un menor rendimiento que Python, PHP, Perl, etc.

La respuesta comunmente aceptada para esta penalización de rendimiento suele ser que Matz es un gran diseñador de lenguajes pero su labor como implementador no es tan brillante. El intérprete de Ruby escrito por Matz, conocido como MRI (de *Matz Reference Implementation*, o Implementación de Referencia de Matz) adolece de no pocos problemas.

Hace tiempo que se ha asumido el problema de rendimiento en la MRI y han surgido no pocos proyectos para corregir esta deficiencia. A diferencia de esfuerzos recientes como Perl 6, el lenguaje Ruby no dispone de una especificación funcional. Por tanto, la única fuente fiable de documentación sobre los entresijos del lenguaje Ruby es el estudio del código fuente en C del intérprete escrito por Matz.

5.1 JRuby

5.1.1 ¿Qué es JRuby?

JRuby es una implementación Java 100% pura del lenguaje de programación Ruby.

5.1.2 Características

· Intérprete Ruby compatible con 1.8.5 escrito 100% en Java.

· Incluidas la mayoría de las clases built-in de Ruby.

· Soporte para la interacción con clases Java y la definición de las mismas dentro de Ruby.

· Distribuido bajo un combinado trilicencia (CPL/GPL/LGPL).

· Integración sencilla con código Java.

· Ruby puede llamar a Java, Java puede llamar a Ruby.

· Escalabilidad con la JVM.

· Threading nativo.

· Compilación a bytecode Java (y jruby -c, o jrubyc)

· Uso de los activos enterprise Java.

· Librerías Java, Herramientas...

5.1.3 Diferencias entre JRuby y Ruby

Normalmente, conocemos al intérprete de Ruby tradicional como cRuby**.**

- Ruby = Intérprete + Lenguaje + (App)
- JRuby = Intérprete + Java Class Libraries

5.2 YARV

YARV de Sasada Koichi, la reimplementación de Ruby usando su propia máquina virtual y un compilador a código de bytes. El trabajo de Sasada Koichi, basado en un enfoque más moderno que la implementación de Matz, es tan brillante que la versión oficial 1.9.1 de Ruby estará basada en YARV. Aquí hay una serie de gráficos que muestran las mejoras de rendimiento de YARV frente a la MRI. Como puede verse, aún hay algún trabajo por hacer pero cabe destacar que, por lo general, una aplicación Rails verá mejoras de rendimiento del 15% sólo por usar YARV.

Como se ha visto, YARV divide el problema de la implementación del lenguaje en dos etapas: un compilador de código de bytes y una máquina virtual que ejecuta dicho bytecode. Surge de manera natural la pregunta de si no sería posible usar alguna máquina virtual ya existente y generar bytecode compatible con ella. Y, hablando de máquinas virtuales, la de Java está más que probada.

5.3 Rubinius

El enfoque de esta iniciativa consiste en escribir la mayor parte del lenguaje en Ruby y dejar que la máquina virtual -escrita en C- ejecute el menor subconjunto posible de funcionalidad. En concreto, Rubinius pone el énfasis en el recolector de memoria y la implementación de forks y threads. A destacar que Rubinius parece surgir del mundo Rails (Evan Phoenix trabaja en Engine Yard, proveedores de alojamiento avanzado Rails) por tanto buena parte de las mejoras que promete en el intérprete atacan a los problemas de rendimiento que más pueden aquejar a nuestras aplicaciones Rails.

5.4 Ruby .NET

Ruby.NET: Se trata de un proyecto de código abierto impulsado por Wayne Kelly y John Cough de la Queensland University of Technology (en Australia) y parcialmente financiado por Microsoft. Sin embargo, según los propios autores, la tarea de soportar RoR aún no está lista para su lanzamiento público.

IronRuby es la segunda implementación de Ruby sobre .NET más interesante quizá porque surge de un desarrollador de Microsoft, John Lam. Lam se ha basado en el Dynamic Language Runtime , originalmente pensado para Python y LISP. Parece ser que el desarrollo de IronRuby ha sido lo suficientemente complejo como para justificar cambios en el propio DLR oficial de Microsoft para acomodar a Ruby.

Si bien los esfuerzos en .NET como vemos parecen estar más verdes que sus equivalentes en Java, conviene tener en cuenta -sobre todo si desarrollamos sobre Windows- a estos dos proyectos.

6. Última actualización de RoR

6.1 Ruby on Rails 2.1

Las principales novedades son las siguientes:

• **Gem dependencies:** Los plugins de Rails son muy útiles porque, al estar incluidos en la aplicación, pueden ofrecen funcionalidades extra sin dependencias externas. Sin embargo, no sucede lo mismo con las gemas, en las que hasta hace poco, no había manera builtin de definir estas dependencias externas mediante programación. Todo esto cambiará con la nueva versión.

• **Dirty tracking con actualizaciones parciales:** Saber si tus objetos de Active Record han sido modificados o no es ahora mucho más sencillo.

• **has_finder toma la forma de named_scope.** El popular plugin has_finder tomará la forma de named_scope en Rails 2.1.

• **Soporte de zona horaria incorporado.** Ya no será necesario forzar la zona horaria con hasta dos plugins.

• **Mejor infraestructura de cacheo.** A partir de ahora, podrás especificar tu motor de cacheo preferido en el fichero de configuración.

2. Búsqueda en internet de información herramientas

Tarea en la cual buscamos las herramientas software necesarias para la realización del proyecto, a continuación mostramos y hacemos una pequeña descripción de cada una de ellas

Herramientas y lenguajes utilizados

- **RubyStack**: BitNami RubyStack simplifica enormemente el desarrollo y despliegue de Ruby on Rails. Incluye listas para ejecutar las versiones de Apache, MySQL, Ruby y Rails y las dependencias necesarias. Se puede implementar utilizando un instalador nativo, como una máquina virtual o en la nube. BitNami RubyStack se distribuye de forma gratuita bajo la licencia Apache2.0.

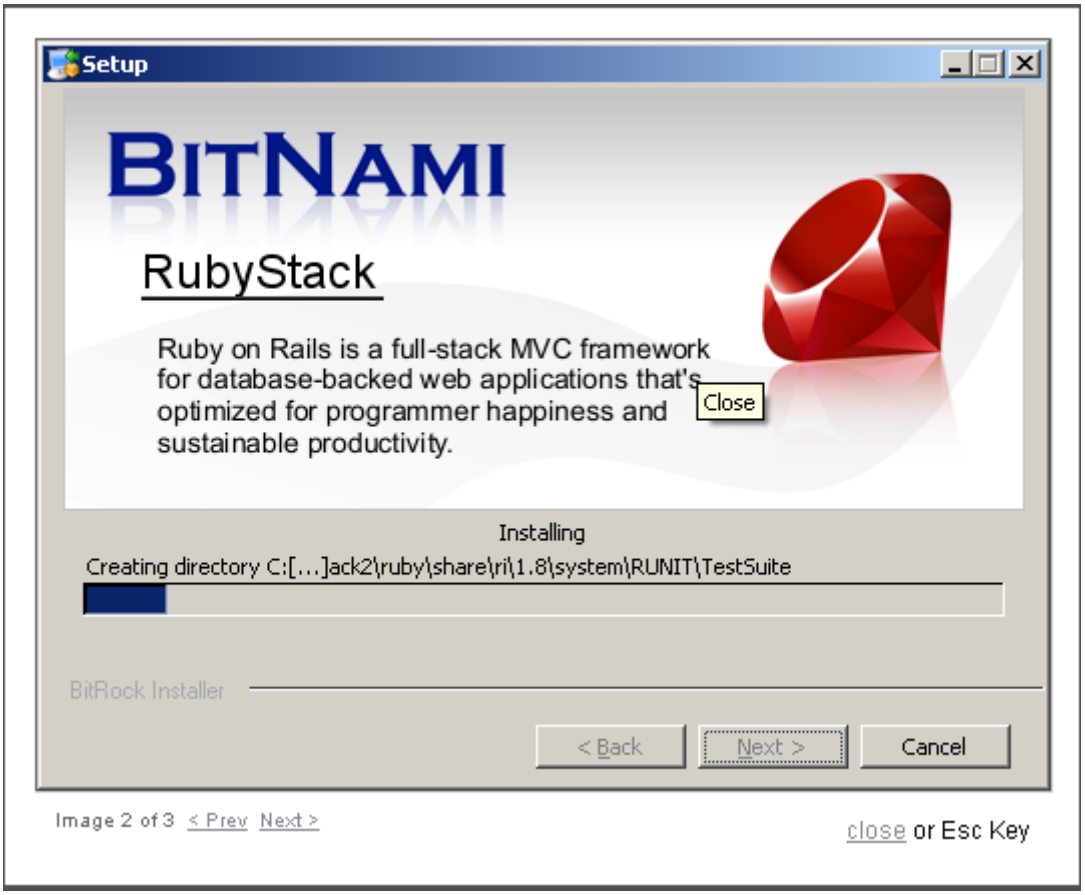


Fig 12. RubyStack

- **Ruby**: es un lenguaje de programación reflexivo y orientado a objetos.

- **Ruby on Rails**: es un framework de aplicaciones web de código abierto escrito en el lenguaje de programación Ruby. Siguiendo el paradigma de la arquitectura Modelo Vista Controlador (MVC). El lenguaje de programación Ruby permite la metaprogramación, de la cual Rails hace uso, lo que resulta en una sintaxis que muchos de sus usuarios encuentran muy legible.

- **Git Bash:** Git es un software libre y abierto, sistema distribuido de control de versiones diseñado para manejar todo, desde pequeños proyectos a muy grandes con velocidad y eficiencia.

  Git se distribuye por la versión del sistema de control concentrado en la velocidad, efectividad y facilidad de uso del mundo real en proyectos grandes. Sus características más destacadas son:

  - El desarrollo distribuido. Como la mayoría de otros sistemas modernos de control de versiones, Git le da a cada desarrollador una copia local de toda la historia del desarrollo y los cambios se copian de un repositorio de este tipo a otra. Estos cambios se importan como ramas de desarrollo adicional, y se pueden combinar en la misma forma que una rama de desarrollo local. A los repositorios se puede acceder fácilmente a través del eficiente protocolo Git (opcionalmente envuelto en ssh para la autenticación y la seguridad) o simplemente utilizando HTTP - puede publicar su repositorio en cualquier lugar sin ningún tipo de configuración del servidor web especial.

  - Fuerte apoyo para el desarrollo no lineal. Git apoya rápida y convenientemente la ramificación y fusión, e incluye potentes herramientas de visualización y navegación de una historia del desarrollo no lineal.

  - Manejo eficiente de grandes proyectos. Git es muy rápido y escalable, incluso cuando se trabaja con grandes proyectos y una larga historia. Es comúnmente un orden de magnitud más rápido que la mayoría de otros sistemas de control de versiones, y en varios órdenes de magnitud más rápido en algunas operaciones. También utiliza un formato extremadamente eficiente embalado para su almacenamiento a largo plazo de revisión que actualmente encabeza cualquier otra versión de código abierto al sistema de control.

  - Autenticación criptográfica de la historia. La historia Git se almacena de tal manera que el nombre de una revisión en particular (un "compromiso" en términos Git) depende de la historia de desarrollo completo que condujeron a ese cometido. Una vez que se publica, no

es posible cambiar las versiones antiguas sin que se note. También, las etiquetas pueden ser firmadas criptográficamente.

- Kit de herramientas de diseño. Siguiendo la tradición Unix, Git es un conjunto de muchas pequeñas herramientas escrito en C, y una serie de secuencias de comandos que proporcionan envoltorios convenientemente. Git proporciona herramientas de fácil uso para humanos así como secuencias de comandos fáciles para llevar a cabo nuevas operaciones inteligentes.

Fig 13: Git

- **Tog** es una nueva **plataforma para la creación de todo tipo de comunidades 2.0**, desde blogs hasta completas redes sociales, desarrollado íntegramente en España.

  Se trata de una plataforma en Ruby&Rails de código abierto, bajo licencia MIT, completamente extensible, ya que con la incorporación de plugins, los portales creados con **Tog** adquirirán todo tipo de funcionalidades sociales según vayan creciendo el número de plugins disponibles.

Fig 14: TOG

- **Microsoft Word 2007**: es el procesador de texto de la suite de Microsoft Office 2007.

- **Rad Rails:** es un desarrollo rápido de aplicaciones para el IDE de Ruby on Rails. El objetivo de RadRails es proporcionar a los desarrolladores de Rails todo lo necesario para desarrollar, administrar, probar y desplegar sus aplicaciones. Entre sus características se incluyen control de código fuente, asistente de código, la refactorización, la depuración, servidores WEBrick, asistentes generador, resaltado de sintaxis, las herramientas de datos, y mucho más.

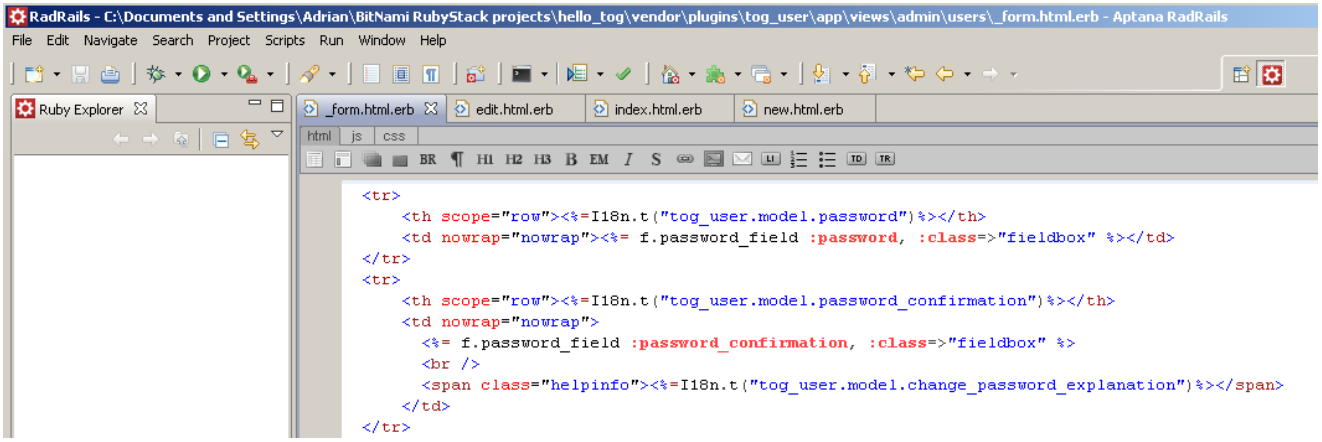


Fig 13: RadRails

3. Instalación del hardware y software necesario para el PFC

En esta tarea describiremos los pasos seguidos para la instalación del software explicado anteriormente, estos son:

1) Instalamos Bitnami Rubystack: http://bitnami.org/files/stacks/rubystack/bitnami-rubystack-1.6-0-windows-installer.exe. En el menú entrante seleccionamos "Use bitnnami ruby stack" y actualizamos las gemas en la consola con "*gem update*".

2) Acto seguido instalamos Git: http://msysgit.googlecode.com/files/Git-1.6.5.1preview20091022.exe (con la opción: "*Run git with the windows command prompt*")

Esto prepararía todo lo que necesitamos para usar rails e instalar Tog.

3) Posteriormente comenzamos la instalación de la plataforma Tog para la cual son necesarios los siguientes requerimientos:

- **Git** ya que muchos plugins son descargados desde los repositorios Github.
- **Gem source** para instalar las gemas necesarias que se encuentran en Github, para ello introduciremos el siguiente comando en la consola "gem sources -a http://gems.github.com"
- **ImageMagick** el cual es necesario para el procesamiento de imágenes porque Tog usa un portapeles. Provee a las redes sociales de características realmente necesarias.

Una vez instalados los requerimientos pasamos a crear nuestra aplicación web rails usando Tog introduciendo el siguiente comando en la consola: "*rails hello_tog -m http://tr.im/tog_current*" . Esto creará la aplicación con tog y todas las gemas y plugins necesarios instalados. Y por último generamos la plantilla con la base de datos y su esquema de tablas mediante el comando tog "*rake rails:template LOCATION=http://tr.im/tog_current*".

Luego en tres sencillos pasos tenemos instalado una pequeña aplicación web con características similares a una red social.

4. Generación de documentación sobre herramientas

Esta tarea consiste en generar la documentación necesaria sobre las herramientas utilizadas en el proyecto. Toda esa información se encuentra en las 3 anteriores tareas.

Actividad 1.2: Identificación del ámbito del sistema.

Actividad en la que estudiaremos la funcionalidad del sistema y definiremos los objetivos y fines de la red social.

1. Búsqueda de información sobre la funcionalidad del sistema y estudio del mismo

En esta tarea fundamentalmente hemos llevado a cabo un estudio sobre el código inicial de la plataforma Tog, es decir realizamos una investigación exhaustiva de los diferentes plugins instalados así como de otros que pudieran sernos de utilidad.

Entre estos plugins son:

- tog_core → sus funcionalidades son las siguientes: controlador de acciones del administrador y miembros de la red social. Cuadro de mando de control, declaración de características de la red social. Control de abuso y observaciones de la administración. Gestión de la configuración y sistema de búsqueda.
- tog_social → sus funcionalidades son las siguientes: crear y configurar perfiles, moderar y crear grupos, invitaciones para unirse a un grupo y seguir a un amigo
- tog_mail → sus funcionalidades son las siguientes: múltiples carpetas, enviar mensajes a tu lista de amigos o a un usuario determinado.
- Otros plugins: tog_user (gestión del sistema a través de la autentificación en la rede social), tog_conversatio (un sistema de blogs simple), tog_picto (gestionar y compartir fotos entre los usuarios de la red social.), tog_headlines (gestión de noticias, comunicados de prensa, etc). Así como otros de menor importancia.

2. Definición de objetivos y fines del sistema

En esta tarea fundamentalmente definimos conjuntamente con los tutores del proyecto los objetivos de mismo así como la finalidad de la red social a construir.

El objetivo fundamental del proyecto es el desarrollo de una red social para PyMes.

Otros objetivos a conseguir con la realización de este proyecto son:

- La realización de una página intuitiva, moderna, ágil, dinámica y funcional.

- Adaptar la red social a construir de manera que forme parte de la nueva generación de aplicaciones Web 2.0
- Dotar a la aplicación de una gran reusabilidad, de forma que se puedan implementar mejoras y cambios con relativa facilidad y sin afectar al diseño original.

Dentro de la finalidad de la aplicación se definen las siguientes tareas:

a) Promover las PyMes de Gran Canaria
b) Posibilidad de creación de cuentas tanto para PyMes como usuarios anónimos.
c) Gestión y control del número de Pymes y del resto de usuarios, así como de las cuentas y uso de las mismas.
d) Encontrar oportunidades de negocios y socios potenciales
e) Publicar ofertas de trabajo y encontrar a los mejores talentos para cada PyMe.
f) Chat y Foro para hacer que la comunicación sea mejor.
g) Mantenimiento actualizado de la red social.

Actividad 1.3: Identificación del ámbito del sistema.

Actividad donde definiremos los requerimientos de la aplicación, así como los casos de uso con sus correspondientes diagramas de flujo.

1. Análisis de requerimientos

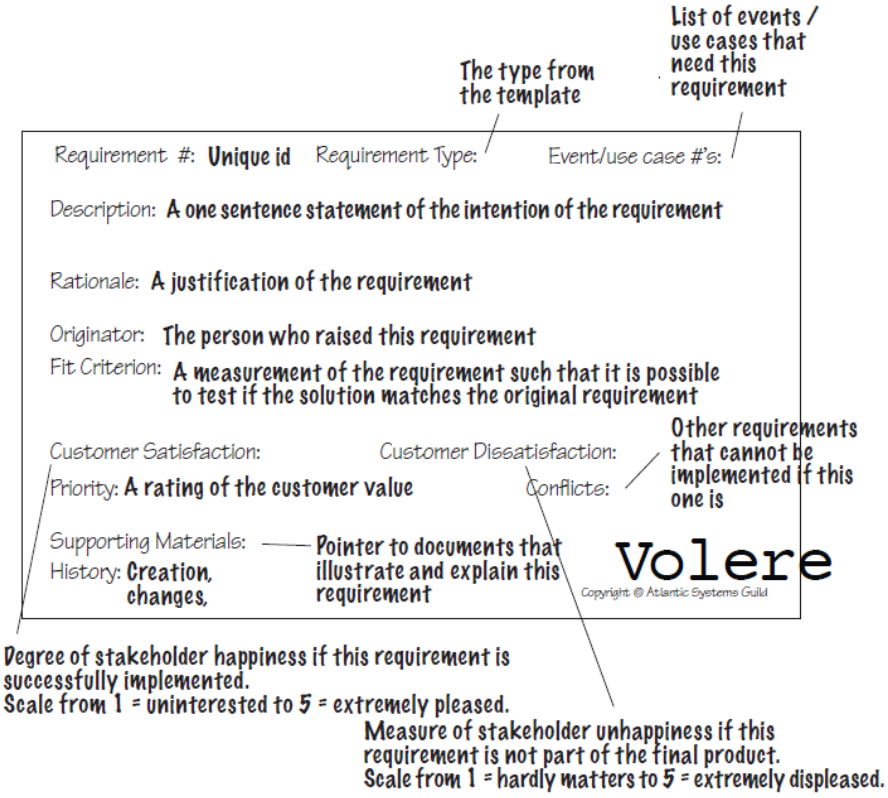
La fase de análisis permite identificar y definir formalmente los requerimientos de la aplicación antes de de pasar a la fases de diseño e implementación. Estos requerimientos según Pressman son la descripción de los servicios proporcionados por el sistema que se pretende desarrollar y sus restricciones operativas (Pressman, 2002).

Para obtener los requerimientos del sistema es necesario un proceso, en el que se parte de una descripción general dada por el cliente (restricciones del proyecto) a la que

se van añadiendo detalles hasta generar una documentación lo suficientemente precisa para abordar el diseño de la aplicación.

Para la obtener los requerimientos del sistema se ha optado por basar el análisis de requerimientos en las plantillas de casos de uso completos (Full Dressed Use cases) propuestas por Alistair Cockburn en el libro "Writting effective use cases" (Cockburn, 2000). Y en la plantilla Volere (http://www.volere.co.uk) garantizada por el grupo de expertos ingenieros "The atlantic systems guild".

La plantilla de Volere propone que para cada requerimiento detectado se rellene una ficha, que posteriormente será enlazada a otros requerimientos o documentos que lo describen. Esta ficha es como la que se muestra a continuación en inglés:



Nosotros la hemos adaptado de la siguiente manera:

Nº de Requerimiento:	Tipo:	Casos de uso:
Nombre:		
Justificación:		
Origen:		

Criterio de validación:	
Satisfacción del cliente:	Insatisfacción del cliente:
Prioridad:	Conflictos:
Historia:	
Descripción:	

- **Número de requerimiento:** Número que identificará únicamente al requerimiento
- **Tipo:** Tipo de requerimiento de entre los descritos en la plantilla.
- **Casos de uso:** Lista de identificadores de casos de uso que son afectados por este requerimiento.
- **Nombre:** Una descripción breve del requerimiento.
- **Justificación:** Motivaciones y justificación de la creación del requerimiento.
- **Origen:** Persona que originó el requerimiento.
- **Criterio de validación:** Condición o medida del requerimiento que permita verificar que la solución final cumple el requerimiento.
- **Satisfacción del cliente:** Grado de satisfacción del cliente si este requerimiento se implementa satisfactoriamente. La escala va desde 1→Sin interés a 5→Extremadamente complacido.
- **Insatisfacción del cliente:** Grado de insatisfacción del cliente si este requerimiento no es incluido en el producto final. La escala va desde 1 →Sin importancia a 5→Extremadamente insatisfecho.
- **Prioridad:** Una medida del valor del requerimiento.
- **Conflictos:** Lista de requerimientos que no podrán ser implementados si este lo es.
- **Historia:** Historia de creación y modificaciones.
- **Descripción:** Descripción y explicación exhaustiva del requerimiento.

1. Restricciones del proyecto

Las restricciones del proyecto son aquellos requerimientos que normalmente son de obligado cumplimiento y que son dados en el inicio del proyecto por el cliente. Las restricciones pueden describir la forma en la que el problema debe ser resuelto, las tecnologías que serán utilizadas, los sistemas con las que tendrá que interactuar, librerías o software con los que se requiere que sea realizado el proyecto, el entorno en el que será implantado el producto o restricciones de planificación y presupuesto. Además se pueden expresar aquí hechos que puedan afectar al producto.

1.2 Nomenclatura y definiciones

A continuación se detalla una nomenclatura utilizada en el proyecto.

- **Administrador:** persona responsable de la seguridad y funcionamiento de la aplicación.
- **Administración Pública:** persona que dispone de un perfil en la red social, con la posibilidad de participar en eventos y foros.
- **AJAX:** Asyncronous Javascript and XML
- **Blog:** en español bitácora, es un sitio web periódicamente actualizado que recopila cronológicamente textos o artículos de uno o varios autores, apareciendo primero el más reciente, donde el editor conserva siempre la libertad de dejar publicado lo que crea pertinente.
- **Clúster:** concentración de Pymes con los mismos intereses o dedicadas al mismo sector.
- **Foro:** sitio web de la red social para dar soporte a discusiones u opiniones en línea, es decir funciona como un sistema de comunicación entre Pymes.
- **Pyme:** acrónimo de pequeñas y medianas empresas, viene a significar aquellas empresas con características distintivas, y tienen dimensiones con ciertos límites ocupacionales y financieros prefijados por los Estados o Regiones
- **Ruby on rails:** lenguaje de programación interpretado, reflexivo y orientado a objetos.

- **Usuario:** persona que tiene un nombre de usuario y contraseña con la que acceder al sistema.

1.3 Restricciones

Requerimientos que configuran las restricciones iniciales del proyecto.

R01. Empleo de la tecnología Ruby on Rails

Tipo: Restricción de solución		Casos de uso:
Nombre:	Empleo de la tecnología Ruby on Rails	
Justificación:	La necesidad de partir de un código libre que implemente una red social, en este caso a partir de la plataforma TOG. Ruby permite el diseño de aplicaciones web de forma rápida y sencilla.	
Origen:	Jonay Santana	
Criterio de validación:	La aplicación dispone de una interfaz que permita el uso de todas sus características a través de un navegador.	
Satisfacción del cliente: 5		Insatisfacción del cliente: 3
Prioridad: 1		Conflictos:
Historia:	14/06/2010 – Creación	
Descripción:	La aplicación será desarrollada mediante la tecnología Ruby on Rails.	

R02. Empleo de la plataforma TOG

Tipo: Restricción de solución		Casos de uso:
Nombre:	Empleo de la plataforma TOG	
Justificación:	La necesidad de partir de un código libre que implemente una red social, en este caso a partir de la plataforma TOG. Este código previo ha sido desarrollado en la tecnología Ruby on Rails.	

Origen:	Jonay Santana
Criterio de validación:	Se exige partir de un código que implemente algunas características de una red social.
Satisfacción del cliente: 5	Insatisfacción del cliente: 3
Prioridad: 1	Conflictos:
Historia:	14/06/2010 – Creación
Descripción:	La aplicación será desarrollada mediante la tecnología Ruby on Rails partiendo del código desarrollado por la plataforma TOG.

R03. Interfaz Web

Tipo: Restricción de solución	Casos de uso:
Nombre:	Interfaz Web
Justificación:	La aplicación debe poder ejecutarse en red y se desea que no sea dependiente de una instalación.
Origen:	Jonay Santana
Criterio de validación:	La aplicación dispone de una interfaz que permita el uso de todas sus características a través de un navegador.
Satisfacción del cliente: 5	Insatisfacción del cliente: 3
Prioridad: 1	Conflictos:
Historia:	14/06/2010 – Creación
Descripción:	La aplicación tendrá una interfaz web generada por el servidor Mongrel.

R04. Empleo de la tecnología AJAX

Tipo: Restricción de solución	Casos de uso:
Nombre:	Empleo de la tecnología AJAX
Justificación:	AJAX permite la creación de interfaces web interactivas, de esta manera el usuario no estará

	mirando la página en blanco del navegador y un icono del reloj de arena esperando a que el servidor haga algo.
Origen:	Jonay Santana
Criterio de validación:	La interfaz n necesita refrescarse completamente cada vez que el usuario lance una acción.

Satisfacción del cliente: 5	Insatisfacción del cliente: 3
Prioridad: 1	Conflictos:
Historia:	14/06/2010 – Creación
Descripción:	Se utilizará AJAX para el desarrollo de una interfaz cercana al usuario.

R05. Compatibilidad con navegadores

Tipo: Entorno de implementación	Casos de uso:
Nombre:	Compatibilidad con navegadores
Justificación:	Se exige que la aplicación sea compatible con los navegadores más utilizados para asegurar que pueda ser ejecutada sobre cualquier plataforma.
Origen:	Jonay Santana
Criterio de validación:	La aplicación dispone de las mismas funcionalidades en rodos los navegadores soportados.

Satisfacción del cliente: 5	Insatisfacción del cliente: 3
Prioridad: 2	Conflictos:
Historia:	14/06/2010– Creación
Descripción:	La apliación podrá ser ejecutado como mínimo en los siguientes navegadores: - Internet Explorer - Mozilla Firefox - Opera - Safari

R06. Portabilidad

Tipo: Entorno de implementación		Casos de uso:
Nombre:	Portabilidad	
Justificación:	Se exige que la aplicación sea ejecutable con independencia del sistema operativo.	
Origen:	Jonay Santana	
Criterio de validación:	La aplicación dispone de las mismas funcionalidades en todos los sistemas operativos soportados.	
Satisfacción del cliente: 5	Insatisfacción del cliente: 3	
Prioridad: 2	Conflictos:	
Historia:	14/06/2010– Creación	
Descripción:	La aplicación debe poderse ejecutar virtualmente en cualquier sistema operativo que pueda ejecutar alguno de los navegadores soportados, pero como mínimo debe garantizar su funcionamiento en los sistemas operativos siguientes: - Windows 98* o superior - Linux 2.2.14* o superior - Mac OS X 10.2.0* o superior (*) Las versiones de los Sistemas operativos se corresponden a las versiones en que alguno de los navegadores pueden ser soportados.	

2. Requerimientos Funcionales

La técnica utilizada para la captura de los requerimientos funcionales del sistema ha sido la obtención de los casos de uso completos. Siguiendo las directrices de Cockburn (Cockburn, 2000) el proceso de obtención de los casos de uso se ha seguido los siguientes pasos:

1. Identificación de los actores: Los actores son entes, humanos o no, que interactúan de alguna forma con el sistema.

2. Listar sus objetivos: Identificar cuáles son los casos de cada actor con respecto al sistema y los objetivos que el actor desea cumplir mediante el uso del sistema.
3. Creación de un listado de los casos de uso según los objetivos de los usuarios, de manera que representen los requisitos funcionales del sistema.
4. Descubrir el escenario de éxito para cada caso de uso: el escenario de éxito es aquel en el que el actor principal consiga su objetivo respetando los intereses de los otros implicados.
5. Identificar excepciones o extensiones: Identificar flujos de ejecución alternativos para cada caso de uso, por ejemplo, condiciones de fallo o formas alternativas de cumplir el objetivo del actor principal.
6. Validar casos de uso: Es conveniente validar los casos de uso en todo momento asegurando que la operación descrita en ellos es la que se espera realmente del sistema.
7. Optimizar casos de uso: Buscar la forma de optimizar los procesos descritos en los casos de uso.

   Los resultados del análisis de requerimientos funcionales se muestran en los siguientes apartados.

2.1 Lista de Actores – Objetivos

La lista de Actores-Objetivos es una primera aproximación al contenido funcional del sistema. Sirve tanto para identificar los actores principales del sistema como especificar cuáles serán los servicios de nivel de usuario que serán soportados por la aplicación. La lista consiste en una tabla de tres columnas. En la primera se especifica el actor en cuestión, la segunda especifica los objetivos del mismo y la tercera es la prioridad del objetivo.

Actor	**Objetivo**	**Prioridad**
Usuario	Manejo de sesiones	1
Administrador de la aplicación	Administrar la red social	4
	Administrar el acceso de usuarios, pymes y	4

	administraciones a la red social.	
Administrador de la Pyme/Administrador de la adm. Pública	Administrar el perfil creado de la Pyme en la red social (subir fotos de productos, crear eventos, etc). También existe el rol de administrador de las administraciones públicas, teniendo exactamente el mismo objetivo que 4un administrador de la Pyme.	4
Administrador del clúster de PyMes	Crear y administrar el clúster de empresas asociadas a un mismo sector o con los mismos intereses.	3
Editor de blog	Persona encargada de administrar el blog, coordinar, borrar, o reescribir los artículos, moderar los comentarios de los lectores, etc.	2

Los roles identificados se solapan entre sí, siendo derivados unos de otros como se muestra en el diagrama siguiente, de forma que todos los roles pueden manejar su sesión.

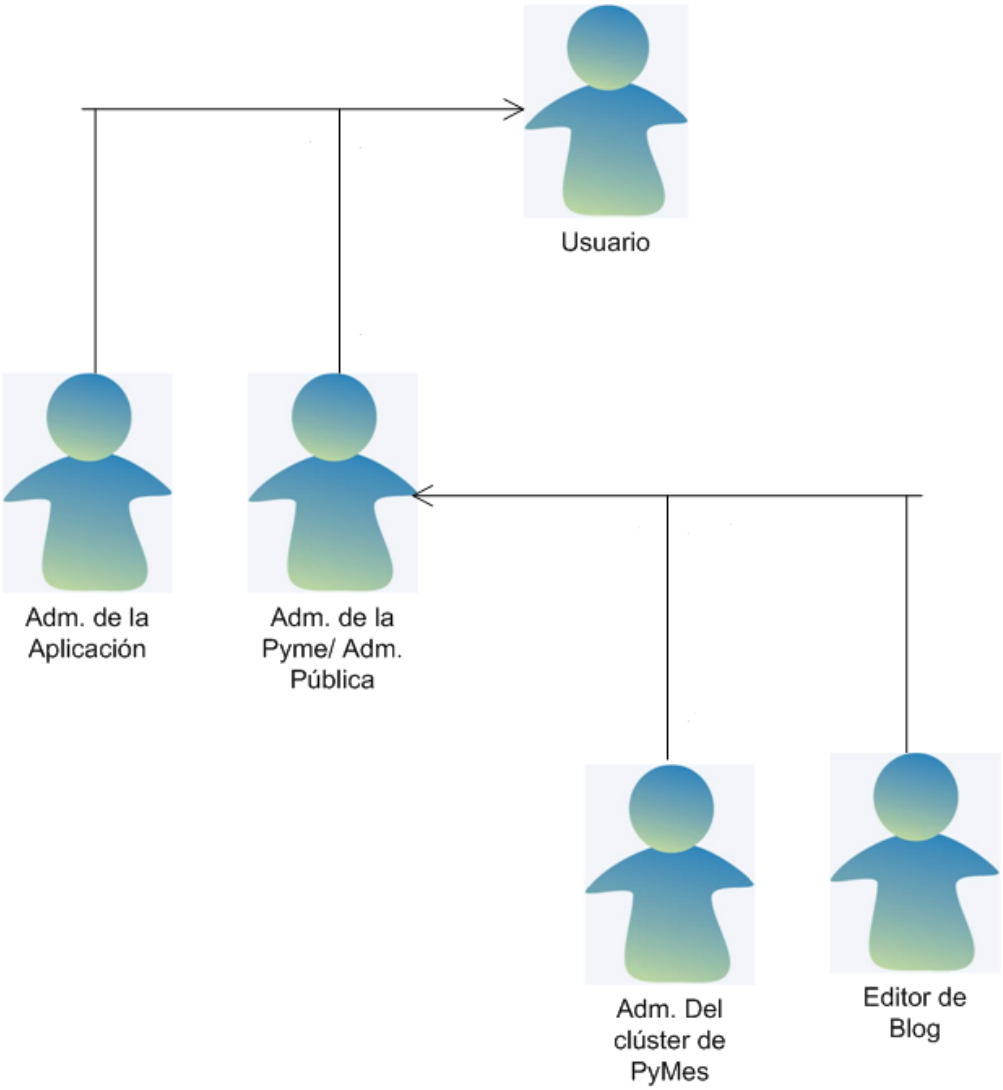


Diagrama 0 – Roles de usuario de la aplicación

2.2 Lista de casos de uso

Procederemos a identificar los casos de uso, mostrando un listado completo de los mismos, ordenándolos por actor.

Administrador de la aplicación	Login/Iniciar Sesión UC 002
	Finalizar Sesión UC 006
	Administrar usuarios
Administrador de la	Registrarse/Darse de alta UC 001

Pyme/Administrador de la adm. Pública	Login/Iniciar Sesión UC 002
	Finalizar Sesión UC 006
	Borrar Cuenta UC 004
	Actualizar Perfil UC 005
	Cambiar Contraseña UC 003
	Crear Clúster UC 018
	Unirse a un clúster UC 019
	Abandonar clúster 020
	Ver clúster UC 021
	Crear álbum de fotos UC 024
	Eliminar álbum de fotos UC 025
	Subir foto UC 026
	Eliminar foto UC 027
	Crear blog UC 028
	Participar en el foro UC 038
	Crear evento UC 037
	Ver evento UC 038
	Publicar oferta/demanda de productos y servicios UC 040
	Ver oferta/demanda de productos y servicios UC 042
	Eliminar oferta/demanda de productos y servicios UC 043
	Buscar en la aplicación UC 007
	Enviar mensaje privado UC 008
	Enviar mensaje UC 013
	Eliminar mensaje UC 014
	Contestar mensaje UC 015
	Ver mensajes recibidos UC 016
	Ver mensajes enviados UC 017
	Enviar comentario UC 043
	Eliminar comentario UC 044

Usuario	Registrarse UC 001
	Login/Iniciar Sesión UC 002
	Finalizar Sesión UC 006
	Borrar Cuenta UC 004
	Actualizar Perfil UC 005
	Cambiar Contraseña UC 003
	Seguir a Pyme UC 010
	Dejar de seguir a una Pyme UC 011
	Buscar en la aplicación UC 007
	Enviar mensaje privado UC 008
	Enviar mensaje UC 013
	Eliminar mensaje UC 014
	Contestar mensaje UC 015
	Ver mensajes recibidos UC 016
	Ver mensajes enviados UC 017
	Enviar comentario UC 043
	Eliminar comentario UC 044
Administrador del clúster de PyMes	Crear clúster UC 018
	Editar clúster UC 022
Editor de blog	Ver/Gestionar editores UC 032
	Crear blog UC 028
	Eliminar blog UC 029
	Añadir editor UC 030
	Eliminar editor UC 031
	Crear post UC 033
	Añadir post UC 034
	Ver post UC 035
	Ver posts UC 036

2.3 Diagramas de casos de uso

Los diagramas de casos de uso proporcionan una visión rápida y global de las relaciones entre los casos de uso. En el presente apartado se muestran los diagramas clasificados por roles.

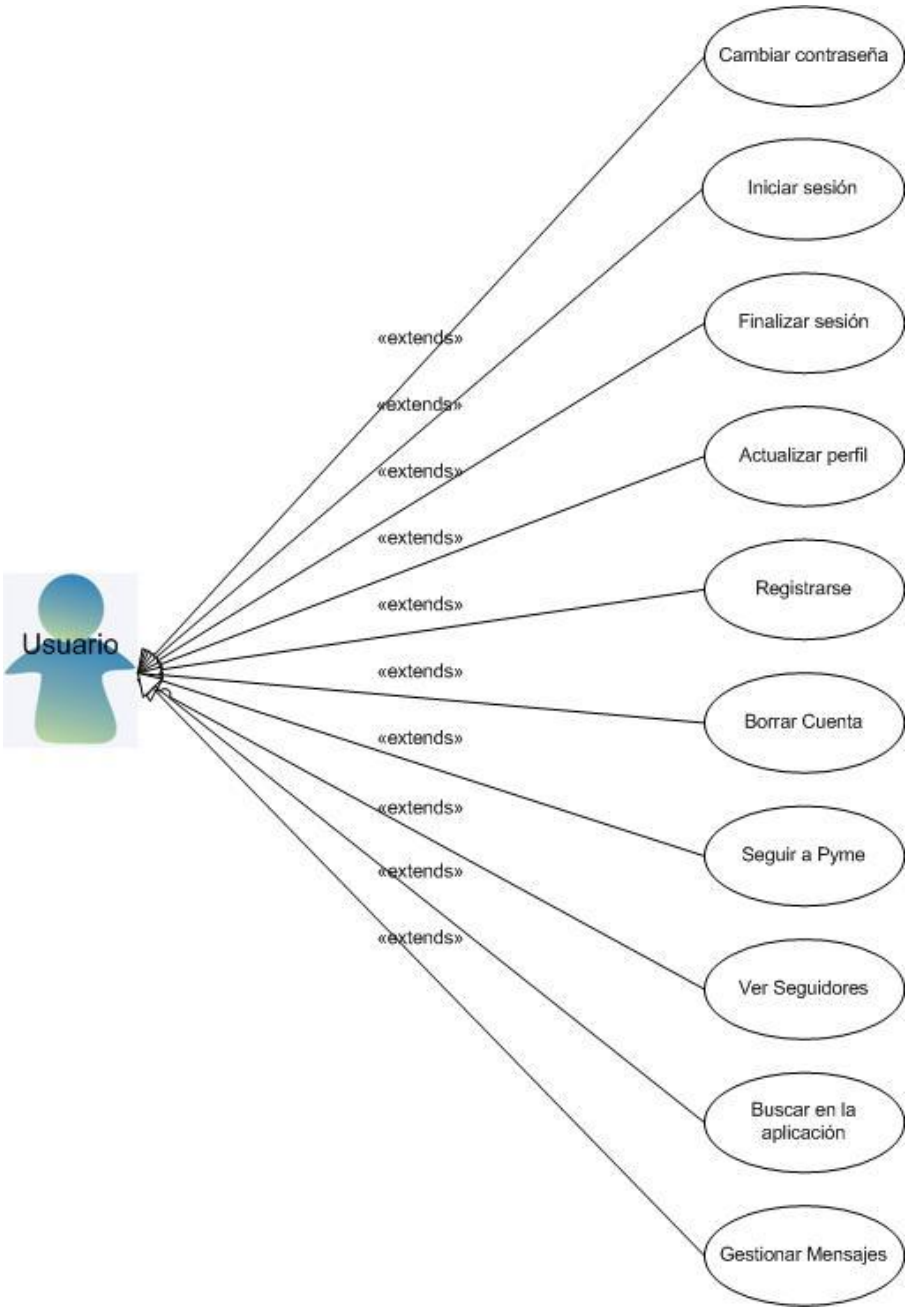


Diagrama 1 – Objetivos de Usuario

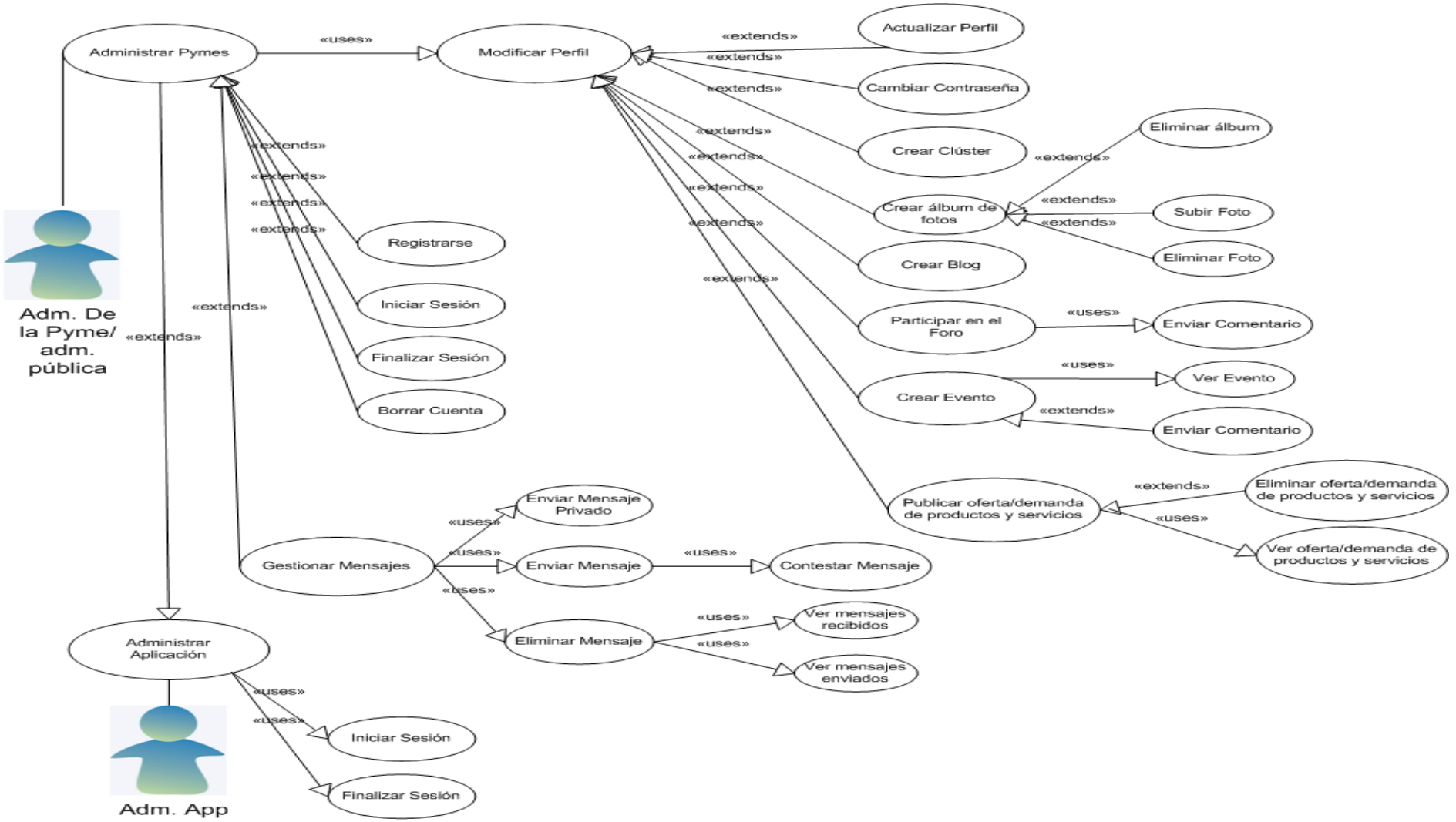


Diagrama 3 –Objetivos del administrador de la aplicación y de las Pymes

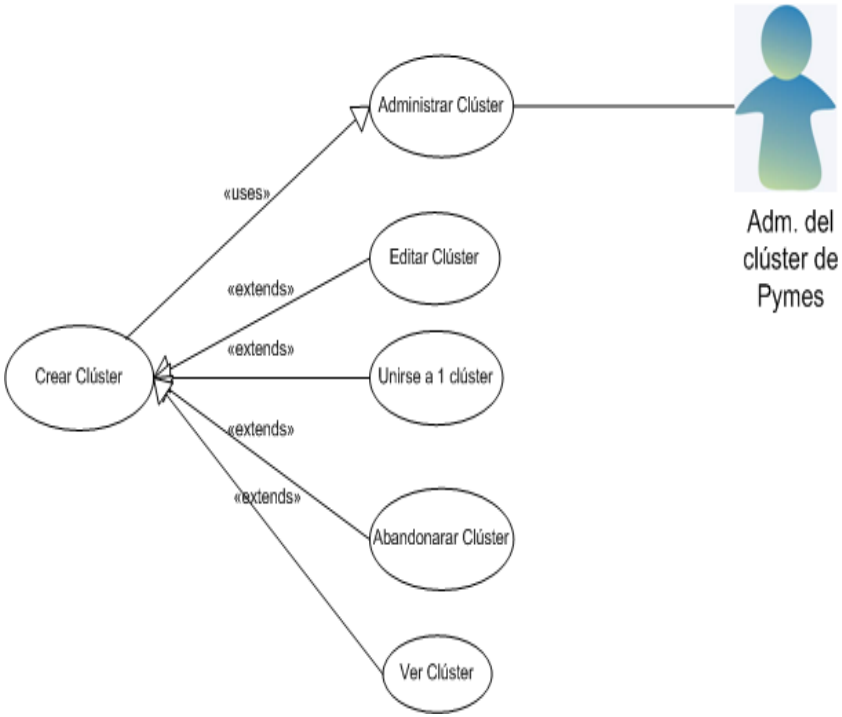


Diagrama3 - Objetivos Administrador del clúster

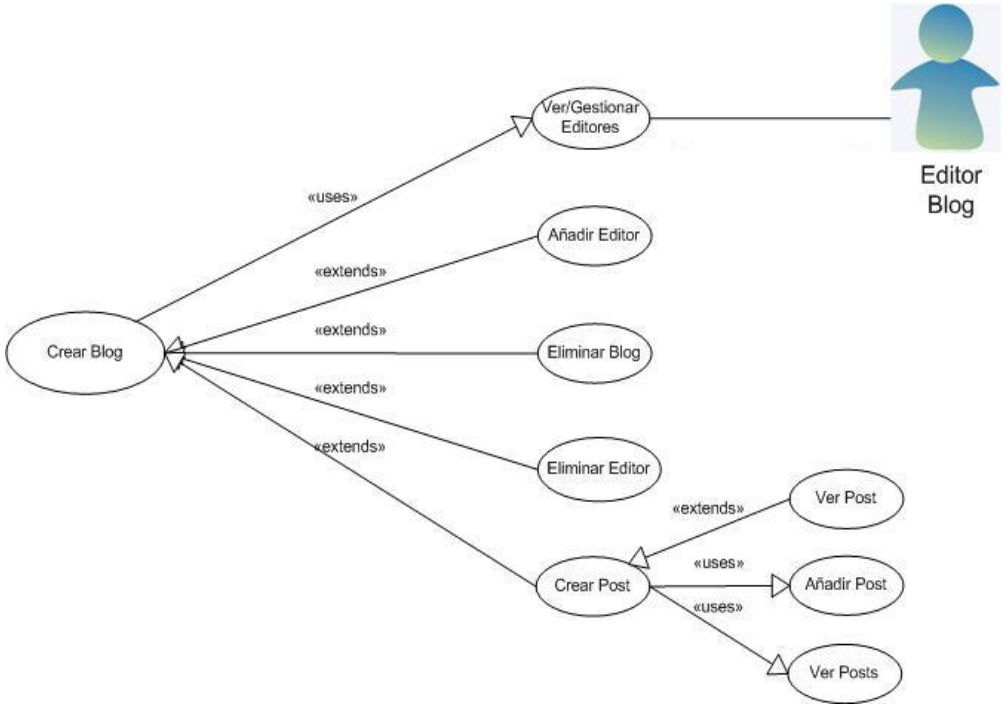


Diagrama4 - Objetivos Editor de Blog

2.4 Listado de Casos de Uso completos

En esta sección se muestran las plantillas de cases de uso completos con la especificación formal de cada uno de ellos. Se ordenan por Identificador para facilitar las búsquedas.

UC 001. Registrarse/Darse de alta

Nombre	Registrarse/Darse de alta	**ID**	001
Creado por	Jonay Santana	**Fecha**	25/05/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	25/05/2010

ID:	001
Ámbito:	Red Social
Nivel:	Función
Descripción:	Este caso de uso consiste en el acceso por primera vez a la red social por parte del usuario.
Actor primario:	Usuario
Individuos e Intereses:	Usuario: hace uso de esta función para poder acceder a la red social. Administrador de la Pyme: hace uso de esta función para crearse una cuenta en la red social. Adm. Pública: se crea un perfil en la red social para participar en la misma.
Trigger:	Invocar a la función registrarse.
Precondición:	Acceder al sitio web donde está alojada la red social.
Poscondición:	El usuario debe acceder a su correo para terminar el registro en la red social.
Flujo Normal:	1. Acceso al sitio web.

	2. Invocar a la función registrarse. 3. Rellenar todos y cada uno de los campos del formulario de registro. Deberá seleccionar si es una Pyme, Adm. Pública o un usuario que quiere conocer las Pymes. 4. Envío de un mail con instrucciones para confirmar la cuenta del usuario. 5. Activación de la cuenta mediante el acceso al email para confirmar su nueva cuenta en la red social.
Flujo Alternativo:	1.a. No dispone de conexión 2.a. Intentar loguearse sin haberse registrado anteriormente. 3.a. El relleno del formulario es incorrecto, y tendría que volver al flujo inicial.
Excepción:	Si cualquiera de los individuos no rellena el formulario adecuadamente al registrarse, deberá corregir los problemas detectados.
Includes:	
Requisitos Especiales:	Rellenar correctamente los campos.
Notas:	

UC 002. Login/Iniciar Sesión

Nombre	Login/Iniciar Sesión	**ID**	002
Creado por	Jonay Santana	**Fecha**	25/05/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	25/05/2010

ID:	002

Ámbito:	Red Social
Nivel:	Función
Descripción:	Este caso de uso consiste en la necesidad del usuario de loguearse para tener acceso a la red social.
Actor primario:	Usuario
Individuos e Intereses:	Usuario: hace uso de esta función para acceder a la red social.
Trigger:	El usuario invoca la funcionalidad del "Login".
Precondición:	El usuario debe haberse registrado anteriormente.
Poscondición:	El usuario puede acceder a la red social.
Flujo Normal:	1. Acceso al sitio web. 2. Invocar a la función login. 3. Rellenar todos y cada uno de los campos del formulario login. 4. Acceso a la red social si se ha logueado correctamente.
Flujo Alternativo:	1.a. El usuario no dispone de conexión. 2.a. Intenta loguearse sin haberse registrado anteriormente. Con lo que tendría que registrarse previamente. 3.a. El relleno de los campos es incorrecto, y tendría que volver al flujo inicial.
Excepción:	Si el usuario no rellena el formulario adecuadamente al loguearse, deberá corregir los problemas detectados.
Includes:	
Requisitos Especiales:	- El usuario debe tener una cuenta en la red social. - El usuario debe rellenar correctamente

	los campos.
Notas:	

UC 003. Cambiar contraseña

Nombre	Cambiar contraseña	**ID**	003
Creado por	Jonay Santana	**Fecha**	25/05/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	25/05/2010

ID:	003
Ámbito:	Red Social
Nivel:	Función
Descripción:	Este caso de uso consiste en que el usuario desea cambiar su contraseña.
Actor primario:	Usuario
Individuos e Intereses:	Usuario: hará uso de esta funcionalidad para realizar un cambio de contraseña.
Trigger:	El usuario entra en su cuenta e invoca la funcionalidad de este caso de uso.
Precondición:	El usuario debe tener una cuenta en la red social.
Poscondición:	El usuario ha cambiado su contraseña actual por una nueva.
Flujo Normal:	1. Acceso al sitio web y loguearse. 2. Indica su cuenta. 3. Rellenar todos y cada uno de los campos del formulario para cambiar la contraseña actual por una nueva. 4. El usuario ha cambiado de contraseña.
Flujo Alternativo:	1.a. El usuario no tiene una cuenta en la

	red social, deberá registrarse. 1.b. No se ha logueado correctamente., deberá hacerlo previamente. 3.a. El usuario no rellena los campos adecuadamente, es decir no introduce los caracteres mínimos o introduce una contraseña actual que no es la suya.
Excepción:	Si el usuario no rellena el formulario adecuadamente del cambio de contraseña, deberá corregir los problemas detectados.
Includes:	
Requisitos Especiales:	- El usuario debe tener una cuenta en la red social. - El usuario debe estar logueado.
Notas:	

UC 004. Borrar Cuenta

Nombre	Borrar cuenta	**ID**	004
Creado por	Jonay Santana	**Fecha**	25/05/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	25/05/2010

ID:	004
Ámbito:	Red Social
Nivel:	Función
Descripción:	Este caso de uso consiste en que el usuario desea eliminar su cuenta de la red social.
Actor primario:	Usuario
Individuos e Intereses:	Usuario: Eliminar su cuenta y borrar todos sus datos.

Trigger:	El usuario invoca esta funcionalidad.
Precondición:	El usuario debe tener una cuenta, donde podrá administrar la misma.
Poscondición:	El usuario habrá eliminado su cuenta y sus datos de la red social. Con lo que ya no podrá acceder a la red social a no ser que se cree una cuenta nueva.
Flujo Normal:	1. Estar logueado. 2. Entrar en su cuenta. 3. Invocar la funcionalidad de borrar su cuenta y todos sus datos. 4. Se indica una advertencia como la siguiente: *"Vamos a borrar tu cuenta y borrar todos tus datos. ¿Estás seguro?"* Y el usuario decidirá.
Flujo Alternativo:	1.a. El usuario no tiene una cuenta en la red social, deberá registrarse. 1.b. No se ha logueado correctamente., deberá hacerlo previamente.
Excepción:	
Includes:	
Requisitos Especiales:	- El usuario debe tener una cuenta en la red social. - El usuario debe estar logueado.
Notas:	El usuario tiene la decisión de si desea o no borrar los datos de su cuenta.

UC 005. Actualizar Perfil

Nombre	Actualizar Perfil	**ID**	005
Creado por	Jonay Santana	**Fecha**	25/05/2010

Modif. por:	Jonay Santana	**Fecha Mod.**	25/05/2010

ID:	005
Ámbito:	Red Social
Nivel:	Función
Descripción:	Este caso de uso consiste en que el usuario puede actualizar su perfil si lo desea.
Actor primario:	Usuario
Individuos e Intereses:	Usuario: Actualizar su perfil.
Trigger:	Entrar en su perfil personal en la red social.
Precondición:	Haber entrado en su perfil, donde el usuario puede actualizarlo.
Poscondición:	El usuario habrá actualizado su perfil.
Flujo Normal:	1. Estar logueado. 2. Entrar en su perfil. 3. Rellenar el formulario para actualizar el perfil. Uno de los campos es subir una foto. 4. El usuario ha actualizado su perfil.
Flujo Alternativo:	1.a. El usuario no tiene una cuenta en la red social, deberá registrarse. 1.b. No se ha logueado correctamente., deberá hacerlo previamente. 4.a. El usuario no rellena los campos adecuadamente, es decir no introduce el formato de las imágenes que son soportadas por la aplicación.
Excepción:	
Includes:	
Requisitos Especiales:	- El usuario debe tener una cuenta en la red social.

	- El usuario debe estar logueado. - El formato de las imágenes puede ser JPG, o JPEG.
Notas:	

UC 006. Finalizar Sesión

Nombre	Finalizar Sesión	**ID**	006
Creado por	Jonay Santana	**Fecha**	25/05/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	25/05/2010

ID:	006
Ámbito:	Red Social
Nivel:	Función
Descripción:	El usuario desea salir de la aplicación
Actor primario:	Usuario
Individuos e Intereses:	Usuario: hace uso de esta función para salir de la red social.
Trigger:	El usuario invoca la funcionalidad del caso de uso.
Precondición:	Haber iniciado sesión anteriormente.
Poscondición:	El usuario habrá cerrado su sesión.
Flujo Normal:	1. Estar logueado. 2. Invocar a Salir. 3. El usuario ha salido de la aplicación.
Flujo Alternativo:	1.a. El usuario no tiene una cuenta en la red social, deberá registrarse. 1.b. No se ha logueado correctamente., deberá hacerlo previamente.
Excepción:	

Includes:	
Requisitos Especiales:	- El usuario debe tener una cuenta en la red social. - El usuario debe estar logueado, es decir debe haber iniciado la sesión anteriormente.
Notas:	

UC 007. Buscar en la aplicación

Nombre	Buscar en la aplicación	**ID**	007
Creado por	Jonay Santana	**Fecha**	25/05/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	25/05/2010

ID:	007
Ámbito:	Red Social
Nivel:	Función
Descripción:	Buscar dentro de la aplicación perfiles, blogs, etc
Actor primario:	Usuario
Individuos e Intereses:	Usuario: Interesado en buscar dentro de la aplicación.
Trigger:	El usuario escribe en el cuadro de búsqueda lo que él desee.
Precondición:	Haber iniciado sesión anteriormente.
Poscondición:	Se mostrarán los resultados de acuerdo con la búsqueda realizada por el usuario.
Flujo Normal:	1. Estar logueado. 2. Escritura en el cuadro de búsqueda. 3. Pulsar Enter en el teclado o invocar a IR.
Flujo Alternativo:	1.a. El usuario no tiene una cuenta en la

	red social, deberá registrarse. 1.b. No se ha logueado correctamente., deberá hacerlo previamente.
Excepción:	
Includes:	
Requisitos Especiales:	- El usuario debe tener una cuenta en la red social. - El usuario debe estar logueado, es decir debe haber iniciado la sesión anteriormente.
Notas:	

UC 008. Enviar Mensaje Privado

Nombre	Enviar Mensaje Privado	**ID**	008
Creado por	Jonay Santana	**Fecha**	25/05/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	25/05/2010

ID:	008
Ámbito:	Red Social
Nivel:	Función
Descripción:	Este caso de uso consiste en que el usuario desea escribir un mensaje privado a un usuario que desconoce.
Actor primario:	Usuario
Individuos e Intereses:	Usuario: hace uso de esta función para enviar un mensaje privado a un usuario que desconoce.
Trigger:	El usuario invoca la funcionalidad enviar

	un mensaje privado a un usuario que desconoce, después de ver su perfil.
Precondición:	- Haber iniciado sesión anteriormente. - Tener un destinatario del mensaje.
Poscondición:	Envío del mensaje privado al destinatario.
Flujo Normal:	1. Estar logueado. 2. Entrar en el perfil de un usuario. 3. Pulsar Enviar mensaje privado 4. Rellenar los campos (De, Para, Asunto y contenido). 5. Invocar a la función Enviar mensaje.
Flujo Alternativo:	1.a. El usuario no tiene una cuenta en la red social, deberá registrarse. 1.b. No se ha logueado correctamente., deberá hacerlo previamente. 2.a. Haber seleccionado un usuario que no conozca.
Excepción:	
Includes:	
Requisitos Especiales:	- El usuario debe tener una cuenta en la red social. - El usuario debe estar logueado, es decir debe haber iniciado la sesión anteriormente. - El usuario debe seleccionar un destinatario del mensaje.
Notas:	

UC 009. Notificar Abuso

Nombre	Notificar Abuso	**ID**	009

Creado por	Jonay Santana	**Fecha**	25/05/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	25/05/2010

ID:	009
Ámbito:	Red Social
Nivel:	Función
Descripción:	Este caso de uso consiste en que el usuario desea notificar un abuso al administrador de la red social.
Actor primario:	Usuario
Individuos e Intereses:	Usuario: Notifica un abuso al administrador. Administrador: Recibe abuso de un usuario.
Trigger:	El usuario invoca a la función notificar un abuso.
Precondición:	Haber iniciado sesión anteriormente.
Poscondición:	Denuncia enviada al administrador de la red social.
Flujo Normal:	1. Estar logueado. 2. Entrar en el perfil de un usuario. 3. Invocar a Notificar de un abuso 4. Rellenar los campos (Asunto y descripción). 5. Se envía la denuncia llamando a Enviar denuncia.
Flujo Alternativo:	1.a. El usuario no tiene una cuenta en la red social, deberá registrarse. 1.b. No se ha logueado correctamente., deberá hacerlo previamente. 3.a. No tiene seleccionado ningún usuario.
Excepción:	

Includes:	
Requisitos Especiales:	- El usuario debe tener una cuenta en la red social. - Asegurarse de explicar correctamente la denuncia en el campo descripción.
Notas:	

UC 010. Seguir a una Pyme

Nombre	Seguir a una Pyme	**ID**	010
Creado por	Jonay Santana	**Fecha**	25/05/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	25/05/2010

ID:	010
Ámbito:	Red Social
Nivel:	Función
Descripción:	Este caso de uso consiste en que un usuario puede seguir a una Pyme de la red social. Es decir tener la posibilidad de conocer cuáles son sus movimientos en la red social.
Actor primario:	Usuario
Individuos e Intereses:	Usuario: especifica a que Pyme quiere seguir.
Trigger:	Invocar a la funcionalidad Seguir a y el nombre de la Pyme.
Precondición:	- Haber iniciado sesión anteriormente. - Tener seleccionado a la Pyme que desea seguir.
Poscondición:	El usuario sigue los movimientos de ese

	Pyme por la red social.
Flujo Normal:	1. Estar logueado. 2. Entrar en el perfil de una Pyme. 3. Invocar a la función Seguir a y el nombre de la Pyme.
Flujo Alternativo:	1.a. El usuario no tiene una cuenta en la red social, deberá registrarse. 1.b. No se ha logueado correctamente., deberá hacerlo previamente. 3.a. No tiene seleccionado ninguna Pyme.
Excepción:	
Includes:	
Requisitos Especiales:	- El usuario debe tener una cuenta en la red social. - El usuario debe elegir a la Pyme que desea seguir.
Notas:	

UC 011. Dejar de seguir a una Pyme.

Nombre	Dejar de seguir a una Pyme	**ID**	011
Creado por	Jonay Santana	**Fecha**	26/05/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	26/05/2010

ID:	011
Ámbito:	Red Social
Nivel:	Función
Descripción:	Este caso de uso consiste en dejar de seguir a una Pyme, al cual antes teníamos la posibilidad de seguir sus movimientos

	por la red social.
Actor primario:	Usuario
Individuos e Intereses:	Usuario: especifica a que Pyme desea dejar de seguir.
Trigger:	Invocar a la funcionalidad dejar de seguir y el nombre de la Pyme.
Precondición:	- Haber iniciado sesión anteriormente. - Tener seleccionado a la Pyme que desea dejar de seguir.
Poscondición:	Dejamos de seguir los movimientos de esa Pyme por la red social.
Flujo Normal:	1. Estar logueado. 2. Entrar en el perfil de una Pyme. 3. Invoca a la función Dejar de seguir a y el nombre de la Pyme.
Flujo Alternativo:	1.a. El usuario no tiene una cuenta en la red social, deberá registrarse. 1.b. No se ha logueado correctamente., deberá hacerlo previamente. 3.a. No tiene seleccionado ninguna Pyme.
Excepción:	
Includes:	
Requisitos Especiales:	- El usuario debe tener una cuenta en la red social. - El usuario debe elegir a la Pyme que desea dejar de seguir.
Notas:	

UC 012. Ver seguidores

Nombre	Ver seguidores	**ID**	012

Creado por	Jonay Santana	**Fecha**	26/05/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	26/05/2010

ID:	012
Ámbito:	Red Social
Nivel:	Función
Descripción:	Este caso de uso consiste en que ver los distintos seguidores que tiene una Pyme.
Actor primario:	Usuario
Individuos e Intereses:	Administrador de la Pyme: Ver sus seguidores.
Trigger:	El administrador de la Pyme entra en su perfil e invoca a la funcionalidad del caso de uso.
Precondición:	- Haber iniciado sesión anteriormente. - Entrar en su perfil.
Poscondición:	Listado con sus seguidores.
Flujo Normal:	1. Estar logueado. 2. Entrar en el perfil de una Pyme. 3. Ver sus seguidores en su perfil
Flujo Alternativo:	1.a. El usuario no tiene una cuenta en la red social, deberá registrarse. 1.b. No se ha logueado correctamente., deberá hacerlo previamente. 3.a. No tiene seleccionado ninguna Pyme. 3.b. No tiene ningún seguidor.
Excepción:	
Includes:	
Requisitos Especiales:	- El usuario debe tener una cuenta en la red social. - El usuario debe elegir a la Pyme de la

	que desea ver sus seguidores.
Notas:	

UC 013. Enviar mensaje

Nombre	Enviar mensaje	**ID**	013
Creado por	Jonay Santana	**Fecha**	26/05/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	26/05/2010

ID:	013
Ámbito:	Red Social
Nivel:	Función
Descripción:	Este caso de uso consiste en enviar un mensaje a un seguidor o personas a las que sigue ese usuario.
Actor primario:	Usuario
Individuos e Intereses:	Usuario: El usuario podrá enviar un mensaje a otro usuario de su red.
Trigger:	Dentro del perfil del usuario, ir a la sección de mensajes. E invocar al caso de uso Crear un nuevo mensaje o alguna de las funciones que usan este caso de uso.
Precondición:	- Haber iniciado sesión anteriormente. - Estar en su perfil en la sección mensajes. - Crear un mensaje.
Poscondición:	Mensaje enviado a su destinatario
Flujo Normal:	1. Estar logueado. 2. Entrar en su perfil de usuario. 3. Entrar en la sección de mensajes. 4. Crear nuevo mensaje

	5. Rellenar campos del mensaje 6. Enviar mensaje
Flujo Alternativo:	1.a. El usuario no tiene una cuenta en la red social, deberá registrarse. 1.b. No se ha logueado correctamente., deberá hacerlo previamente. 2.a. No tiene seleccionado su perfil. 3.a. No está en la sección mensajes. 4.a. Invocar a Crear nuevo mensaje. 4.b. Entrar en la bandeja de mensajes recibidos "Inbox", seleccionar un mensaje e invocar a la función Contestar. 5.a. Si el mensaje a crear ha sido mediante la función Crear un nuevo mensaje los campos a rellenar serán: - Para: seleccionar el destinatario o los destinatarios. - Asunto: indicar el asunto del mensaje a enviar. - Contenido: indicar el contenido del mensaje a enviar. 5.b. Si el mensaje a crear ha sido mediante el acceso a un correo recibido y el usuario desea contestar, los campos a rellenar son: - Asunto: si el usuario desea puede modificar el campo asunto. - Contenido: indicar el contenido del mensaje a enviar. 6.a. El usuario debe indicar siempre un destinatario.
Excepción:	
Includes:	UC 015 →Contestar mensaje

	UC 016 →Ver mensajes recibidos (Inbox)
Requisitos Especiales:	- El usuario debe tener una cuenta en la red social. - El usuario debe elegir al usuario al que desea enviar el mensaje.
Notas:	

UC 014. Eliminar mensaje

Nombre	Eliminar mensaje	**ID**	014
Creado por	Jonay Santana	**Fecha**	26/05/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	26/05/2010

ID:	014
Ámbito:	Red Social
Nivel:	Función
Descripción:	Este caso de uso consiste en la eliminación de un mensaje por parte del usuario, este mensaje puede ser un mensaje enviado o uno recibido.
Actor primario:	Usuario
Individuos e Intereses:	Usuario: El usuario puede eliminar mensajes que haya enviado o recibido.
Trigger:	El usuario o algún de los casos de uso que utiliza este caso de uso invocan la funcionalidad del mismo.
Precondición:	- Haber iniciado sesión anteriormente. - Estar en su perfil en la sección mensajes. - Seleccionar un mensaje
Poscondición:	El mensaje seleccionado será eliminado.

Flujo Normal:	1. Estar logueado. 2. Entrar en su perfil de usuario. 3. Entrar en la sección de mensajes. 4. Seleccionar un mensaje de la bandeja de mensajes recibidos o de la bandeja de mensajes enviados. 5. El usuario invoca a eliminar mensaje. 6. Se elimina el mensaje.
Flujo Alternativo:	1.a. El usuario no tiene una cuenta en la red social, deberá registrarse. 1.b. No se ha logueado correctamente., deberá hacerlo previamente. 2.a. No tiene seleccionado su perfil. 3.a. No está en la sección mensajes.
Excepción:	
Includes:	UC016 → Ver mensajes recibidos (Inbox) UC017 → Ver mensajes enviados(Outbox)
Requisitos Especiales:	- El usuario debe tener una cuenta en la red social. - Entrar en cualquiera de las bandejas de mensajes y seleccionar uno.
Notas:	

UC 015. Contestar mensaje

Nombre	Contestar mensaje	**ID**	015
Creado por	Jonay Santana	**Fecha**	26/05/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	26/05/2010

ID:	015
Ámbito:	Red Social
Nivel:	Función
Descripción:	Este caso de uso consiste en contestar a un mensaje recibido.
Actor primario:	Usuario
Individuos e Intereses:	Usuario: Hace uso de la posibilidad de contestar a un mensaje recibido.
Trigger:	El usuario o algún de los casos de uso que utiliza este caso de uso invocan la funcionalidad del mismo.
Precondición:	- Haber iniciado sesión anteriormente. - Estar en su perfil en la sección mensajes. - Seleccionar un mensaje recibido.
Poscondición:	El mensaje seleccionado será contestado, o lo que es lo mismo envía un mensaje.
Flujo Normal:	1. Estar logueado. 2. Entrar en su perfil de usuario. 3. Entrar en la sección de mensajes. 4. Seleccionar un mensaje de la bandeja de mensajes recibidos. 5. El usuario invoca a contestar mensaje. 6. Relleno de los campos del mensaje a enviar (asunto y contenido). 7. Invoca a enviar mensaje.
Flujo Alternativo:	1.a. El usuario no tiene una cuenta en la red social, deberá registrarse. 1.b. No se ha logueado correctamente., deberá hacerlo previamente. 2.a. No tiene seleccionado su perfil. 3.a. No está en la sección mensajes.
Excepción:	

Includes:	UC 013 → Enviar mensaje.
Requisitos Especiales:	- El usuario debe tener una cuenta en la red social. - Tener seleccionado un mensaje.
Notas:	

UC 016. Ver mensajes recibidos (Inbox)

Nombre	Ver mensajes recibidos (Inbox)	**ID**	016
Creado por	Jonay Santana	**Fecha**	26/05/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	26/05/2010

ID:	016
Ámbito:	Red Social
Nivel:	Función
Descripción:	Se muestran los mensajes que ha recibido el usuario.
Actor primario:	Usuario
Individuos e Intereses:	Usuario: utiliza el caso de uso para poder ver los mensajes que ha recibido.
Trigger:	El usuario invoca la funcionalidad de la aplicación.
Precondición:	- Haber iniciado sesión anteriormente. - Estar en su perfil en la sección mensajes.
Poscondición:	- Listado de los mensajes recibidos.
Flujo Normal:	1. Estar logueado. 2. Entrar en su perfil de usuario. 3. Entrar en la sección de mensajes. 4. Seleccionar bandeja de mensajes

	recibidos “Inbox”.
Flujo Alternativo:	1.a. El usuario no tiene una cuenta en la red social, deberá registrarse. 1.b. No se ha logueado correctamente., deberá hacerlo previamente. 2.a. No tiene seleccionado su perfil. 3.a. No está en la sección mensajes.
Excepción:	
Includes:	
Requisitos Especiales:	- El usuario debe tener una cuenta en la red social.
Notas:	

UC 017. Ver mensajes enviados (Outbox)

Nombre	Ver mensajes enviados (Outbox)	**ID**	017
Creado por	Jonay Santana	**Fecha**	26/05/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	26/05/2010

ID:	017
Ámbito:	Red Social
Nivel:	Función
Descripción:	Se muestran los mensajes que ha enviado el usuario.
Actor primario:	Usuario
Individuos e Intereses:	Usuario: utiliza el caso de uso para poder ver los mensajes que ha enviado.
Trigger:	El usuario invoca la funcionalidad de la aplicación.

Precondición:	- Haber iniciado sesión anteriormente. - Estar en su perfil en la sección mensajes.
Poscondición:	- Listado de los mensajes enviados.
Flujo Normal:	1. Estar logueado. 2. Entrar en su perfil de usuario. 3. Entrar en la sección de mensajes. 4. Seleccionar bandeja de mensajes recibidos "Inbox".
Flujo Alternativo:	1.a. El usuario no tiene una cuenta en la red social, deberá registrarse. 1.b. No se ha logueado correctamente., deberá hacerlo previamente. 2.a. No tiene seleccionado su perfil. 3.a. No está en la sección mensajes.
Excepción:	
Includes:	
Requisitos Especiales:	- El usuario debe tener una cuenta en la red social.
Notas:	

UC 018. Crear clúster

Nombre	Crear clúster	**ID**	018
Creado por	Jonay Santana	**Fecha**	27/05/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	27/05/2010

ID:	018
Ámbito:	Red Social
Nivel:	Función
Descripción:	El usuario tiene la posibilidad de crear un

	clúster donde tendrá la oportunidad de conocer a Pymes con sus mismos intereses.
Actor primario:	Administrador del clúster
Individuos e Intereses:	Administrador del clúster: utiliza el caso de uso para crear un grupo en donde tenga la oportunidad de conocer a Pymes con sus mismos intereses.
Trigger:	Invocar la funcionalidad de la aplicación.
Precondición:	- Haber iniciado sesión anteriormente. - Estar en la sección clúster.
Poscondición:	Creación de un nuevo clúster.
Flujo Normal:	1. Estar logueado. 2. Entrar en la sección clústeres. 3. Invocar a Crear un nuevo clúster. 4. Rellenar formulario del nuevo grupo a crear, los campos a rellenar son (nombre, descripción, etiquetas, imagen, y tipo de grupo si es privado o moderado). 5. Invocar a crear.
Flujo Alternativo:	1.a. El usuario no tiene una cuenta en la red social, deberá registrarse. 1.b. No se ha logueado correctamente., deberá hacerlo previamente. 2.a. No está en la sección clúster.
Excepción:	
Includes:	
Requisitos Especiales:	- El usuario debe tener una cuenta en la red social.
Notas:	

UC 019. Unirse a un clúster

Nombre	Unirse a un clúster	**ID**	019
Creado por	Jonay Santana	**Fecha**	27/05/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	27/05/2010

ID:	019
Ámbito:	Red Social
Nivel:	Función
Descripción:	Posibilidad de unirse a un clúster, para tener la oportunidad de conocer a Pymes con sus mismos intereses.
Actor primario:	Administrador de la Pyme
Individuos e Intereses:	Administrador de la Pyme: utiliza el caso de uso para unirse a un clúster.
Trigger:	El usuario invoca la funcionalidad del caso de uso.
Precondición:	- Haber iniciado sesión anteriormente. - Estar en la sección clúster.
Poscondición:	Unirse y formar parte del clúster.
Flujo Normal:	1. Estar logueado. 2. Entrar en la sección clúster. 3. Invocar a Unirse a este clúster.
Flujo Alternativo:	1.a. El usuario no tiene una cuenta en la red social, deberá registrarse. 1.b. No se ha logueado correctamente., deberá hacerlo previamente. 2.a. No pertenecer a un clúster..
Excepción:	
Includes:	
Requisitos Especiales:	- El usuario debe tener una cuenta en la red social.

Notas:	

UC 020. Abandonar clúster

Nombre	Abandonar clúster	**ID**	020
Creado por	Jonay Santana	**Fecha**	27/05/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	27/05/2010

ID:	020
Ámbito:	Red Social
Nivel:	Función
Descripción:	Posibilidad de abandonar un clúster después de haber formado parte del mismo.
Actor primario:	Usuario
Individuos e Intereses:	Usuario: utiliza el caso de uso para dejar un clúster en el que estuvo.
Trigger:	El usuario invoca la funcionalidad del caso de uso.
Precondición:	- Haber iniciado sesión anteriormente. - Estar en la sección clústeres y formar parte de uno.
Poscondición:	Dejar de formar parte del clúster.
Flujo Normal:	1. Estar logueado. 2. Entrar en la sección clúster. 3. Invocar a Abandonar este clúster. 4. Se muestra advertencia de si está seguro de abandonar ese clúster.
Flujo Alternativo:	1.a. El usuario no tiene una cuenta en la red social, deberá registrarse.

	1.b. No se ha logueado correctamente., deberá hacerlo previamente. 2.a. No pertenece a un clúster. 4.a. El usuario puede aceptar, en ese caso vuelve a la pantalla del clúster con la posibilidad de volver a unirse. 4.b. En caso de que el usuario cancele abandonar ese grupo, volverá a ver el clúster.
Excepción:	
Includes:	
Requisitos Especiales:	- El usuario debe tener una cuenta en la red social. - Estar unido a un clúster.
Notas:	

UC 021. Ver clústeres

Nombre	Ver clústeres	**ID**	021
Creado por	Jonay Santana	**Fecha**	27/05/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	27/05/2010

ID:	021
Ámbito:	Red Social
Nivel:	Función
Descripción:	Se muestran los clústeres creados en la red social.
Actor primario:	Usuario
Individuos e Intereses:	Usuario: hace uso de ver clústeres para ver los grupos creados en la red social.

Trigger:	El usuario invoca la funcionalidad del caso de uso.
Precondición:	- Haber iniciado sesión anteriormente. - Estar en la sección clústeres. - Estar en el perfil de un usuario que haya creado un clúster.
Poscondición:	Se muestra una lista de los clústeres creados.
Flujo Normal:	1. Estar logueado. 2. Entrar en la sección clústeres. 3. Se muestra un listado con clústeres.
Flujo Alternativo:	1.a. El usuario no tiene una cuenta en la red social, deberá registrarse. 1.b. No se ha logueado correctamente., deberá hacerlo previamente. 2.a. Acceder a través de sus clústeres, es decir a través de los clústeres que hayan sido creados por el usuario. 2.b. Acceder a través de la sección clústeres, donde se mostrarán los clústeres creados en la red social.
Excepción:	
Includes:	
Requisitos Especiales:	- El usuario debe tener una cuenta en la red social.
Notas:	

UC 022. Editar clúster

Nombre	Editar clúster	**ID**	022
Creado por	Jonay Santana	**Fecha**	27/05/2010

Modif. por:	Jonay Santana	**Fecha Mod.**	27/05/2010

ID:	022
Ámbito:	Red Social
Nivel:	Función
Descripción:	Con este caso de uso se consigue editar un clúster.
Actor primario:	Usuario
Individuos e Intereses:	Usuario: editar un clúster creado por el usuario.
Trigger:	El usuario invoca la funcionalidad del caso de uso.
Precondición:	- Haber iniciado sesión anteriormente. - Estar en la sección los clúster creados por el usuario y ser administrador del clúster.
Poscondición:	El grupo ha sido editado
Flujo Normal:	1. Estar logueado. 2. Entrar en la sección sus clústeres. 3. Invocar la funcionalidad del caso de uso. 4. Se muestra las características del clúster creado por el usuario. 5. Actualizar el clúster.
Flujo Alternativo:	1.a. El usuario no tiene una cuenta en la red social, deberá registrarse. 1.b. No se ha logueado correctamente., deberá hacerlo previamente. 3.a. No podrá editar ningún clúster si no es el creador o administrador del mismo.
Excepción:	
Includes:	

Requisitos Especiales:	- El usuario debe tener una cuenta en la red social. - El usuario que edita el grupo debe ser el moderador del mismo, es decir el administrador.
Notas:	

UC 023. Traducir aplicación/ Seleccionar Idioma

Nombre	Traducir aplicación / Seleccionar Idioma	**ID**	022
Creado por	Jonay Santana	**Fecha**	28/05/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	28/05/2010

ID:	022
Ámbito:	Red Social
Nivel:	Función
Descripción:	Traducir el texto estático de la aplicación.
Actor primario:	Usuario
Individuos e Intereses:	Usuario: Hace uso del caso para traducir el texto estático de la red social y escoger el idioma que desea.
Trigger:	Invocar a la funcionalidad del caso de uso.
Precondición:	- Haber iniciado sesión anteriormente.
Poscondición:	Texto estático de la red social traducido.
Flujo Normal:	1. Entrar el sitio donde se aloja la red social. 2. Seleccionar Idioma. 3. Se muestra la página traducida.
Flujo Alternativo:	

Excepción:	
Includes:	
Requisitos Especiales:	
Notas:	

UC 024. Crear álbum de fotos

Nombre	Crear álbum de fotos	**ID**	024
Creado por	Jonay Santana	**Fecha**	28/05/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	28/05/2010

ID:	024
Ámbito:	Red Social
Nivel:	Función
Descripción:	Esta caso de uso consiste en la creación de un álbum de fotos por parte de un usuario de la red social.
Actor primario:	Usuario
Individuos e Intereses:	Usuario: Hace uso de la creación de un álbum de fotos, donde podrá subir fotos.
Trigger:	El usuario invoca la funcionalidad del caso de uso.
Precondición:	- Haber iniciado sesión anteriormente. - Entrar en la sección Fotos.
Poscondición:	Creación del álbum de fotos.
Flujo Normal:	1. Estar logueado. 2. Entrar en la sección fotos 3. Invocar la funcionalidad Crear álbum de fotos.

	4. Rellenar campos del álbum (título, descripción y etiquetas). 5. Invocar a crear álbum.
Flujo Alternativo:	1.a. El usuario no tiene una cuenta en la red social, deberá registrarse. 1.b. No se ha logueado correctamente., deberá hacerlo previamente.
Excepción:	
Includes:	
Requisitos Especiales:	- El usuario debe tener una cuenta en la red social.
Notas:	

UC 025. Eliminar álbum de fotos

Nombre	Eliminar álbum de fotos	**ID**	025
Creado por	Jonay Santana	**Fecha**	28/05/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	28/05/2010

ID:	025
Ámbito:	Red Social
Nivel:	Función
Descripción:	Este caso de uso consiste en la eliminación de un álbum de fotos por parte de un usuario de la red social.
Actor primario:	Usuario
Individuos e Intereses:	Usuario: Hace uso de la eliminación de un álbum de fotos, creado antes por él.

Trigger:	El usuario invoca la funcionalidad del caso de uso.
Precondición:	- Haber iniciado sesión anteriormente. - Entrar en la sección Fotos.
Poscondición:	Eliminación del álbum de fotos.
Flujo Normal:	1. Estar logueado. 2. Entrar en la sección fotos y en los álbumes de fotos creados por el usuario. 3. Acceder al álbum de fotos a eliminar 4. Invocar la funcionalidad Eliminar álbum de fotos. 5. Álbum de fotos eliminado.
Flujo Alternativo:	1.a. El usuario no tiene una cuenta en la red social, deberá registrarse. 1.b. No se ha logueado correctamente., deberá hacerlo previamente. 2.a Si no ha creado ningún álbum no podrá álbum alguno.
Excepción:	
Includes:	
Requisitos Especiales:	- El usuario debe tener una cuenta en la red social. - El usuario debe tener seleccionado un álbum que haya sido creado por él anteriormente.
Notas:	

UC 026. Subir foto

Nombre	Subir foto	**ID**	026
Creado por	Jonay Santana	**Fecha**	28/05/2010

Modif. por:	Jonay Santana	**Fecha Mod.**	28/05/2010

ID:	026
Ámbito:	Red Social
Nivel:	Función
Descripción:	Este caso de uso consiste en subir fotos a un álbum de fotos del usuario.
Actor primario:	Usuario
Individuos e Intereses:	Usuario: Utiliza el caso de uso para subir fotos a un álbum de fotos creado por él.
Trigger:	El usuario invoca la funcionalidad de la red social.
Precondición:	- Haber iniciado sesión anteriormente. - Entrar en la sección Fotos y en uno de los álbum creados por él.
Poscondición:	Álbum de fotos con nuevas imágenes.
Flujo Normal:	1. Estar logueado. 2. Entrar en la sección fotos y en los álbumes de fotos creados por el usuario. 3. Acceder al álbum de fotos en el que quiera subir fotos. 4. Invocar al caso de uso. 5. Rellenar campos de la foto a subir (foto, etiquetas, álbum, título de foto y descripción). 6. Invocar a subir. 7. Guardar álbum.
Flujo Alternativo:	1.a. El usuario no tiene una cuenta en la red social, deberá registrarse. 1.b. No se ha logueado correctamente., deberá hacerlo previamente. 2.a Si no ha creado ningún álbum no podrá

	álbum alguno. 6.a. Si el usuario desea subir más fotos, invoca a subir más fotos.
Excepción:	
Includes:	
Requisitos Especiales:	- El usuario debe tener una cuenta en la red social. - El usuario debe tener seleccionado un álbum que haya sido creado por él anteriormente.
Notas:	

UC 027. Eliminar foto

Nombre	Eliminar foto	**ID**	027
Creado por	Jonay Santana	**Fecha**	28/05/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	28/05/2010

ID:	027
Ámbito:	Red Social
Nivel:	Función
Descripción:	Este caso de uso consiste en subir fotos a un álbum de fotos del usuario.
Actor primario:	Usuario
Individuos e Intereses:	Usuario: Utiliza el caso de uso para subir fotos a un álbum de fotos creado por él.
Trigger:	El usuario invoca la funcionalidad de la red social.
Precondición:	- Haber iniciado sesión anteriormente. - Entrar en la sección Fotos y en uno de

	los álbum creados por él.
Poscondición:	Álbum de fotos con nuevas imágenes.
Flujo Normal:	1. Estar logueado. 2. Entrar en la sección fotos y en los álbumes de fotos creados por el usuario. 3. Acceder a la foto a eliminar. 4. Invocar al caso de uso. 5. Se elimina la foto.
Flujo Alternativo:	1.a. El usuario no tiene una cuenta en la red social, deberá registrarse. 1.b. No se ha logueado correctamente., deberá hacerlo previamente. 2.a Si no ha creado ningún álbum no podrá álbum alguno. 3.a. Picando directamente sobre la foto e invocar al caso de uso. 3.b. Yendo al álbum de fotos y eliminar la foto.
Excepción:	
Includes:	
Requisitos Especiales:	- El usuario debe tener una cuenta en la red social. - El usuario debe haber creado un álbum con fotos. - El usuario debe tener seleccionado una foto a eliminar.
Notas:	

UC 028. Crear blog

Nombre	Crear blog	**ID**	028

Creado por	Jonay Santana	**Fecha**	01/06/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	01/06/2010

ID:	028
Ámbito:	Red Social
Nivel:	Función.
Descripción:	Se crea un blog que recopila cronológicamente textos o artículos apareciendo primero el más reciente, donde el usuario conserva siempre la libertad de dejar publicado lo que crea pertinente.
Actor primario:	Usuario
Individuos e Intereses:	Usuario: Utiliza el caso de uso para darse a conocer a otros usuarios de la red social, que compartan temas similares.
Trigger:	El usuario invoca la funcionalidad del caso de uso.
Precondición:	- Haber iniciado sesión anteriormente. - Entrar en la sección Blogs.
Poscondición:	Blog creado.
Flujo Normal:	1. Estar logueado. 2. Entrar en la sección blogs. 3. Invocar al caso de uso. 4. Rellenar campos del blog (título y descripción). 5. Se crea el blog.
Flujo Alternativo:	1.a. El usuario no tiene una cuenta en la red social, deberá registrarse. 1.b. No se ha logueado correctamente., deberá hacerlo previamente.
Excepción:	

Includes:	
Requisitos Especiales:	- El usuario debe tener una cuenta en la red social. - El usuario debe estar en la sección blogs.
Notas:	

UC 029. Eliminar blog

Nombre	Eliminar blog	**ID**	029
Creado por	Jonay Santana	**Fecha**	01/06/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	01/06/2010

ID:	029
Ámbito:	Red Social
Nivel:	Función
Descripción:	Se elimina un blog creado anteriormente por el usuario.
Actor primario:	Usuario
Individuos e Intereses:	Usuario: Utiliza el caso de uso para eliminar el blog creado anteriormente.
Trigger:	El usuario invoca la funcionalidad del caso de uso.
Precondición:	- Haber iniciado sesión anteriormente. - Entrar en la sección Blogs. - Haber creado antes un blog en la red social.
Poscondición:	Blog eliminado.
Flujo Normal:	1. Estar logueado. 2. Entrar en la sección blogs. 3. Entrar en el blog a eliminar

	4. Eliminarse como editor del blog en el apartado editores, siempre y cuando el usuario no haya incluido más editores a parte de él. 5. Se elimina el blog
Flujo Alternativo:	1.a. El usuario no tiene una cuenta en la red social, deberá registrarse. 1.b. No se ha logueado correctamente., deberá hacerlo previamente.
Excepción:	
Includes:	UC 031 → Eliminar editor.
Requisitos Especiales:	- El usuario debe tener una cuenta en la red social. - El usuario debe estar en la sección blogs.
Notas:	Luego el blog no se eliminará hasta que deje de haber un editor del mismo.

UC 030. Añadir editor

Nombre	Añadir editor	**ID**	030
Creado por	Jonay Santana	**Fecha**	01/06/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	01/06/2010

ID:	030
Ámbito:	Red Social
Nivel:	Función
Descripción:	Consiste en como indica el caso de uso añadir un editor al blog creado por el usuario.
Actor primario:	Usuario.

Individuos e Intereses:	Usuario: Utiliza este caso de uso para añadir a su blog otro editor a parte de él.
Trigger:	Invocar la funcionalidad del caso de uso.
Precondición:	- Haber creado antes un blog en la red social. - Entrar en la sección editores del blog.
Poscondición:	Editor añadido.
Flujo Normal:	1. Estar logueado. 2. Entrar en la sección blogs. 3. Entrar en la sección editores del blog y se muestra un listado con los editores del blog. 4. Seleccionar otro editor a través de un cuadro desplegable, donde se le muestran al usuario las distintas alternativas a elegir como editores. 5. Invocar al caso de uso 6. Se añade editor al blog.
Flujo Alternativo:	1.a. El usuario no tiene una cuenta en la red social, deberá registrarse. 1.b. No se ha logueado correctamente., deberá hacerlo previamente. 3.a. Entrar en el blog creado por el usuario e ir a editores del blog. 3.b. En la sección blogs invocar a gestionar editores.
Excepción:	
Includes:	UC 032 → Ver editores/ Gestionar editores
Requisitos Especiales:	- El usuario debe tener una cuenta en la red social. - El usuario debe haber creado un blog.
Notas:	

UC 031. Eliminar editor

Nombre	Eliminar editor	**ID**	031
Creado por	Jonay Santana	**Fecha**	01/06/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	01/06/2010

ID:	031
Ámbito:	Red Social
Nivel:	Función
Descripción:	Consiste en como indica el caso de uso eliminar un editor al blog creado por el usuario.
Actor primario:	Usuario.
Individuos e Intereses:	Usuario: Utiliza este caso de uso para eliminar de su blog a un editor a parte de él.
Trigger:	Invocar la funcionalidad del caso de uso.
Precondición:	- Haber creado antes un blog en la red social. - Entrar en la sección editores del blog.
Poscondición:	Editor eliminado.
Flujo Normal:	1. Estar logueado. 2. Entrar en la sección blogs. 3. Entrar en la sección editores del blog. 4. Se muestra un listado con los editores del blog. 5. Invocar al caso de uso del editor que desea eliminar. 6. Se elimina editor del blog.
Flujo Alternativo:	1.a. El usuario no tiene una cuenta en la red social, deberá registrarse. 1.b. No se ha logueado correctamente., deberá hacerlo previamente.

	3.a. Entrar en el blog creado por el usuario e ir a editores del blog. 3.b. En la sección blogs invocar a gestionar editores.
Excepción:	
Includes:	UC 032 → Ver editores/ Gestionar editores
Requisitos Especiales:	- El usuario debe tener una cuenta en la red social. - El usuario debe haber creado un blog.
Notas:	Si el editor eliminado es el propio usuario y no hay más editores el blog habrá sido eliminado.

UC 032. Ver editores/ Gestionar editores

Nombre	Ver editores/Gestionar editores	**ID**	032
Creado por	Jonay Santana	**Fecha**	01/06/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	01/06/2010

ID:	032
Ámbito:	Red Social
Nivel:	Función
Descripción:	Consiste en como indica el caso de uso ver los diferentes editores de un blog creado por el usuario.
Actor primario:	Usuario.
Individuos e Intereses:	Usuario: Utiliza este caso de uso para ver quiénes son los editores del blog.
Trigger:	Invocar la funcionalidad del caso de uso.
Precondición:	- Haber creado antes un blog en la red

	social.
Poscondición:	Listado de editores del blog.
Flujo Normal:	1. Estar logueado. 2. Entrar en la sección blogs. 3. Invocar al caso de uso 4. Se muestra un listado con los editores del blog.
Flujo Alternativo:	1.a. El usuario no tiene una cuenta en la red social, deberá registrarse. 1.b. No se ha logueado correctamente., deberá hacerlo previamente. 3.a. Entrar en el blog creado por el usuario e ir a editores del blog. 3.b. En la sección blogs invocar a gestionar editores.
Excepción:	
Includes:	
Requisitos Especiales:	- El usuario debe tener una cuenta en la red social. - El usuario debe haber creado un blog.
Notas:	

UC 033. Crear post

Nombre	Crear post	**ID**	033
Creado por	Jonay Santana	**Fecha**	01/06/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	01/06/2010

ID:	033
Ámbito:	Red Social

Nivel:	Función
Descripción:	Este caso de uso consiste en la creación de un nuevo post en el blog creado por el usuario, el creador del post puede ser cualquier editor del blog.
Actor primario:	Usuario
Individuos e Intereses:	Usuario: Utiliza el caso de uso para la creación de un post en un blog del que es editor.
Trigger:	Invocar la funcionalidad del caso de uso.
Precondición:	- Haber creado antes un blog en la red social.
Poscondición:	Nuevo post creado en el blog.
Flujo Normal:	1. Estar logueado. 2. Entrar en la sección blogs y en el blog en el que se desea crear un nuevo post. 3. Invocar al caso de uso. 4. Rellenar campos del post a crear (etiquetas, tipo de post borrador o publicado, fecha de publicación, y el contenido del mismo que incluye el título y cuerpo.) 5. Se crea el post.
Flujo Alternativo:	1.a. El usuario no tiene una cuenta en la red social, deberá registrarse. 1.b. No se ha logueado correctamente., deberá hacerlo previamente. 3.a. Entrar en el blog creado por el usuario e ir a posts. Y allí invocar a crear post. 3.b. Entrar en el blog creado por el usuario e invocar a Añadir Post.
Excepción:	

Includes:	UC 034 → Añadir Post. UC 035 → Ver Posts
Requisitos Especiales:	Ser editor del blog.
Notas:	

UC 034. Añadir post

Nombre	Añadir post	**ID**	034
Creado por	Jonay Santana	**Fecha**	01/06/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	01/06/2010

ID:	034
Ámbito:	Red Social
Nivel:	Función
Descripción:	Este caso de uso consiste en la creación de un nuevo post en el blog creado por el usuario, el creador del post puede ser cualquier editor del blog.
Actor primario:	Usuario
Individuos e Intereses:	Usuario: Utiliza el caso de uso para la creación de un post en un blog del que es editor.
Trigger:	Invocar la funcionalidad del caso de uso.
Precondición:	- Haber creado antes un blog en la red social.
Poscondición:	Nuevo post creado en el blog.
Flujo Normal:	1. Estar logueado. 2. Entrar en la sección blogs y en el blog en el que se desea añadir un nuevo post. 3. Invocar al caso de uso.

	4. Rellenar campos del post a añadir (etiquetas, tipo de post borrador o publicado, fecha de publicación, y el contenido del mismo que incluye el título y cuerpo.) 5. Se añade el post.
Flujo Alternativo:	1.a. El usuario no tiene una cuenta en la red social, deberá registrarse. 1.b. No se ha logueado correctamente., deberá hacerlo previamente. 3.a. Entrar en el blog creado por el usuario e invocar a Añadir Post. 3.b. Entrar en el blog creado por el usuario e ir a posts. Y allí invocar a crear post.
Excepción:	
Includes:	UC 035 → Ver Posts
Requisitos Especiales:	Ser editor del blog.
Notas:	

UC 035. Ver posts

Nombre	Añadir post	**ID**	035
Creado por	Jonay Santana	**Fecha**	01/06/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	01/06/2010

ID:	035
Ámbito:	Red Social
Nivel:	Función
Descripción:	Este caso de uso consiste en ver los post publicados en el blog del usuario.

Actor primario:	Usuario
Individuos e Intereses:	Usuario: Utiliza el caso de uso para ver los diferentes post publicados en un blog.
Trigger:	Invocar la funcionalidad del caso de uso.
Precondición:	- Haber creado antes un blog en la red social. - Haber publicado en ese blog algún post.
Poscondición:	Listado de Post
Flujo Normal:	1. Estar logueado. 2. Entrar en la sección blogs. 3. Invocar al caso de uso. 4. Se muestra un listado con los posts creados.
Flujo Alternativo:	1.a. El usuario no tiene una cuenta en la red social, deberá registrarse. 1.b. No se ha logueado correctamente., deberá hacerlo previamente. 3.a. Entrar en el blog creado por el usuario 3.b. Entrar en el blog creado por el usuario e ir a posts. Y allí invocar a crear post.
Excepción:	
Includes:	
Requisitos Especiales:	Ser editor del blog.
Notas:	

UC 036. Ver post

Nombre	Ver post	**ID**	036
Creado por	Jonay Santana	**Fecha**	01/06/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	01/06/2010

ID:	036
Ámbito:	Red Social
Nivel:	Función
Descripción:	Este caso de uso consiste en ver un post publicados en el blog de un usuario.
Actor primario:	Usuario
Individuos e Intereses:	Usuario: Utiliza el caso de uso para ver un post publicado en un blog.
Trigger:	Acceder al post del blog.
Precondición:	- Haber iniciado sesión anteriormente. - Entrar en la sección Blogs.
Poscondición:	Visualización del post deseado.
Flujo Normal:	1. Estar logueado. 2. Entrar en la sección blogs. 3. Acceder al post del blog. 4. Se muestra el texto del post.
Flujo Alternativo:	1.a. El usuario no tiene una cuenta en la red social, deberá registrarse. 1.b. No se ha logueado correctamente., deberá hacerlo previamente. 2.a. Puede que no haya sido creado ningún post.
Excepción:	
Includes:	
Requisitos Especiales:	
Notas:	

UC 037. Crear Evento

Nombre	Crear Evento	**ID**	037
Creado por	Jonay Santana	**Fecha**	02/06/2010

Modif. por:	Jonay Santana	**Fecha Mod.**	02/06/2010

ID:	037
Ámbito:	Red Social
Nivel:	Función
Descripción:	El usuario en este caso las Pymes tienen la posibilidad de crear un evento, es decir suceso importante y programado, de índole social, académica, artística o deportiva organizado por una Pyme o adm. Pública.
Actor primario:	Pyme
Individuos e Intereses:	Pyme: programar un evento relacionado con la propia empresa. Adm. Pública: dar a conocer un acontecimiento importante para las Pymes.
Trigger:	Invocar al caso de uso.
Precondición:	- Haber iniciado sesión anteriormente. - Entrar en la sección Eventos.
Poscondición:	Evento creado.
Flujo Normal:	1. Estar logueado. 2. Entrar en la sección eventos. 3. Invocar al caso de uso. 4. Rellenar el formulario del evento indicando Nombre, descripción, lugar, hora, etc. 5. Evento Publicado.
Flujo Alternativo:	1.a. El usuario no tiene una cuenta en la red social, deberá registrarse. 1.b. No se ha logueado correctamente., deberá hacerlo previamente. 4.a. El relleno de los campos es incorrecto,

	y tendría que volver al formulario.
Excepción:	
Includes:	
Requisitos Especiales:	Ser una Pyme o adm. pública.
Notas:	

UC 038. Ver Evento

Nombre	Ver Evento	**ID**	038
Creado por	Jonay Santana	**Fecha**	02/06/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	02/06/2010

ID:	038
Ámbito:	Red Social
Nivel:	Función
Descripción:	El usuario tiene la posibilidad de ver un evento, es decir suceso importante y programado, de índole social, académica, artística o deportiva organizado por una Pyme o adm. Pública.
Actor primario:	Pyme
Individuos e Intereses:	Usuario: ver un evento creado por una pyme o adm. pública
Trigger:	Invocar al caso de uso.
Precondición:	- Haber iniciado sesión anteriormente. - Entrar en la sección Eventos.
Poscondición:	Evento creado.
Flujo Normal:	1. Estar logueado. 2. Entrar en la sección eventos. 3. Invocar al caso de uso.

	4. Evento mostrado
Flujo Alternativo:	1.a. El usuario no tiene una cuenta en la red social, deberá registrarse. 1.b. No se ha logueado correctamente., deberá hacerlo previamente.
Excepción:	
Includes:	
Requisitos Especiales:	
Notas:	

UC 039. Participar en el foro

Nombre	Participar en el foro	**ID**	039
Creado por	Jonay Santana	**Fecha**	03/06/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	03/06/2010

ID:	039
Ámbito:	Red Social
Nivel:	Función
Descripción:	El usuario desea participar en el foro, escribiendo un tema o discusión nuevo.
Actor primario:	Usuario
Individuos e Intereses:	Usuario: ser partícipe del foro, teniendo la oportunidad de participar en una discusión.
Trigger:	Invocar al caso de uso.
Precondición:	- Haber iniciado sesión anteriormente. - Estar en la sección del Foro.
Poscondición:	- Participación en el foro.
Flujo Normal:	1. Estar logueado.

	2. Invocar al caso de uso 3. Escribir un nuevo tema o discusión.
Flujo Alternativo:	
Excepción:	
Includes:	UC 043 →Enviar comentario
Requisitos Especiales:	-Haber entrado en el foro y escribir un nuevo tema.
Notas:	

UC 040. Publicar oferta/demanda de productos y servicios

Nombre	Publicar oferta/demanda de productos y servicios	**ID**	040
Creado por	Jonay Santana	**Fecha**	03/06/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	03/06/2010

ID:	040
Ámbito:	Red Social
Nivel:	Función
Descripción:	El usuario desea publicar una oferta o demanda de un producto o servicio.
Actor primario:	Pyme, Adm. Pública.
Individuos e Intereses:	Pyme: presentar una oferta o demanda de un producto o servicio. Adm. Pública: al igual que la Pyme dar a conocer una oferta o demanda del producto o servicio.
Trigger:	Invocar al caso de uso.
Precondición:	- Haber iniciado sesión anteriormente. - Estar en la sección de Productos y

	Servicios.
Poscondición:	Oferta o demanda publicada.
Flujo Normal:	1. Estar logueado. 2. Entrar en la sección productos y servicios. 3. Invocar al caso de uso. 4. Rellenar los campos de la oferta/demanda. 5. Oferta o demanda publicada.
Flujo Alternativo:	
Excepción:	
Includes:	UC 042 → Ver oferta/demanda de productos y servicios.
Requisitos Especiales:	
Notas:	

UC 041. Eliminar oferta/demanda de productos y servicios

Nombre	Eliminar oferta/demanda de productos y servicios	**ID**	041
Creado por	Jonay Santana	**Fecha**	03/06/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	03/06/2010

ID:	041
Ámbito:	Red Social
Nivel:	Función
Descripción:	El usuario desea eliminar una oferta o demanda de un producto o servicio que ha sido publicada anteriormente.
Actor primario:	Pyme, Adm. Pública.

Individuos e Intereses:	Pyme: eliminar oferta o demanda de un producto o servicio. Adm. Pública: al igual que la Pyme eliminar una oferta o demanda del producto o servicio.
Trigger:	Invocar al caso de uso.
Precondición:	- Haber iniciado sesión anteriormente. - Estar en la sección de Productos y Servicios.
Poscondición:	Oferta o demanda eliminada.
Flujo Normal:	1. Estar logueado. 2. Entrar en la sección productos y servicios. 3. Invocar al caso de uso. 4. Oferta o demanda eliminada.
Flujo Alternativo:	
Excepción:	
Includes:	UC 042 → Ver oferta/demanda de productos y servicios.
Requisitos Especiales:	
Notas:	

UC 042. Ver oferta/demanda de productos y servicios

Nombre	Publicar oferta/demanda de productos y servicios	**ID**	042
Creado por	Jonay Santana	**Fecha**	03/06/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	03/06/2010

ID:	042

Ámbito:	Red Social
Nivel:	Función
Descripción:	El usuario desea ver una oferta o demanda de un producto o servicio.
Actor primario:	Pyme, Adm. Pública. Usuario.
Individuos e Intereses:	Pyme: ver una oferta o demanda de un producto o servicio, realizada por otra Pyme o una adm. Pública. Adm. Pública: al igual que la Pyme ver una oferta o demanda de un producto o servicio, realizada por otra adm. Pública. o una Pyme. Usuario: ver una oferta/demanda de un producto o servicio de una Pyme o adm. Pública.
Trigger:	Invocar al caso de uso.
Precondición:	- Haber iniciado sesión anteriormente. - Estar en la sección de Productos y Servicios.
Poscondición:	Oferta o demanda vista.
Flujo Normal:	1. Estar logueado. 2. Entrar en la sección productos y servicios. 3. Invocar al caso de uso. 4. Oferta o demanda mostrada por pantalla.
Flujo Alternativo:	
Excepción:	
Includes:	
Requisitos Especiales:	
Notas:	

UC 043. Enviar comentario

Nombre	Enviar comentario	**ID**	043
Creado por	Jonay Santana	**Fecha**	03/06/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	03/06/2010

ID:	043
Ámbito:	Red Social
Nivel:	Función
Descripción:	El usuario desea enviar un comentario
Actor primario:	Usuario
Individuos e Intereses:	Usuario: escribir un comentario en relación a una foto, blog, o en el foro.
Trigger:	Invocar al caso de uso.
Precondición:	- Haber iniciado sesión anteriormente. - Estar en cualquiera de las secciones donde puede enviar un comentario.
Poscondición:	- Comentario enviado
Flujo Normal:	1. Estar logueado. 2. Entrar en cualquiera de las secciones donde el usuario pueda comentar. 3. Invocar al caso de uso. 4. Escribir el comentario. 5. Comentario enviado.
Flujo Alternativo:	
Excepción:	
Includes:	
Requisitos Especiales:	- Estar en una foto y escribir un comentario relacionado con la misma. - Estar en un blog en el cual pueda editar el usuario y escriba un comentario. - Estar en el foro, entrar en algún tema o

	discusión y escribir un comentario.
Notas:	

UC 044. Eliminar comentario

Nombre	Eliminar comentario	**ID**	044
Creado por	Jonay Santana	**Fecha**	03/06/2010
Modif. por:	Jonay Santana	**Fecha Mod.**	03/06/2010

ID:	044
Ámbito:	Red Social
Nivel:	Función
Descripción:	El usuario desea eliminar un comentario que ha sido publicado en una foto suya, en su blog o en el foro.
Actor primario:	Usuario
Individuos e Intereses:	Usuario: eliminar un comentario en relación a una foto suya, en su blog, o en el foro que ha sido publicado anteriormente.
Trigger:	Invocar al caso de uso.
Precondición:	- Haber iniciado sesión anteriormente. - Estar en cualquiera de las secciones donde puede eliminar un comentario.
Poscondición:	- Comentario eliminado.
Flujo Normal:	1. Estar logueado. 2. Entrar en cualquiera de las secciones donde el usuario pueda comentar. 3. Invocar al caso de uso. 4. Comentario eliminado.

Flujo Alternativo:	
Excepción:	
Includes:	
Requisitos Especiales:	- Estar en una foto y eliminar un comentario que ha sido escrito anteriormente por otro usuario. - Estar en un blog en el cual pueda editar el usuario y elimine un comentario publicado anteriormente por otro editor. - Estar en el foro, en algún tema o discusión y eliminar un comentario.
Notas:	

2.5 Requerimientos funcionales adicionales

En esta sección se definen los requerimientos funcionales que no se pueden expresar mediante casos de uso. Estos requerimientos se asocian a los casos de uso como restricciones sobre su funcionamiento formas concretas de hacer algunas acciones.

R04. Restricciones del nombre de un usuario

Tipo: Requerimiento funcional		Casos de uso:
Nombre:	Restricciones del nombre de un usuario	
Justificación:	Se debe garantizar que el nombre de un usuario sea único e idéntico independientemente de la plataforma o idioma del usuario.	
Origen:	Jonay Santana	
Criterio de validación:	Los nombres de los usuarios son únicos y se componen de una combinación de caracteres alfanuméricos y guiones bajos.	
Satisfacción del cliente: 2		Insatisfacción del cliente: 2
Prioridad: 5		Conflictos:
Historia:	15/06/2010 – Creación	

Descripción:	El nombre de cada usuario debe ser único y debe estar compuesto por una combinación de caracteres alfanuméricos y guiones bajos.

2.6 Requerimientos no funcionales del sistema

Conjunto de requerimientos que no tienen que ver con la definición del funcionamiento del sistema sino con características adicionales de relevancia.

2.6.1 Requerimientos de Usabilidad

R07. Eficiencia de uso

Tipo: Requerimiento de Usabilidad		Casos de uso:
Nombre:	Eficiencia de uso	
Justificación:	El usuario debe poder acceder a cualquier área de la aplicación de la forma más eficiente.	
Origen:	Jonay Santana	
Criterio de validación:	El usuario puede lanzar cualquier caso de uso mediante el número mínimo de acciones.	
Satisfacción del cliente: 5	Insatisfacción del cliente: 4	
Prioridad: 3	Conflictos:	
Historia:	15/06/2010 – Creación	
Descripción:	Si se define la eficiencia como la posibilidad de que un usuario use el sistema de forma rápida y precisa, se busca con este requerimiento maximizar esa eficiencia.	

R08. Aplicación intuitiva

Tipo: Requerimiento de usabilidad		Casos de uso:
Nombre:	Aplicación intuitiva	
Justificación:	La aplicación debe facilitar su uso a usuarios novatos o que lleven mucho tiempo sin usar la aplicación.	

Origen:	Jonay Santana
Criterio de validación:	El 80% de los usuarios sin experiencia son capaces de utilizar el sistema adecuadamente.
Satisfacción del cliente: 5	Insatisfacción del cliente: 3
Prioridad: 3	Conflictos:
Historia:	15/06/2010 – Creación
Descripción:	La aplicación debe ser lo más intuitiva posible para facilitar a los usuarios su uso sin que sea necesario recurrir frecuentemente a los manuales de usuario. Una vía para conseguir este objetivo es mostrar ayuda en línea y estudiar los nombres de las etiquetas o los botones de la aplicación para que reflejen su utilidad de forma precisa.

R09. Sencillez del entorno

Tipo: Requerimiento de usabilidad	Casos de uso:
Nombre:	Sencillez del entorno
Justificación:	El entorno de la aplicación debe ser sencillo de entender para personas sin conocimiento sobre redes sociales.
Origen:	Jonay Santana
Criterio de validación:	El 80% de las personas son capaces de usar la aplicación tras un período muy corto de familiarización con la misma (1 semana).
Satisfacción del cliente: 5	Insatisfacción del cliente: 2
Prioridad: 2	Conflictos:
Historia:	15/06/2010 – Creación
Descripción:	Se debe garantizar un entorno intuitivo para que sea fácil de usar.

R10. Localización del producto

Tipo: Requerimiento de usabilidad		Casos de uso:
Nombre:	Localización del producto	
Justificación:	La aplicación podrá ser usada por usuarios de diferentes países.	
Origen:	Jonay Santana	
Criterio de validación:	La aplicación permite seleccionar el lenguaje.	
Satisfacción del cliente: 5	Insatisfacción del cliente: 0	
Prioridad: 2	Conflictos:	
Historia:	15/06/2010 – Creación	
Descripción:	La aplicación seré mayoritariamente utilizada por usuarios españoles por lo que como mínimo tendrá soporte para el idioma español. Sería interesante sin embargo que soportase además inglés o bien que estuviera preparado para soportarlo con pequeñas modificaciones.	

2.6.1 Requerimientos de rendimiento

Los requerimientos de rendimiento imponen límites a algunos parámetros del sistema como el tiempo de respuesta de la aplicación, la tolerancia a fallos de de la aplicación o el tiempo de respuesta de la misma.

R11. Velocidad de respuesta

Tipo: Requerimiento de latencia		Casos de uso:
Nombre:	Velocidad de respuesta	
Justificación:	Un usuario debe saber en todo momento que la aplicación está respondiendo a sus órdenes y que no se ha bloqueado.	
Origen:	Jonay Santana	
Criterio de validación:	La aplicación muestra al usuario alguna indicación del progreso antes de 2 segundos.	

Satisfacción del cliente: 5	Insatisfacción del cliente: 3
Prioridad: 3	Conflictos:
Historia:	15/06/2010 – Creación
Descripción:	Cada vez que el usuario ejecute una orden, la aplicación mostrará una indicación del progreso.

R12. Recuperación ante pérdida de contacto con el servidor

Tipo: Requerimiento de robustez		Casos de uso:
Nombre:	Recuperación ante pérdida de contacto con el servidor	
Justificación:	Definir el comportamiento deseable de la aplicación si un usuario pierde el contacto con el servidor sin finalizar su sesión.	
Origen:	Jonay Santana	
Criterio de validación:	El usuario recibe información sobre la pérdida de conexión.	
Satisfacción del cliente: 4		Insatisfacción del cliente: 3
Prioridad: 3		Conflictos:
Historia:	15/06/2010 – Creación	
Descripción:	El usuario recibe información sobre la pérdida de la conexión.	

Actividad 1.4: Prototipo de Interfaz.

Actividad en la que desarrollaremos un prototipo de la interfaz.

1. Desarrollo de prototipo de la interfaz

El prototipado de la aplicación permite refinar el análisis de los requerimientos anteriormente efectuado o identificar casos de uso aún no identificados y definir de esta forma

la interfaz del usuario, además de servir como prueba de concepto de la implementación de la aplicación antes de integrar ideas sobre la misma. Para ello diseñamos varios prototipos planteados conjuntamente con la ayuda de los tutores, permitiendo de este modo buscar fallos o mejoras hasta obtener una interfaz óptima y una identificación de casos de uso completa.

El prototipo fueron realizados en HTML y se les dotó de ciertas funcionalidads que simulen el resultado final de la aplicación mediante JQuery, se trata de una biblioteca o framework de Javascript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el arbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la tecnología AJAX a páginas web.

Este primer prototipo permite identificar los requisitos técnicos de la interfaz de la aplicación, ayudando a determinar las herramientas más adecuadas para construirlas.

1.1 Primera aproximación a la interfaz de la aplicación

En esta primera aproximación se creó una interfaz HTML + JQuery que simula el aspecto de la interfaz, en concreto cuando un usuario accede por primera vez a la plataforma social, siendo necesario registrarse. También creamos la interfaz de la página una vez el usuario se ha logueado.

En la siguiente imagen se muestra la página de inicio, es decir aquella cuando el usuario accede por primera vez

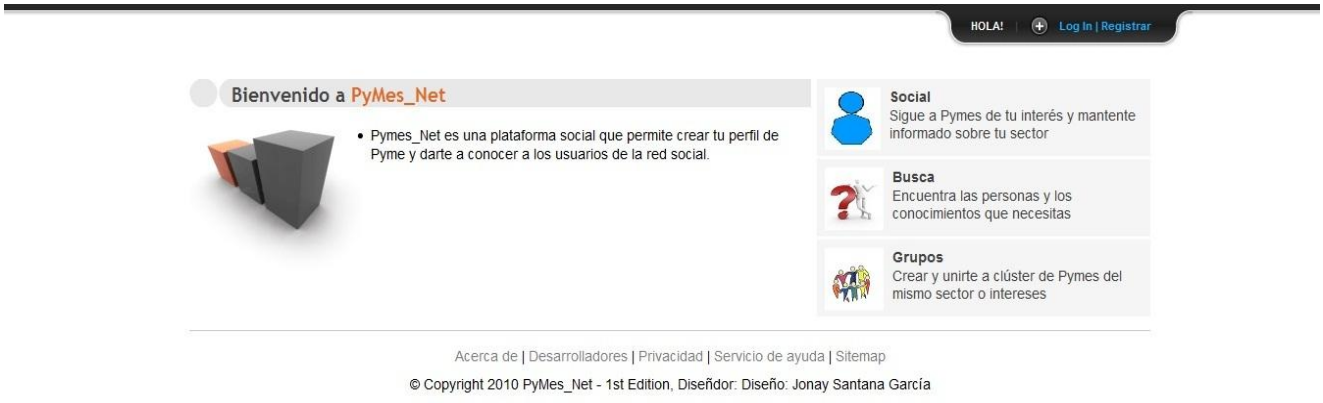


Imagen – Página de inicio

Si nos fijamos bien en la imagen anterior vemos como en la parte superior derecha se muestra un botón que dice “Login | Registrar”. Si pulsamos en él veremos un panel donde el usuario podrá registrarse o loguearse si ya dispone de una cuenta en la red social.

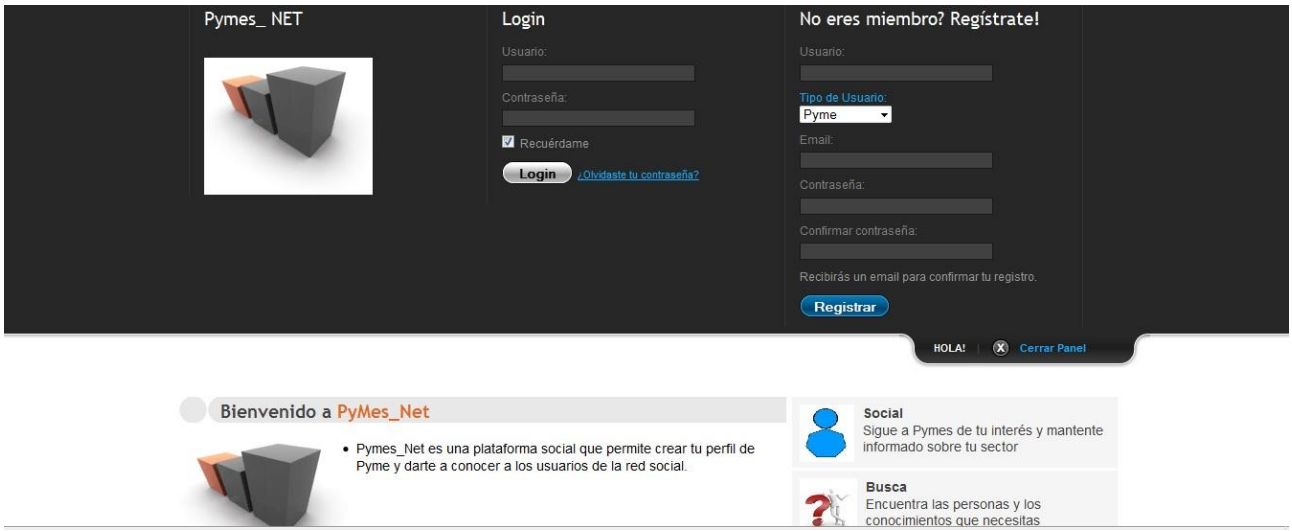


Imagen – Página Login | Registro

Y por último una vez que el usuario ya se ha registrado se le muestra la página de la plataforma con todas sus funcionalidades. Estas se encuentran en un menú en la parte superior de la página. Además el usuario también verá su perfil.

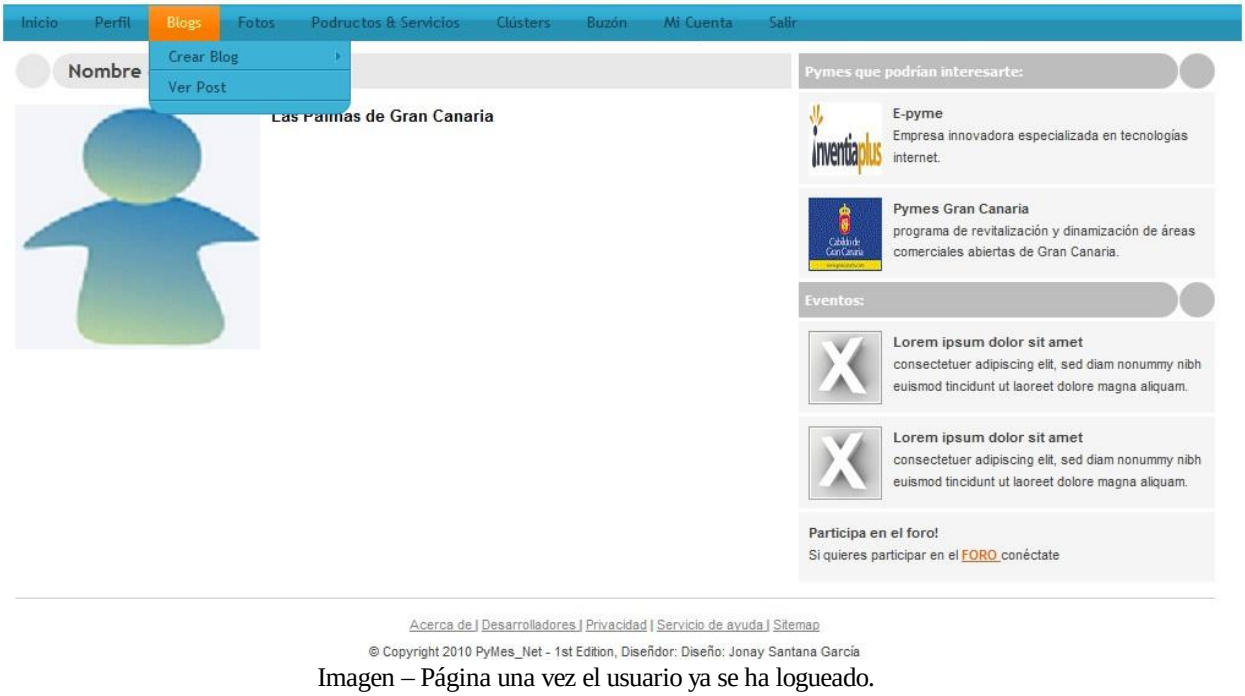


Imagen – Página una vez el usuario ya se ha logueado.

Etapa 2 de Mi proyecto: Diseño

Desarrollo de la etapa 2 de mi proyecto

Resultados y conclusiones.

Anexo. Manuales de usuario

Apéndice. Detalles sobre la implementación del proyecto

Trabajo Futuro.

Bibliografía.

Para la realización de este proyecto se han tenido en cuenta las siguientes fuentes bibliográficas:

Libros:

- Advanced Rails Recipes. Mike Clark and the Rails Community. The Pragmatic Bookself.
- Agile Web Development with Rails, Second Edition. Dave Thomas, David Heinemeier Hansson. The Pragmatic Bookself.
- Programming Ruby. David Thomas, Andrew Hunt. The Pragmatic Programmer´s Guide. Addison Wesley Professional.
- Rails for PHP Developers. Derek DeVries, Mike Naberezny. The Pragmatic Bookself.
- Software Engineering. Sommerville. Addison Weasley.
- Writting Effective Use Cases. Alistair Cockburn. Addison Weasley.

Internet:

- Blog de Ruby: http://weblog.rubyonrails.org/2007/12/7/rails-2-0-it-s-done
- Foro de Ruby: http://www.ruby-forum.org
- Gemas de Ruby: http://www.rubygems.org/
- Grupo TOG: http://groups.google.es/group/tog_users
- Manual de Ruby on-line: http://www.devarticles.com/c/a/Ruby-on-Rails/Ruby-Operators-and-Arrays/
- TOG: http://www.toghq.org/
- Wikipedia: http://es.wikipedia.org
- ISI Web of Knowledge: http://sauwok.fecyt.es/apps/UA_GeneralSearch_input.do?product=UA&search_mode=GeneralSearch&SID=R1N1a79Ino7IhhHEMbC&preferencesSaved=

Artículos:

Adler, P. & Kwon, S. (2002). Social capital: prospects for a new concept. Academy Management Review.

Anderson, A. & Jack, S. (2002). The articulation of social capital in entrepreneurial networks: a glue or a lubricant? Entrepreneurship & Regional Development.

Anderson, A. & Drakopoulou, J. (2005). The role of family members in entrepreneurial networks: beyond the boundaries of the family □rm. Family Business Review.

Andresen, E., Bergman, A. and Hallen, L. (2006), *"The role of e-mail communication in strategic networks: patterns observed over time",* 22nd IMP Conference, Milan.

Anria Sophia van Zyl (2008) *The impact of Social Networking 2.0 on organisations,* Stellenbosch, South Africa

Ariyur, K. (2008), *"The Wikinomics playbook: mass collaboration in action".*

Benkler, Y. (2006), *The Wealth of Networks: How Social Production Transforms Markets and Freedom,* Yale University Press, London.

Boshoff, M. and du Plessis, T. (2008), *"Preferred communication methods and technologies for organisational knowledge sharing and decision making",* South African Journal of Information Management, Vol. 10 No. 2.

Boyd, D.M. and Ellison, N.N. (2007), *"Social network sites: definition, history, and scholarship",* Journal of Computer-Mediated Communication, Vol. 13 No. 1.

Boyd, S. (2006), *"Are you ready for social software?".*

Brown, J.S. and Duguid, P. (2000), *The Social Life of Information,* HBS Press, Boston, MA.

Burger, E. and Rensleigh, C. (2007), *"Investigating e-mail overload in the South African banking industry",* South African Journal of Information Management, Vol. 9 No. 2.

Burt, R. (2000). The network structure of social capital. En R. Sutton y B. Staw (Eds.), Research in organizational behavior. Greenwich, UK: JAI Press.

Butler, J. & Hansen, G. (1991). Network evolution, entrepreneurial success, and regional development. Entrepreneurship and Regional Development

Casson, M. & Della Giusta, M. (2007). Entrepreneurship and social capital: Analysing the impact of social networks on entrepreneurial activity from a rational action perspective. International Small Business Journal.

Cairncross, F. (2001), *The Death of Distance 2.0: How the Communications Revolution Will Change Our Lives,* Texere, London.

ClearSwift (2007a), *"15 Common mistakes in web security: enterprise vulnerabilities that invite attack".*

ClearSwift (2007b), *Content Security 2.0: The Impact of Web 2.0 on Corporate Security.*

ClearSwift (2007c), *"Data leakage: the stealth threat to business".*

ClearSwift (2007d), *"Demystifying Web 2.0: opportunities, threats, defences",* available

Coase, R.H. (1937), *"The nature of the firm",* Economica, Vol. 4 No. 16, pp. 386-405.

Davenport, E. (2001), *"Knowledge management issues for online organisations: 'Communities of Practise' as an exploratory framework",* Journal of Documentation, Vol. 57 No. 1, pp. 66-75.

Fu, F., Liu, L. and Wang, L. (2007), *"Empirical analysis of online social networks in the age of the Web 2.0",* Physica A, Vol. 378 Nos 2/3, pp. 678-85.

Graham, D. and Hall, H. (2004), *"Creation and recreation: motivating collaboration to generate knowledge capital in online communities",* International Journal of Information Management, Vol. 105 No. 2/4, pp. 235-46.

Granovetter, M. (1983), *"The strength of weak ties: a network theory revisited",* Sociological Theory, Vol. 1 No. 1, pp. 201-33.

Granovetter, M. (2004), *"The impact of social structure on economic outcomes",* Journal of Economic Perspectives, Vol. 19 No. 1, pp. 33-50.

Granovetter, M.S. (1973), *"The strength of weak ties",* American Journal of Sociology, Vol. 78 No. 6, pp. 1360-80.

Granovetter, M. (1973). The strength of weak ties. American Journal of Sociology

Godwin-Jones, R. (2006), *"Emerging technologies: tag clouds in the blogosphere: electronic literacy and social networking",* Language, Learning & Technology, Vol. 10 No. 2, p. 8.

Gorge, M. (2007), *"Security for third level education organisations and other educational bodies"*, Computer Fraud & Security, Vol. 2007 No. 7, pp. 6-9.

Hansen, E. (1995). Entrepreneurial networks and new organization growth. Entrepreneurship Theory and Practice.

IBM (2007), *"Achieving tangible business benefits with social computing".*

Jenssen, J. & Koenig, H. (2002). The effect of social networks on resource access and business start-ups. European Planning Studies.

Johannison, B. (2000). Networking and entrepreneurial growth. En D. Sexton y H. Landstrom (Eds.), The Blackwell handbook of entrepreneurship.

Kasperskylab (2008), *Security Trends 2008*.

Larson, A. & Starr, J. (1993). A network model of organization formation. Entrepreneurship: Theory and Practice.

Leitch, S. and Warren, M. (2006), "Social engineering and its impact via the internet", in Valli, C. and Woodward, A. (Eds), *Proceedings of the 4th Australian Information Security*

Management Conference, Western Australia: Edith Cowan University, Perth, pp. 184-9.

Malecki, J. & Veldhoen, M. (1993). Network activities, information and competitiveness in small firms. Geograska Annaler.

Malewicki, D. (2005). Member involvement in entrepreneur network organizations: The role of commitment and trust. Journal of Developmental Entrepreneurship.

Matuszak, G. (2007), *Enterprise 2.0: Fad or Future? The Business Role for Social Software Platforms, KPMG.*

McAfee, A. (2006a), *Enterprise 2.0, Version 2.0*.

McAfee, A. (2006b), *"Enterprise 2.0: the dawn of emergent collaboration",* MIT Sloan Management Review, Vol. 47 No. 3, pp. 21-8.

MessageLabs (2007a), *"Online social networking: the employer's dilemma".*

MessageLabs (2007b), *"Social networking: a brave new world or revolution from hell?"*.

Nahapiet, J. & Ghoshal, S. (1998). Social capital, intellectual capital and the organizational advantage. Academy of Management Review.

NETconsent Limited (2004), *Employee Internet Access: Effective Management of the Organisational Risk.*

O'Reilly, T. (2005), *"What isWeb 2.0? Design patterns and business models for the next generation of software".*

Orlikowski, W.J. (2002), *"Knowing in practice: enacting a collective capability in distributed organizing",* Organizational Science, Vol. 13 No. 3, pp. 249-73.

Richtel, M. (2008), *"Lost in e-mail, tech firms face self-made beast",* New York Times, 14 June.

Rosen, C. (2007), *"Virtual friendship and the new narcissism",* The New Atlantis, Vol. 17, Summer, pp. 15-31.

Shirky, C. (2008), *Here Comes Everybody. The Power of Organising without Organisations*, Penguin Books, New York, NY.

Smith, M. and Kollock, P. (1999), *Communities in Cyberspace*, Routledge, London.

Tapscott, D. and Williams, A.D. (2006), Wikinomics: *How Mass Collaboration Changes Everything*, Portfolio, New York, NY.