ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

DESARROLLO DEL PROTOTIPO DE UNA RED SOCIAL PARA LA EPN BASADO EN RICH INTERNET APPLICATIONS (RIAs)

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

GARZÓN VITERI DANIEL EDMUNDO
(daniel_garzon_infor@yahoo.com)

DIRECTOR: ING. ANDRÉS LARCO
(andres.larco@epn.edu.ec)

QUITO, OCTUBRE 2010

DECLARACIÓN

Yo, Daniel Edmundo Garzón Viteri, declaro bajo juramento que el trabajo aquí descrito es de mi autoría, que no ha sido previamente presentado para ningún grado o calificación profesional, y que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____

Daniel Edmundo Garzón Viteri

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Daniel Edmundo Garzón Viteri, bajo mi supervisión.

_____

Ing. Andrés Larco

DIRECTOR DE PROYECTO

CONTENIDO

INDICE DE GRÁFICOS

INDICE DE TABLAS

RESUMEN

El presente trabajo presenta el desarrollo del ***prototipo*** de Sitio de Red Social para la Escuela Politécnica Nacional (EPN), mediante la aplicación de Rich Internet Applications (RIAs) o *Aplicaciones de Internet Enriquecidas.*

En el primer capítulo se menciona la manera como las RIAs han ido modificando e implementando las interfaces web, volviéndolas más amigables al usuario, con tiempos de respuesta mas rápidos y con varias funcionalidades que solo eran propias de aplicaciones de escritorio. De igual manera, se introduce el concepto de *Red Social,* para entenderlo desde su contexto sociológico, para así poder estudiarlo como servicio en línea. Una vez determinados estos parámetros iniciales, establecemos los justificativos por los cuales la EPN necesita de un Sitio de Red Social, enfatizando en la necesidad de la integración de los miembros de la comunidad politécnica.

En el segundo capitulo, se definen los diferentes aspectos que contemplan el desarrollo del prototipo: la selección de herramientas, tecnología, metodología de desarrollo. Luego se plantean los objetivos a conseguir y las condiciones que el prototipo alcanzará en sus fases finales, así como sus proyecciones post-mortem y de mantenimiento.

En el tercer capítulo se muestran las pruebas realizadas sobre el prototipo en sus diferentes fases de desarrollo, así como una comparación de los resultados obtenidos con los objetivos propuestos y un análisis de costos para la implementación del prototipo en fases de producción.

En el cuarto capítulo se contemplan recomendaciones y conclusiones a partir de los resultados obtenidos en el desarrollo del prototipo.

PRESENTACIÓN

El presente trabajo muestra la posibilidad del desarrollo de aplicaciones web de índole social, utilizando Rich Internet Applications (RIAs) o *Aplicaciones de Internet Enriquecidas,* para cubrir necesidades de índole social para una población determinada, en este caso, los miembros de la comunidad politécnica.

Se establecen los lineamientos principales para el estudio y comprensión de las redes sociales como servicio en línea, así como la evolución del desarrollo de aplicaciones web a través del uso de tecnologías de RIAs, para luego escoger la que más se adapte a las necesidades de construcción de un ***prototipo*** de red social para la EPN.

Se muestran las distintas fases del desarrollo del prototipo del sitio de red social, así como las distintas pruebas llevadas a cabo en el prototipo, para luego definir las consideraciones necesarias para la posible implementación del sitio de red social, a nivel de producción.

CAPITULO 1: PLANTEAMIENTO DEL PROBLEMA

1.1 USO DE RIAS EN EL DESARROLLO DE APLICACIONES WEB.

Desde el advenimiento de la Internet como herramienta fundamental de los procesos existentes, el desarrollo de aplicaciones web se ha tornado en un estándar aceptado, de tal manera que pocos aspectos en la vida cotidiana se encuentran exentos de una interacción con entornos web, conexión o interacción en tiempo real.

Por tal motivo, las aplicaciones web tradicionales han tenido que evolucionar para convertirse en experiencias satisfactorias al usuario, que cada vez demanda menores tiempos de respuesta, contenidos de mejor calidad y cantidad, mayor interacción de los entornos web y ejecución de tareas más complejas con tan solo un par de clics, sobre todo en tareas que se han popularizado en los últimos años, como los procesos de comunicación entre los miembros de la comunidad global, que requiere estar conectada no solo por la necesidad social inherentemente humana, sino también como parte de los procesos de negocios en todo el mundo, en todas las áreas, todo el tiempo.

Ante esta creciente exigencia por parte de los usuarios, y por consiguiente, por el mercado (una aplicación poco eficiente implica pérdida de clientes, disminución de ventas y perdida de posicionamiento a nivel mercantil), los desarrolladores de aplicaciones web han visto en las Rich Internet Applications (RIAs), la alternativa adecuada para el desarrollo de aplicativos que presentan las ventajas de una aplicación tradicional de escritorio, con las facilidades de conexión y disposición de los entornos web.

1.1.1 COMPLEJIDADES DE LAS APLICACIONES WEB ACTUALES

A pesar de apenas una década de vida comercial que ha tenido la Internet y los entornos web, en relación a otras tecnologías referentes al desarrollo de aplicaciones, se ha convertido prácticamente en el entorno de desarrollo por

defecto, lo cual ha derivado en el desarrollo de nuevas tecnologías dentro de esta línea de desarrollo, y lo cual ha significado la creación de una expectativa en los usuarios con respecto a las nuevas aplicaciones web y de lo que estas pueden ser capaces para solventar altas demandas de mencionados usuarios.

Pero cumplir con las exigencias del usuario promedio, en aplicaciones web implicó determinar hasta donde las aplicaciones web tradicionales (en su mayoría, estaban capacitadas para tal fin, así como las complejidades que suponía el cumplimiento de los nuevos requerimientos de los usuarios de aplicaciones web.

Entre las complejidades principales a superar podemos encontrar:

- ***Complejidad de los procesos[1]:*** Esta complejidad se refiere a procesos que impliquen tarea de opción múltiple o que demanden una alta interacción. Un ejemplo de dicha complejidad son los formularios de registro: tradicionalmente, los formularios de registro requerían ser llenados en su totalidad, para que luego se efectúe un barrido de cada campo para validarlo y posteriormente informar al usuario de los cambios que se requerían para poder enviar el formulario. Si un formulario para solicitar un crédito en una institución bancaria, de unos 50 campos, requería ser llenado por una persona de entre 40 y 60 años (que independientemente de su formación, requiere de mayor esfuerzo para comprender los comportamientos de una tecnología joven como las aplicaciones web), tal tarea se convertía en un evento complejo; si tan solo se equivocara en dos o tres campos, tendría que localizar estos campos y volver a enviar el formulario, lo que constituye una pérdida de tiempo valioso, además de la frustración del usuario, que pensará dos veces en optar por esa institución bancaria, la misma que, como la mayoría de sus competidoras, se “actualizó” y transportó todos sus procesos a entornos web, supuestamente para “agilitar los procesos”, fenómeno que en la realidad de nuestro país suele tomar más tiempo del que normalmente se

[1] DUHL, Joshua.(2003).**Rich Internet Applications (Noviembre 2003).**IDC White Paper.

requiere.

- ***Complejidad en los datos***[2]*:*Complejidad que se ve relacionada en las múltiples interrelaciones que existen entre varios tipos de datos, y que las aplicaciones tradicionales son muy limitadas de tratar, ya que generalmente procesan información a través de datos con valores determinados y discretos: esto equivale a tener, como mencionamos en el punto anterior, formularios extensos que derivan en pérdida de tiempo y frustración al usuario; un manejo de datos de manera interrelacionada en la aplicación en necesaria.
- ***Complejidad de la configuración***[3]*:* Esta complejidad se refiere a la configuración de parámetros sobre los productos y como esta configuración afecta al resultado final producido en una aplicación de escritorio. En muchos programas de escritorio que manejan un volumen considerable de contenido visual, el cambio de parámetros no representa más allá de una programación de alteración de ciertos atributos para obtener este tipo de interacción sobre el resultado final generado en la aplicación (por ejemplo: una aplicación para usuarios de edad escolar, que consista en un lienzo con un dibujo a ser coloreado; la aplicación debe poseer una paleta de colores, para poder cambiar el color de cada una de las partes del dibujo según el pequeño usuario desee). En los sitios web tradicionales, esta funcionalidad es casi imposible de conseguir sin elementos de enriquecimiento gráfico (como por ejemplo, las hojas de estilo en cascada CSS), que los webmasters y desarrolladores web implementaron a sus proyectos.
- ***Complejidad en la escalabilidad y retroalimentación de datos***[4]*:* A nivel de datos, las aplicaciones tradicionales en entornos web se manejan de manera estática, con un número considerable de páginas refrescadas para los nuevos datos y baja interacción entre las interfaces y la lógica del negocio de las aplicaciones; por ejemplo, un filtro interactivo en un buscador que reaccione en tiempo real a medida que el usuario va depurando su búsqueda es extremadamente difícil de realizar en un

[2]DUHL, Joshua. (2003). **Rich Internet Applications (Noviembre 2003).**IDC White Paper.
[3]DUHL, Joshua. (2003). **Rich Internet Applications (Noviembre 2003).**IDC White Paper.
[4]DUHL, Joshua. (2003). **Rich Internet Applications (Noviembre 2003).**IDC White Paper.

ambiente web tradicional puro.

- ***Complejidad en la retroalimentación***[5]*:* En sitios web tradicionales, es prácticamente imposible obtener un bucle de retroalimentación que nos permita repasar los eventos realizados sobre los elementos de estos sitios: un formulario de registro en una página web pura tiene que volver a ser llenado en su totalidad si existen errores, independientemente de que los demás campos estén correctamente llenados.

Por las complejidades vistas anteriormente, y dada la creciente demanda de aplicaciones más robustas en entornos web, ciertos parámetros requerían implementarse en las nuevas generaciones de mencionados aplicativos:

- Las aplicaciones web debían ejecutarse de igual manera en Internet, independientemente del tipo de plataforma sobre la cual se ejecutaban.
- Debían responder de igual forma con velocidades de conexión altas o bajas.
- Para una mejor experiencia con el usuario, necesitaban interfaces con el alto índice de interacción.
- Integrarse con aplicaciones y sistemas antiguos.
- Permitir la adhesión de nuevos componentes a los ya existentes en la aplicación, así como la adaptación a los potenciales entornos web a desarrollarse en el futuro.[6]

1.1.2 LAS RIAs Y LAS APLICACIONES WEB

Ante las necesidades que requerían las aplicaciones web, en función de las demandas de los usuarios, surgieron nuevas herramientas que los webmasters y desarrolladores web utilizan hasta el día de hoy: los scripts.

Estos scripts (conjuntos de código invocado desde la página web y ejecutado en

[5]DUHL, Joshua. (2003). **Rich Internet Applications (Noviembre 2003).**IDC White Paper.
[6]DUHL, Joshua. (2003). **Rich Internet Applications (Noviembre 2003).**IDC White Paper.

la maquina cliente) permiten acciones y eventos que van desde efectos visuales en la interface (como los menús emergentes) hasta el control y validación de datos ingresados en un formulario HTML tradicional.[7]

Mientras más complejas las aplicaciones, más cercanas a un programa de escritorio de asemejaban, en tanto surgían tecnologías que revolucionarían los entornos web de una manera radical:

- La instauración de la conexión de Internet de banda ancha, desplazando la tradicional conexión *dial-up,* permitiendo descargas de contenidos de mayor tamaño y calidad en tiempos reducidos.[8]
- El mejoramiento en el poder de computo de los equipos informáticos, ya que tanto computadoras de escritorio, servidores, portátiles y, por tanto, todos los dispositivos habilitado para desplegar contenido web, han tenido un repunte significativo en este aspecto; tanto es así que ahora dispositivos tan pequeños como un celular ofrecen audio y video de calidad. Las RIAs utilizan esta potencialidad para enviar y recibir las aplicaciones web, creando un entorno ideal para el desarrollo de este tipo de aplicaciones.[9]
- El aparecimiento de tecnologías para páginas web paralelas a la plataforma Flash (las primeras versiones de AJAX y las versiones multiplataforma de Java Script, en el 2000), que se convertirían en sus principales competidoras de enriquecimiento visual de páginas web.

Tales implementaciones obligarían a los desarrolladores web a concebir un nuevo modelo de aplicaciones web: las RIAs propiamente dichas, siendo definidas como “un cruce entre las aplicaciones de escritorio tradicionales y aplicaciones web, tomando parte del proceso y poniéndolo en el lado del

[7]VARIOS AUTORES. **Rich Internet Applications: Design, Measurement, and Management Challenges.** Keynote White Paper. *http://www.keynote.com/company/resource_library/whitepapers.html*.
[8]NADA, Tom; HELWIG, Shawn. (2005)**Rich Internet Applications: Technical Comparison and Case Studies of AJAX, Flash and Java based RIA (16 de noviembredel 2005).** Best Practice Reports. UW E-Business Consortium.University of Wisconsin-Madison.
[9]NADA, Tom; HELWIG, Shawn. (2005) **Rich Internet Applications: Technical Comparison and Case Studies of AJAX, Flash and Java based RIA (16 de noviembredel 2005).** Best Practice Reports. UW E-Business Consortium.University of Wisconsin-Madison.

cliente y dejando el resto en la parte del servidor de aplicaciones"[10]. Según esta definición, las características que una RIA posee son:

- Su riqueza visual en comparación a aplicaciones web tradicionales, no solo en diseño sino también en comportamiento.
- Su ejecución en cualquier equipo con conexión a Internet, independientemente de su sistema operativo.
- Su facilidad de instalación y mantenimiento, así como su seguridad, por su arquitectura cliente-servidor.
- Y, por tanto, su capacidad de solventar las complejidades de las demandas de aplicaciones web actuales.

En el siguiente gráfico podemos apreciar el modelo de comportamiento de las RIAs, comprendiendo el contexto de uso y el entorno de la aplicación (cuadros verdes), las expectativas de usuario con respecto a la aplicación (cuadros amarillos) y como se comporta la RIA (cuadros azules).

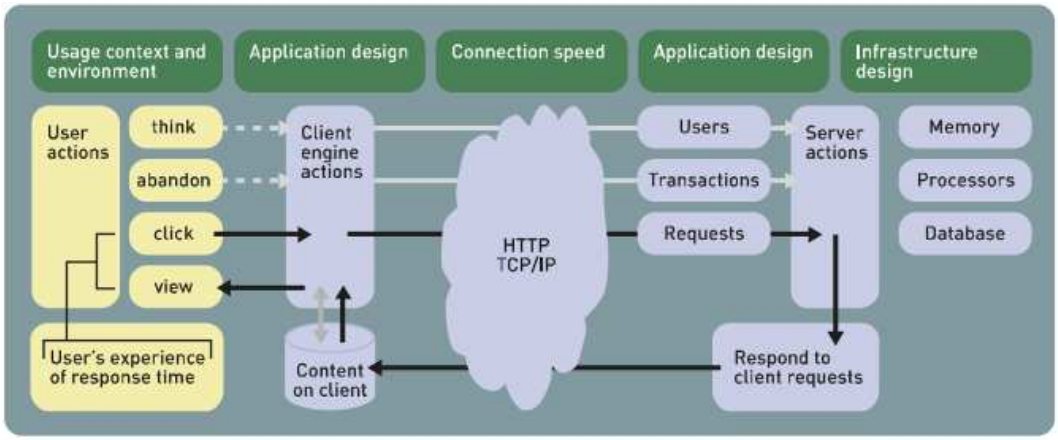

Figura 1.1 Modelo de comportamiento de las RIAs.[11]

En base a los lineamientos propuestos en el modelo de comportamiento de una RIA, deben considerarse ciertos aspectos importantes a la hora del diseño y manejo de este tipo de aplicaciones, teniendo como objetivos últimos la

[10]DUHL, Joshua. (2003). **Rich Internet Applications (Noviembre 2003).**IDC White Paper.
[11]VARIOS AUTORES. **Rich Internet Applications: Design, Measurement, and Management Challenges.** Keynote White Paper. *http://www.keynote.com/company/resource_library/whitepapers.html*.

satisfacción del usuario y el cumplimiento de los metas de los creadores de la aplicación. Estos aspectos son:

- ***Disponibilidad****:* Una RIA debe ser desarrollada en un framework que no comprometa, bajo ningún concepto, su ejecución en cualquier browser actualizado.
- ***Capacidad de respuesta****:* Una RIA debe manejar la complejidad de la comunicación entre el cliente (navegador) y el servidor de aplicaciones a través de una eficiente gestión de su código.
- ***Claridad****:* Una RIA debe ser simple y natural; es decir, debe ser predecible, sencilla de utilizar y consistente, en su fondo y forma.
- ***Utilidad****:* Una RIA debe siempre ofrecer lo que el usuario requiere, utilizando únicamente el browser en lo posible, y siempre implementar demandas y sugerencias del entorno en donde trabaja.

En el siguiente gráfico se puede comparar estas cualidades de las RIAs, desglosándolas en distintos ítems de carácter técnico, evaluando a tres de tecnologías de RIAs más utilizadas:

Table 1. Platform Comparisons (Source: UWEBC)

Good Average Poor

	AJAX	Macromedia Flash	Java
Graphical Richness	Average (Same as HTML)	Very Rich	Rich
Container/Engine Footprint	Very Light (browser built-in)	Light	Heavy
Application Download	Fast	Slow	Slow
Audio/Video Support	Poor (unless use ActiveX)	Excellent	OK
Consistency on Different Computing Environments	Varies	Very consistent	Relatively consistent
Server Requirements	None or very minimal (TIBCO General Interface)	Yes (Flex or Open Laszlo)	Yes or No (Nexaweb, Java Web Start)
Plug-in/Runtime Requirement on Client	No	Flash (Player)	Java Runtime (JRE)
Development Challenge	Very complex without tools such as TIBCO, and high skills required (JavaScript, CSS, XML, XSLT, DOM, ActiveX...)	Relatively easy with tools such as Flex or Open Laszlo (XML, DOM, JavaScript, Flash, ActionScript)	Relatively easy with tools such as Nexaweb (XML, JavaScript, Java)
Security Concerns	JavaScript codes are open to public. Everybody can see source codes if desire	Flash files (compressed binary) are created. Flash Player becomes a sandbox	Class/Jar compressed binary files are created. JVM (Java Runtime) becomes a sandbox
Cost	Custom Build - Free TIBCO - Unknown	Open Laszlo - Free Flex - $15,000 per CPU	Java Web Start - Free Nexaweb - Unknown

Figura 1.2 Cualidades de tecnologías RIA mas utilizadas.[12]

[12]NADA, Tom; HELWIG, Shawn. (2005) **Rich Internet Applications: Technical Comparison and Case Studies of AJAX, Flash and Java based RIA (16 de noviembredel 2005).** Best Practice Reports. UW E-Business Consortium.University of Wisconsin-Madison.

1.2 INTRODUCCIÓN A LAS REDES SOCIALES.

Al hablar de redes sociales, se debe poner en consideración que la expresión “Redes Sociales”, en los días actuales, tiene más de un significado:

- Las redes sociales pueden ser, desde un punto de vista meramente sociológico, estructuras sociales en donde sus miembros (también denominados nodos), establecen interrelaciones de distinta índole, enlazándose o conectándose entre si.
- Las redes sociales también pueden ser definidas como servicios en línea, soportadas por plataformas y las cuales reflejan o establecen comportamientos idénticos a los de las redes sociales definidas como estructura social.

Es evidente que para poder concebir una red social como un servicio en línea, es necesario comprender a la misma red social como un elemento social, ya que la aplicación en línea es una extensión sensorial de los comportamientos interdependientes entre los miembros de una red social sociológicamente definida.

1.2.1 LA RED SOCIAL, EN CONTEXTO SOCIOLÓGICO

Las redes sociales, como estructura social y sociológica, se conciben como el conjunto de individuos de una población (ya sean estas personas o instituciones), a los cuales se les denomina nodos, y que se encuentran conectados por diferentes formas de relaciones: afinidad, parentesco, amistad, etc.

Tradicionalmente, el estudio social habría sido realizado de forma empírica, tomando muestras poblacionales (individuos elegidos con o sin un patrón o consideración específicos) y estudiándolas individualmente, alejándolos de la parte social como tal, ya que se despreciaba todo efecto o influencia que se ejerciera sobre ellas.

No obstante, fue necesario determinar una estructura para determinar las relaciones posibles que puedan manifestarse entre los miembros de una determinada población: la red social.

La aproximación del concepto de red social se basa en la noción de que, el determinar relaciones sociales en los cuales los actores sociales están inmiscuidos, tiene consecuencias sobre todos y cada uno de ellos.[13]
Por tanto, no podemos concebir una red social sin necesariamente estudiar a los miembros o nodos de la misma junto (e incluso a partir) de las relaciones que poseen entre ellos, y de cómo este tipo de relaciones afectan a cada nodo.

A lo largo de la historia, el hombre siempre ha sido inherentemente un ente social, ya sea formando vínculos genealógicos o expandiendo su círculo de relación a partir de sus congéneres, coidearios, familiares, entre otros, de manera que determinar cual sería la primera red social creada. Lo que si se puede mencionar con certeza es que el estudio de las redes sociales como estructura social, se remonta al siglo XVI, época en la cual Augusto Comte plantearía la primera perspectiva estructural de la vida social, como parte de su intento de acuñar ala sociología como ciencia.[14]

Posteriormente, a finales del siglo XVII el sociólogo francés Émile Durkheim y su colega alemán Ferdinand Tönnies darían las primeras nociones de red social basados en los estamentos de Comte: el primero definió a una sociedad moderna como una “unificación orgánica”, que se caracterizaba por el apoyo mutuo entre individuos de una población sin descuidar su naturaleza de personas con características únicas; el segundo, determinó que la existencia de un grupo social es posible en tanto existan enlaces sociales directos y personales entre los individuos de este grupo, compartiendo creencias o pensamientos que desemboquen en enlaces sociales efectivos.[15]

---

[13]FREEMAN, Linton C.(2004). **The Development of Social Network Analysis: A study in the sociology science.** Empirical Press. Vancouver, British Columbia. Canada.
[14]FREEMAN, Linton C.(2004). **The Development of Social Network Analysis: A study in the sociology science.** Empirical Press. Vancouver, British Columbia. Canada.
[15]ANONIMO.(2011).**History of Social Networking (02 de febrero de 2011).** Obtenido de: *http://www.bighistory.net/history-of-social-networking/*.

Más tarde se harían varios estudios formales de las redes sociales, lo que daría lugar a lo que hoy se conoce como el “Análisis de la Redes Sociales”, asignatura de corte sociológico que permite conocer, entender e incluso predecir el comportamiento de una red social y de sus componentes o nodos. Cabe destacar que para el estudio de esta asignatura se utiliza teoría de grafos para establecer las relaciones entre nodos y poder tener una apreciación visual del comportamiento de los elementos de una red social.

1.2.2 LA RED SOCIAL, EL SERVICIO EN LÍNEA

La constante necesidad del hombre y la sociedad de adaptarse a los constantes y rápidos cambios tecnológicos, ha llevado también a que su entorno social tienda a “digitalizarse”, es decir, a usar tecnología para mantener sus vínculos sociales para complementar, e incluso sustituir, la presencia física de las interacciones sociales de los miembros de una red social sociológicamente definida.

No obstante, para hablar de la red social como servicio en línea, es necesario mencionar que su origen se remonta a los anales de la comunicación mediante el uso de computadoras, y como estos servicios de comunicación han ido evolucionando incorporando funcionalidades que facilitan la comunicación de los miembros de una red social, a favor de su crecimiento como comunidad, eliminando barreras como la distancia, y en un sentido literal, haciendo posible la comunicación de sus miembros alrededor del mundo. Este es el verdadero propósito del “Software Social”.

El software social, en su forma más simple, se refiere a cualquier sistema que permita la interacción entre usuarios, así como la compartición de datos e información, es decir, un sistema que permita el trabajo colaborativo de los miembros de un grupo. Desde este punto de vista, cualquier herramienta de comunicación entre dos o más maquinas y/o permitan compartir información bajo un determinado protocolo, podrían considerarse software social, sean estos: comandos de terminal, salas de chat, foros, blogs, wikis, servicios de mensajería

instantánea, servidores de archivos, etc.

Sin embargo, el software social va mas allá, ya que desde los primeros experimentos de comunicación de ARPANet en los 60s hasta la acuñación de términos como “Software Social” y “Comunidad Virtual” en los primeros años de este milenio, su objetivo primario ha sido no solo la comunicación de los miembros de un grupo, sino la posibilidad de un trabajo común a favor de un objetivo específico compartido por los miembros de ese grupo.[16]

Por lo tanto, estas necesidades de comunicación, expansión y trabajo grupal de los diferentes grupos sociales, ha resultado en una constante y acelerada evolución de los diferentes servicios de comunicación, como las ya obsoletas salas de chat, sustituidas posteriormente por servicios de mensajería instantánea; por otro lado aparecieron los blogs y las wikis, como otra representación digital de las diferentes redes y comunidades sociales existentes alrededor del mundo, e incluso la herramienta más utilizada en términos informáticos comunicacionales: el correo electrónico, que ha ido cambiando constantemente su estructura, adhiriendo funcionalidades y tornándose más versátil.

Y es precisamente aquí, en la evolución del correo electrónico, donde las redes sociales, como servicio en línea, va tomando forma, a partir de la creación del concepto del “perfil de usuario”, un conjunto de datos que definen a un sujeto con gustos, relaciones y particularidades propias de un ente social. SixDegrees.com fue el primer intento de red social como servicio en línea, permitiendo a sus usuarios publicar sus perfiles, mientras que por otro lado, ICQ y AIM permitían crear listas de amigos como parte de su servicio de mensajería instantánea. A partir de estas dos singulares prestaciones, posteriormente surgieron los primeros sitios de redes sociales propiamente dichos, como MySpace.com y Friendster, entre los más destacados.[17]

---

[16]ALLEN, Christopher (2004). **Tracing the Evolution of Social Software (13 de Octubre del 2004).**Obtenido de: *http://www.lifewithalacrity.com/2004/10/tracing_the_evo.html*.

[17]BOYD, Danah m.; ELLISON, Nicole B.(2007). **Sitios de Redes Sociales: Definición, Historia y Conocimiento.** Journal of Computer-MediatedCommunication, 13(1), (traducción).

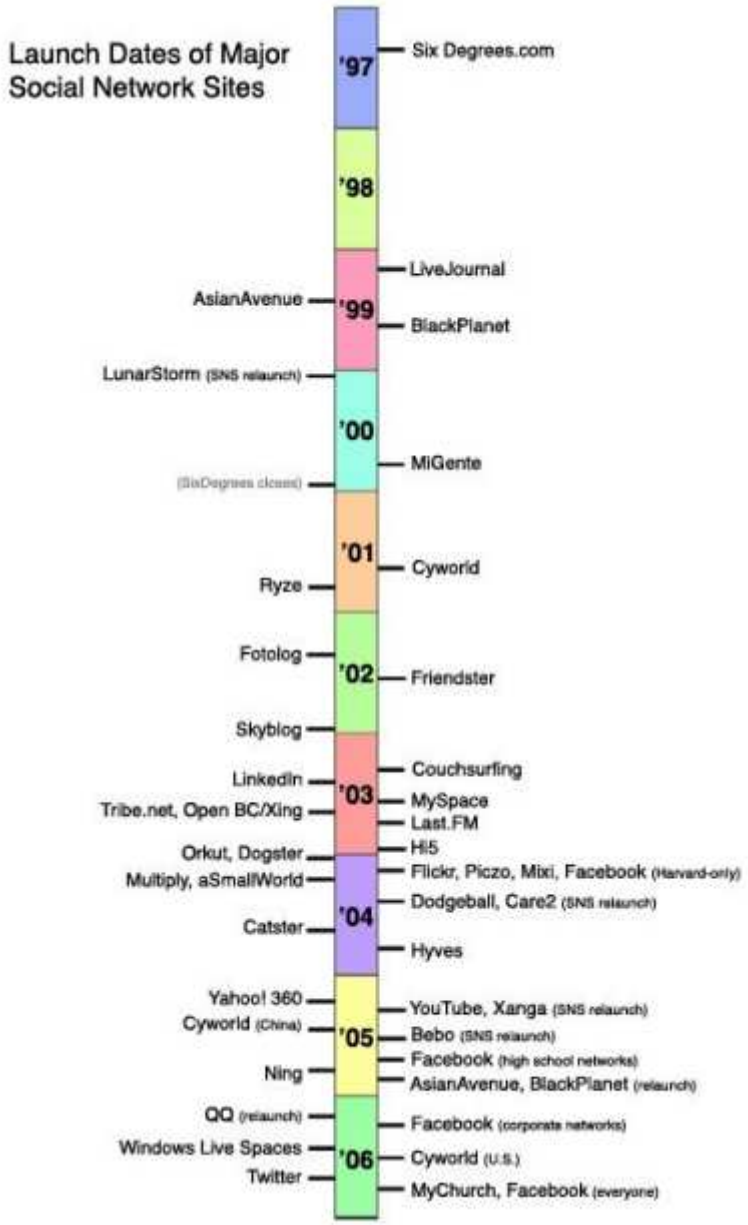

Figura 1.3Línea histórica de los sitios de redes sociales más importantes.[18]

Hoy en día, de entre los usuarios de Internet entre los 12 y 35 años en su mayoría, tiene un perfil de por lo menos un sitio de red social, ya sea por la facilidad de comunicación con los miembros de su círculo social, por necesidad laboral o por moda. No obstante, desde el surgimiento de los primeros sitios de redes sociales, estos sitios representaron la revolución de las comunicaciones a nivel global, integrando las tecnologías ya existentes en una sola aplicación, cumpliendo así con el objetivo del software social.

[18]BOYD, Danah m.; ELLISON, Nicole B.(2007). **Sitios de Redes Sociales: Definición, Historia y Conocimiento.** Journal of Computer-Mediated Communication, 13(1), (traducción).

1.3 IMPACTO DE LAS REDES SOCIALES EN LA COMUNICACIÓN

Las redes sociales, dada su popularidad, se han convertido en una de las razones principales por las cuales la gente utiliza Internet. Ciertamente el uso más común de las redes sociales es, evidentemente, reproducir la red o entorno social de los usuarios; no obstante, hoy en día las redes sociales han proliferado de tal manera que no solo sirven para reproducir nuestras relaciones de manera virtual: existen varias redes sociales que apuntan a distintos tipos de usuarios o entornos. Por ejemplo: mientras Facebook es una de las redes sociales más famosas, cuyo objetivo es más bien un público joven que desea reproducir/expandir su círculo social, también existen redes sociales de tipo corporativo, como LinkedIn, donde se propone la promoción de las hojas de vida de sus usuarios, así como búsqueda laboral, publicación de eventos de tipo corporativo, entre otras funcionalidades.[19]

Asimismo, las empresas apuestan por promocionarse en los sitios de redes sociales, debido a la masiva concurrencia que estos poseen; contrario a lo que se podría pensar, las empresas permiten que sus empleados demuestren su creatividad a favor del posicionamiento de la marca a través de las redes sociales generalmente usadas para usuarios individuales, como Facebook, Hi5 o MySpace, para a través de sus servicios de mensajería captar la atención de potenciales clientes, publicando promociones, nuevos productos o servicios, etc.[20]

Por otra parte, un sector que ha visto un nicho de mercado en las redes sociales ha sido el de la telefonía móvil, la cual permite una mayor venta de smartphones y demás gadgets de esta línea con paquetes atractivos que permiten la conexión a los sitios de redes sociales, además de promociones para gestión de comunicaciones y datos en dichos dispositivos. De igual manera, la integración de aplicaciones propias de estos dispositivos sobre las plataformas en las que se

[19]BORTNIK, Sebastián.(2010). **Dudas y certezas sobre las redes sociales en las Empresas (02 de agosto del 2010).** ESET Latinoamérica. Buenos Aires.
[20]PORRUA GARCÍA, Manuel.(2009-2010). **Impacto de las redes sociales(Diciembre 2009/Enero 2010).**Dintel.

construyen los sitios de redes sociales, presuponen en un futuro una estandarización en las tecnologías a usar en el desarrollo para los teléfonos móviles.

Los sitios de redes sociales han significado uno de los logros más importantes de la Web 2.0, y por tanto, uno de los capitales representativos de varias empresas, las cuales han visto en las redes sociales, un lugar privilegiado para promocionarse, como el caso de Bebo.com, adquirida por AOL. Otras aplicaciones como Youtube, comprada por Google, o Skype, comprada por E-Bay, también han gozado del boom de la web 2.0.

Sin embargo, un impacto que muchos asocian como negativo, es que las redes sociales, en ciertos aspectos, ha disminuido la relación “cara a cara”, debido a que muchos de los usuarios poseen perfiles en las redes sociales por moda, sin poner mucho énfasis en el verdadero objetivo de los sitios de redes sociales, como software social que, como ya se ha dicho antes, se refiere al trabajo colaborativo a través de la interacción de los nodos de una red social.

1.4 JUSTIFICACIÓN DEL DESARROLLO DE UNA RED SOCIAL PARA LA EPN.

La comunicación entre los distintos departamentos y facultades de la Escuela Politécnica Nacional se encuentra en un estado de inercia, a tal punto que muchas de las actividades que existen en una facultad determinada, son prácticamente ignorados por el resto de facultades y departamentos de la EPN, lo que provoca que todos los componentes estructurales de la EPN se encuentren aislados entre si, impidiendo que se conozca de la múltiple actividad que existe en la institución, así como la integración entre facultades, no solo a nivel social y humano, sino además para el desarrollo de nuevos proyectos de investigación multidisciplinarios, a diferencia de como se hace en universidades del extranjero.

En la EPN, existen ocho facultades, de las cuales solo cinco poseen sitios web oficiales; en lo referente a los 19 departamentos que existen en la comunidad

politécnica, si bien se encuentran mencionados en la web oficial de la universidad junto con un correo por departamento bajo el servidor de correo de la EPN para contactos, más una lista de correos bajo mencionado servidor y teléfonos de los jefes de mencionados departamentos, apenas uno tiene un sitio web oficial, el departamento de Automatización y Control Industrial, mientras que los demás poseen páginas informativas dentro de los sitios oficiales de aquellas facultades que poseen un portal web: inclusive, varios departamentos no tienen la información levantada o actualizada, como es el caso puntual de los departamentos de la Facultad de Ingeniería Eléctrica o Electrónica, sin existir una plataforma tecnológica que pueda integrarla a través de recursos de comunicación tales como: carteleras, avisos, perfiles personales y grupales , informes de actividades, para el conocimiento general de quienes integramos la EPN. Cabe destacar que la nueva versión del sitio web oficial de la EPN recopila información departamental con la que antes no se disponía; pero se requiere que esta información también sea utilizada a favor de los miembros de la comunidad politécnica de manera activa, para un mayor conocimiento de los elementos que la conforman y los servicios que prestan, así como una adecuada explotación del servidor de correos de la EPN, para que todos sus miembros tengan a disposición este servicio, ya sean estudiantes, profesores o trabajadores dentro de la comunidad politécnica.

Es por eso que se plantea en el siguiente proyecto la construcción de un prototipo de sitio de red social para la EPN, que, persiguiendo el verdadero significado del software social previamente estudiado, permita el trabajo común de los miembros de la comunidad politécnica para juntos lograr varios objetivos como:

- La formación de equipos de trabajo multidisciplinario,
- El conocimiento y uso de las diferentes dependencias que existen dentro de la comunidad politécnica, y
- La integración de los miembros de la comunidad politécnica en un entorno común.

CAPITULO 2: DESARROLLO DEL PROTOTIPO DE LA RED SOCIAL PARA LA EPN

Para el desarrollo del prototipo de red social para la EPN, se ha seleccionado la metodología de desarrollo XP (eXtreme Programming), ya que es un proyecto pequeño y que requiere de un contacto más cercano con los usuarios, debido a su naturaleza social.

2.1 EXPLORACIÓN

Tomando en cuenta que el prototipo de red social propuesto en el siguiente proyecto tiene fines didácticos, se considerará como espacio de trabajo la Facultad de Ingeniería en Sistemas (FIS), y las historias de usuarios se determinarán a partir de escenarios en donde se puedan evaluar:

- Los niveles de comunicación entre las diferentes entidades de la misma (tipo 1),
- Los criterios generales de una aplicación web enriquecida visualmente para su correcta funcionalidad y desenvolvimiento (tipo 2) y,
- Los factores que determinan la selección de un sitio de red social (tipo 3).

Al finalizar el análisis de las historias de usuario y seleccionando las historias adecuadas para el prototipo, alojadas en el anexo 1, el sitio de red social que desarrollará debe tener las siguientes características:

- Debe ser rápido en sus tiempos de carga y en su ejecución.
- Debe ser personalizable en los aspectos individuales del usuario (ej: su perfil).
- Debe ser lo suficientemente interactivo para justificar su enriquecimiento visual, ofreciendo a una interfaz fácil de utilizar, amigable y en algún grado, conocida para el usuario.
- Esta interfaz, a su vez, debe estar diseñada para poder establecer comunicaciones con los miembros del espacio de trabajo (para el caso particular del presente prototipo, la FIS).

Una vez determinadas las características del sitio de red social, se procederá a definir el alcance del proyecto. Por tratarse de un prototipo, lo que se pretende conseguir es demostrar la posibilidad de implementar una plataforma de comunicación enriquecida visualmente, que permita una comunicación más fluida y activa entre los miembros del espacio de trabajo considerado, es decir, la FIS. Por lo tanto, lo que el prototipo será capaz de hacer, en función de las características que debe tener el sitio de red social, será:

- **Utilizar las prestaciones de una tecnología de RIAs para la construcción de la interfaz del prototipo:** Se ha seleccionado Ext JS, de la empresa Sencha, que es una tecnología reciente en RIAs, derivada de la tecnología tradicional JavaScript y AJAX, que ha sido escogida por los similares valores que comparte con la metodología de desarrollo usada en el presente proyecto: es simple de implementar, permite pruebas in situ en los browsers para corrección y pruebas, y por ser reciente, representa un desafío a nivel de desarrollo web.
- **Manejar elementos propios de una plataforma de red social:** Para esta parte del proyecto, se definirá las tecnologías apropiadas para el envío recepción de datos. Se considera en primera instancia utilizar jsp (Java Server Pages, tecnología JAVA que permite mostrar contenido dinámico en una web) que interactúen con una base de datos, pero se tiene en cuenta la posibilidad de utilizar servlets (programas JAVA que corren dentro de un servidor), proponiendo así una arquitectura MVC para el prototipo.

El prototipo, por su naturaleza, presenta las siguientes limitaciones:

- No funcionará bajo condiciones reales de producción. Es decir, solo funcionará bajo condiciones reales que se alcancen en las fases finales de desarrollo.
- No utilizará equipos pertenecientes a la FIS. Por ser un prototipo, se ha optado por el uso de servidores virtuales de ser necesario.
- No utilizará datos reales de la FIS. Si bien los criterios de seguridad en la

plataforma serán adecuadamente levantados, no se utilizará información sensible de ninguno de los miembros de la FIS. El prototipo tiene como objetivo demostrar funcionalidad bajo escenarios controlados.

2.2 PLANIFICACIÓN DE LA ENTREGA

Para la planificación de las entregas se ha considerado adecuado entregar tres versiones del prototipo, por el tamaño del mismo y por el tiempo total que se ha definido para el proyecto:

- Una primera entrega o *spike,* que contemplará los aspectos básicos del sitio de red social y el correcto despliegue de las interfaces de usuario. Siendo esta la entrega base para todo el sitio, cubriendo las historias de usuario de tipo 2 y 3.
- Una segunda entrega o *spike*, en donde se implementará las funcionalidades de la red social a nivel de prototipo, las cuales comprenden: la gestión de usuarios, la gestión de archivos, la interrelación entre usuarios miembros de un mismo grupo dentro de la red, y todas las interacciones de la interface de la red con un modelo de datos prototipo, probando diferentes tecnologías sobre todo en la capa de control, cubriendo las historias de usuario de tipo 1 y 3.
- Una entrega final o *spike* definitivo, en donde se definirá el modelo de datos final, los perfiles de los usuarios a utilizar la red, la ubicación definitiva de los archivos de usuario en el servidor y el control de comportamiento del sitio de red social ante varios escenarios de uso. En este punto se cubrirá las historias de usuario que hasta ese momento no se hayan podido cumplir o que por alguna razón hayan quedado pendientes.

2.3 ITERACIONES

Para la planificación de las iteraciones en cada una de las entregas se ha considerado generar una aplicación funcional para evaluar si se cumplen los requerimientos especificados en las historias de los usuarios, junto con un reporte de pruebas y errores o bugs que se presenten antes de la presentación del

respectivo spike.

2.4 PRODUCCIÓN

Una vez analizadas y clasificadas las historias de usuario que se utilizarán para la construcción del prototipo, se procedió a elegir las herramientas con las cuales se implementaran los spikes:

- Se seleccionó a Eclipse Galileo como marco de trabajo, debido a su fácil integración con varias arquitecturas que utilizan JAVA y sus variantes como lenguaje de programación, así como con motores de bases de datos y servidores.
- Para la parte del servidor, se seleccionó Apache Tomcat, por su facilidad de implementación de servlets y jsp, lo que deja la posibilidad abierta de utilizar una de las dos opciones para la capa de control de datos.
- Como motor de base de datos se seleccionó MySql, por su simplicidad y por disponer de varias herramientas gráficas que permiten la gestión de las bases o tablas creadas. Además, MySql es usado para Facebook, ya que este sitio utiliza una arquitectura LAMP (Linux, Apache, MySql y PHP) para su ejecución.

Si bien la estructura de la red social a implementarse en el prototipo no está aun no está del todo definida en el inicio de la fase de desarrollo, bien puede plantearse una primera arquitectura que tendrá la aplicación propiamente dicha:

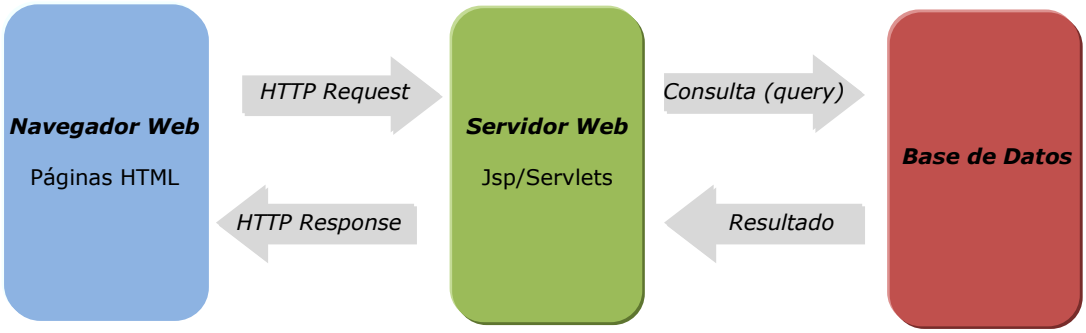

Figura 2.1 Primera versión de la estructura de la red social

En el gráfico anterior se puede apreciar que, sin importar si se utiliza JSP o servlets, el prototipo manejará objetos de sesión para la gestión de usuarios.

2.4.1 PRIMER SPIKE

Para la realización del primer spike, se tomo como punto de partida los ejercicios propuestos por el sitio oficial de ExtJS (www.sencha.com). En este sitio, existe una sección de tutoriales, en donde se revisan los conceptos esenciales para el uso de la librería ExtJS[21], mediante el uso de ejemplos sencillos que manipulan objetos HTML con la librería seleccionada. Por lo tanto, la primera iteración fue la codificación y ejecución del ejemplo en el marco de trabajo que seleccionamos para la aplicación:

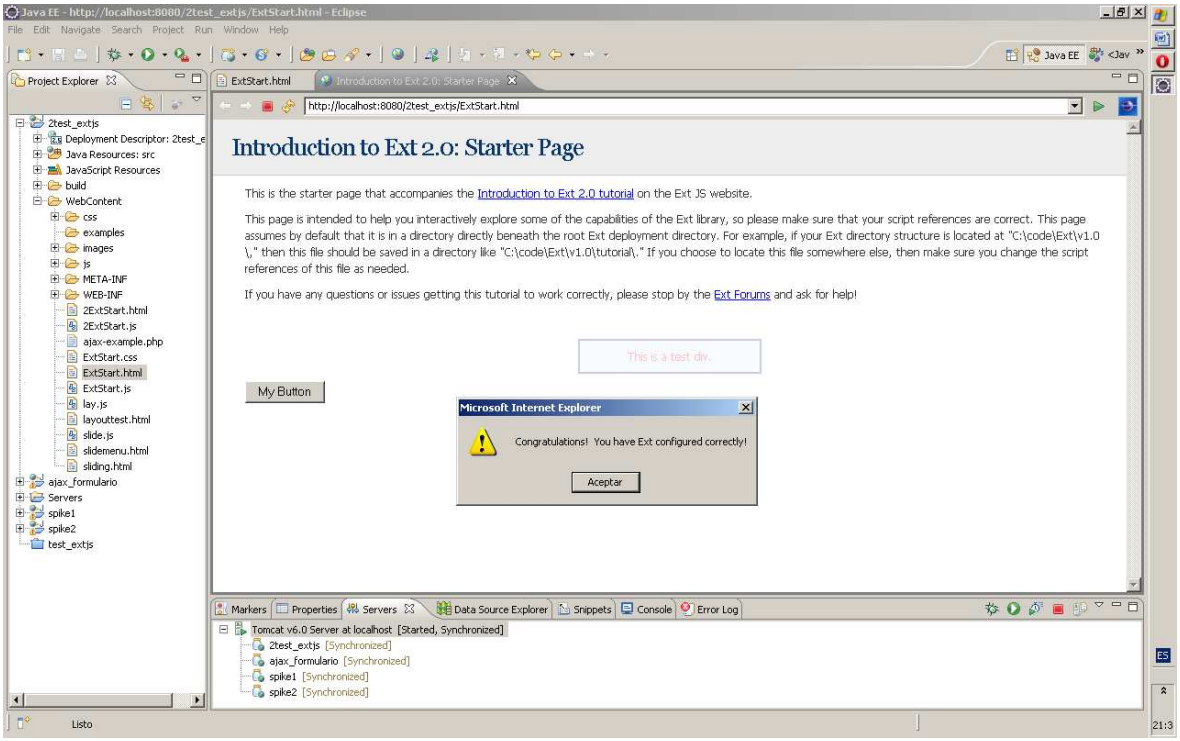

Figura 2.2 Captura de pantalla de un ejemplo de ExtJS correctamente implementado.

La librería ExtJS, al ser una derivación del Javascript tradicional, funciona exactamente de la misma manera; es decir, la librería es invocada en el código HTML a través de las etiquetas `<script>`, entre las cuales puede colocarse

[21] MOESKAU, Bryan. (2007). **ExtJs Essentials.** Obtenido de: *httphttp://www.sencha.com/learn/ext-js-essentials/*.

código referente a ExtJS o enlazar un archivo que contenga código ExtJS.

La estructura de un script de ExtJS recuerda mucho a otras librerías basadas en Javascript, como Jquery. Sin embargo, ExtJS tiene una sintaxis determinada, con elementos propios para construir las aplicaciones utilizando la librería. A continuación se muestra un ejemplo de código con las sintaxis de ExtJS para el primer ejemplo implementado en este spike, con algunas de las funciones más representativas de la librería:

```
Ext.onReady(function() {

	varmyDiv = Ext.get('myDiv');

	myDiv.highlight();	// The element
yellow then fade back
	myDiv.addClass('red');
ExtStart.css)
	myDiv.center();		// Center the e
	myDiv.setOpacity(.25);	// Make the ele

	// Note: For the purposes of following along with the tutorial, all
	// new code should be placed inside this method.  Delete the following
	// line after you have verified that Ext is installed correctly.

	alert("Congratulations!  You have Ext configured correctly!");

Ext.get('myButton').on('click', function(){
alert("Elé.... me hiciste click");
});

	varparagraphClicked = function(e) {
		var paragraph = Ext.get(e.target);
	paragraph.highlight();

	Ext.MessageBox.show({
	title: 'Parrafoclickeado',
	msg: paragraph.dom.innerHTML,
	width:400,
	buttons: Ext.MessageBox.OK,
	animEl: paragraph
		});
	}
	Ext.select('p').on('click', paragraphClicked);


});
```

Método Inicial para referenciar elementos de ExtJS

Metodo "get" para tomar elementos HTML

Métodos, configuraciones y propiedades aplicadas a la variable definida en el método "get".

Mensaje de alerta con estilo predeterminado por el browser.

Función para un evento "click" sobre un elemento.

Mensaje de alerta con estilo de la librería.

Una vez que el ejemplo fue codificado correctamente, se procedió a implementar un ejemplo propio utilizando los elementos que se muestran en el ejemplo anterior, además de la implementación de nuevos elementos que la librería ExtJS posee, y que se encuentran detallados en su documentación oficial; la idea del ejemplo fue ya plantear un modelo que la interface del sitio va a tener, utilizando menús desplegables:

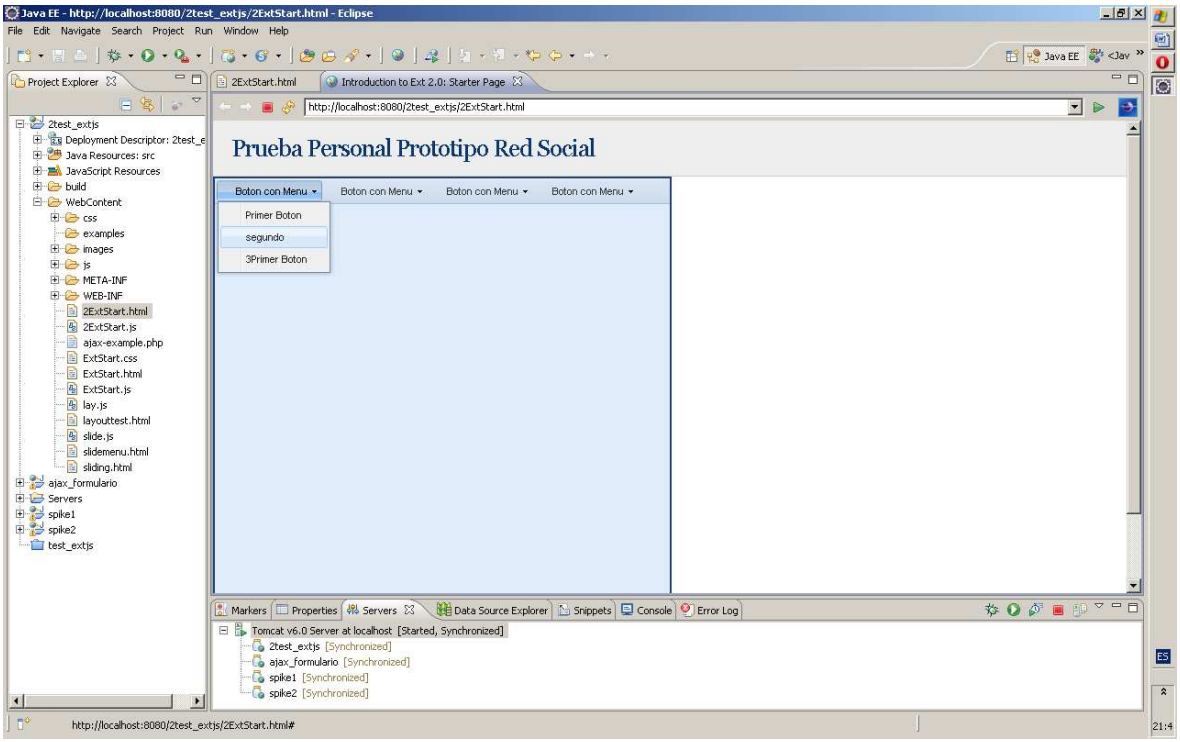

Figura 2.3 Captura de pantalla de un ejemplo propio de ExtJS para la interface del sitio.

Con un ejemplo real a partir de cero ejecutado correctamente, se procedió con la construcción de algunas páginas que ya formarían, en un inicio, parte del prototipo: en este caso, formularios de registro e ingreso de usuarios. Al mismo tiempo, y por tratarse de un formulario, se planteó el uso de servlets sencillos que permitan manejar los datos al enviar el formulario, junto con algunas jsp que controlarían la gestión de usuarios a través de objetos de sesión, así como el manejo de redirecciones de páginas entre los formularios y la aplicación per se. Se creó una base de datos genérica para probar el envío y recepción de datos tanto en la base de datos como en la interface. Como en este punto se definieron tanto servlets como modelo de datos básico, también se desarrollaron clases

JAVA que realizan las consultas a la base de datos, así como también definen los objetos de los distintos tipos de datos que la aplicación va a manejar. A continuación se presenta una clase java que define al objeto “usuario” (en este spike), cuya sintaxis se asemejará para el resto de tipos de datos que manejará el prototipo:

```
package vo;

public class usuario {

private String nombre;
private String apellido;

public usuario(){

  }

 //setters and getters de los atributos del usuario

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

publicvoidSetApellido(String apellido){
this.apellido=apellido;
}

public String GetNombre(){
return this.nombre;
}

public String GetApellido(){
return this.apellido;
  }

}
```

Para el manejo de sesiones, en este primer spike se utilizaron jsp para la captura de datos de sesión a partir del tipo de dato “usuario” previamente definido, y el objeto de sesión *HTTPsession.* He aquí el código que levanta la sesión a partir de una consulta en la base de datos para verificar la existencia para permitirle el acceso a la aplicación:

```
<%@page contentType="text/html" pageEncoding="UTF-8" import="java.sql.*" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<%@page import="java.util.*" session="true" %>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Procesar</title>
</head>
<body>
<%
    String user="root";
    String clave="shinjikun";
    String ruta="jdbc:mysql://localhost:3306/ext";
    Connection conexion=null;
    Statement Sentencias = null;
ResultSettabla = null;
Class.forName("com.mysql.jdbc.Driver").newInstance();
conexion =DriverManager.getConnection(ruta,user,clave);
Sentencias = conexion.createStatement();


int b=0;//variable booleana
Stringnm=null; //para tomar el usuario de la base de datos
Stringap=null; // para tomar el pass de la base de datos

//recogemos lo que viene del formulario
String nombre= request.getParameter("nombre");
String apellido = request.getParameter("apellido");

//la consulta SQL
        tabla = Sentencias.executeQuery("selectnombre,apellidofrom persona where nombre='"+nombre+"' "  );
try{

while(tabla.next())//desplegamos los resultados
{
              nm= tabla.getString("nombre");
ap= tabla.getString("apellido");
              if(nm.equals(nombre) &&ap.equals(apellido))
{
                              b=1;//si se cumple es 1
                    }
else
{
                              b=0; //sino es 0
                    }

}

//luego comparamos si es uno es usuario existente y bien validado sino error
if (b==1){
        String nom = new String (nombre);
        String app = new String (apellido);
HttpSession op = request.getSession();
op.putValue("nombre",nom);
op.putValue("apellido",app);
out.println("<script>alert('UsuarioIdentificadoCorrectamente')</script>");
out.println("<script>alert('UsuarioIdentificadoCorrectamente')</script>");
out.println("<meta http-equiv='refresh' content='0;url=ind.html'");
        }
        else{

out.println("<script>alert('DatosErroneos  ')</script>");
out.println("<meta http-equiv='refresh' content='0;url=login.html'");
        }

}
catch(Exception e){e.printStackTrace();}

%>

</body>
</html>
```

Directiva JSP para establecer sesión

Inicio de Scriplet JSP para consulta y validación de usuario

Por último, se construyeron los formularios utilizando la librería ExtJS.

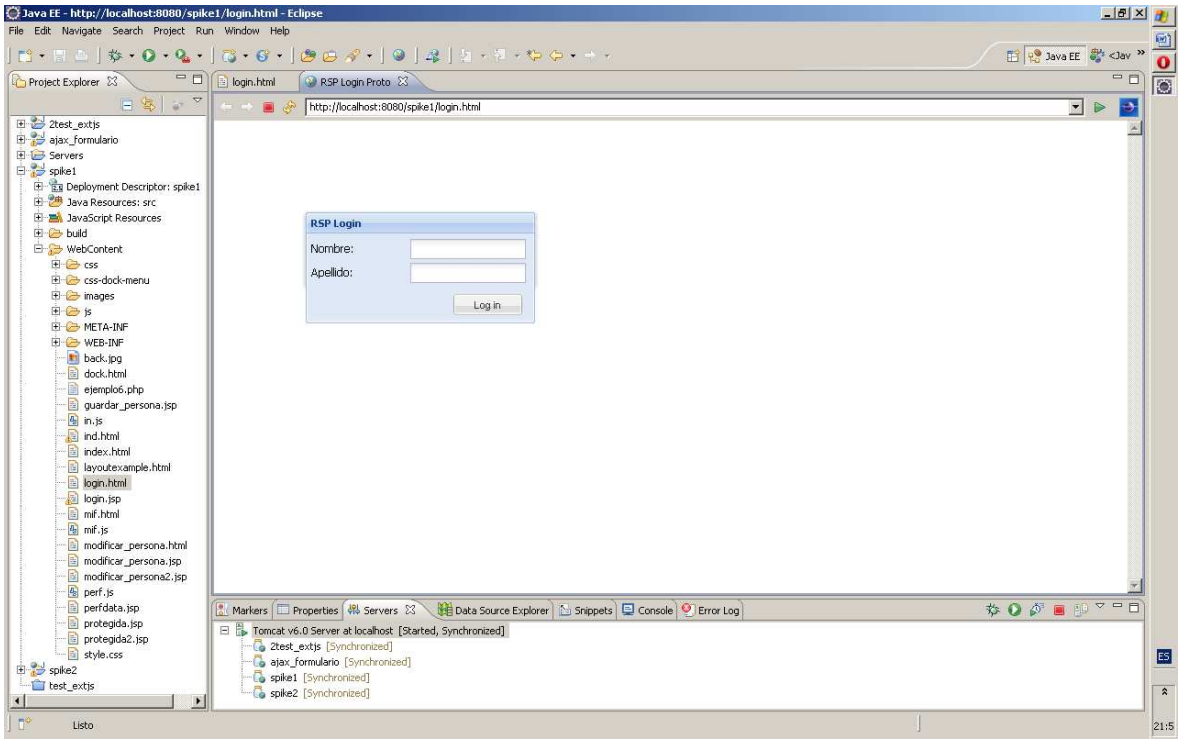

Figura 2.4 Captura de pantalla del formulario de login del primer spike.

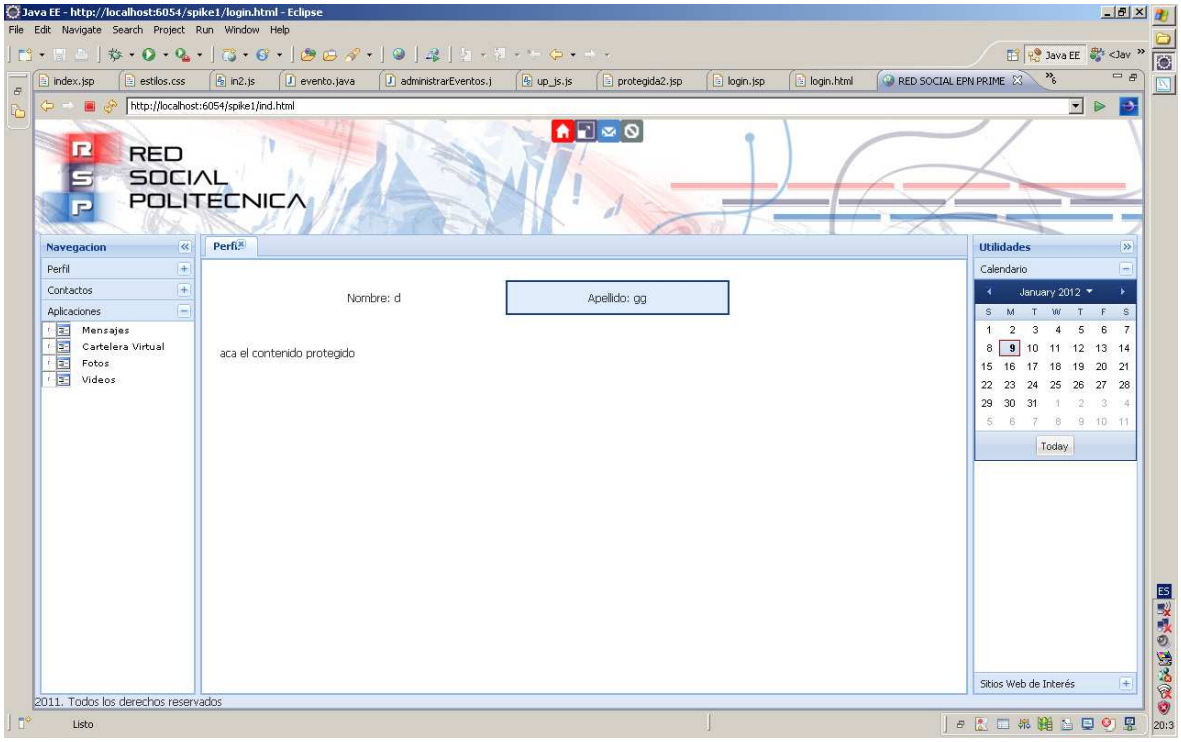

Figura 2.5 Captura de pantalla de la interface del prototipo en el primer spike, con usuario referenciado desde la base de datos genérica.

2.4.2 SEGUNDO SPIKE

En la elaboración del segundo spike, se consideraron las siguientes tareas:

1. ***Establecer las funcionalidades básicas que el prototipo de red social tendrá, en función de las necesidades sociales de la comunidad politécnica, establecidos en los objetivos para la construcción del prototipo de red social.-*** En esta tarea se tomó como sujetos de estudio dos sitios de redes sociales: Google+ y Facebook, ya que comparten muchas similitudes tanto estéticas como de comportamiento, y, en el caso del segundo sitio, goza de aceptación a nivel mundial, por su constante renovación en prestaciones.

   A partir de breves interacciones con los sujetos de estudio, y analizando los objetivos que justifican la construcción del prototipo de red social, así como las características de los sitios de redes sociales mencionados, las funcionalidades que tendrá el prototipo serán las siguientes:

   - Manejo de perfil de usuario: el usuario estará en capacidad de editar su perfil, cambiar la apariencia de su interface (para el prototipo, con temas predeterminados), e incluso de dar de baja su perfil.
   - Gestión de contactos: en donde el usuario podrá buscar contactos a partir de su nombre y listar los contactos que tiene dentro de su red social, así como agregar contactos alojados en la base de datos del sitio.
   - Aplicaciones: el usuario podrá tener acceso a varias funcionalidades que faciliten la comunicación con los miembros de su red social, tales como: publicación de novedades y eventos, gestión de fotografías e imágenes y mensajería.

2. ***Definir módulos individuales para cada una de las funcionalidades del prototipo de red social, y probarlas una a una, sin ninguna interacción entre las mismas, salvo en los casos en que sea estrictamente necesario.-*** Como en el primer spike se pudo probar a breves rasgos las

capacidades de la librería ExtJS a través de tutoriales y ejemplos originales, se propuso la construcción de las funcionalidades de la red como si cada una de ellas fuera un sitio individual, lo que permitió conocer otras funcionalidades que ofrecía la librería.

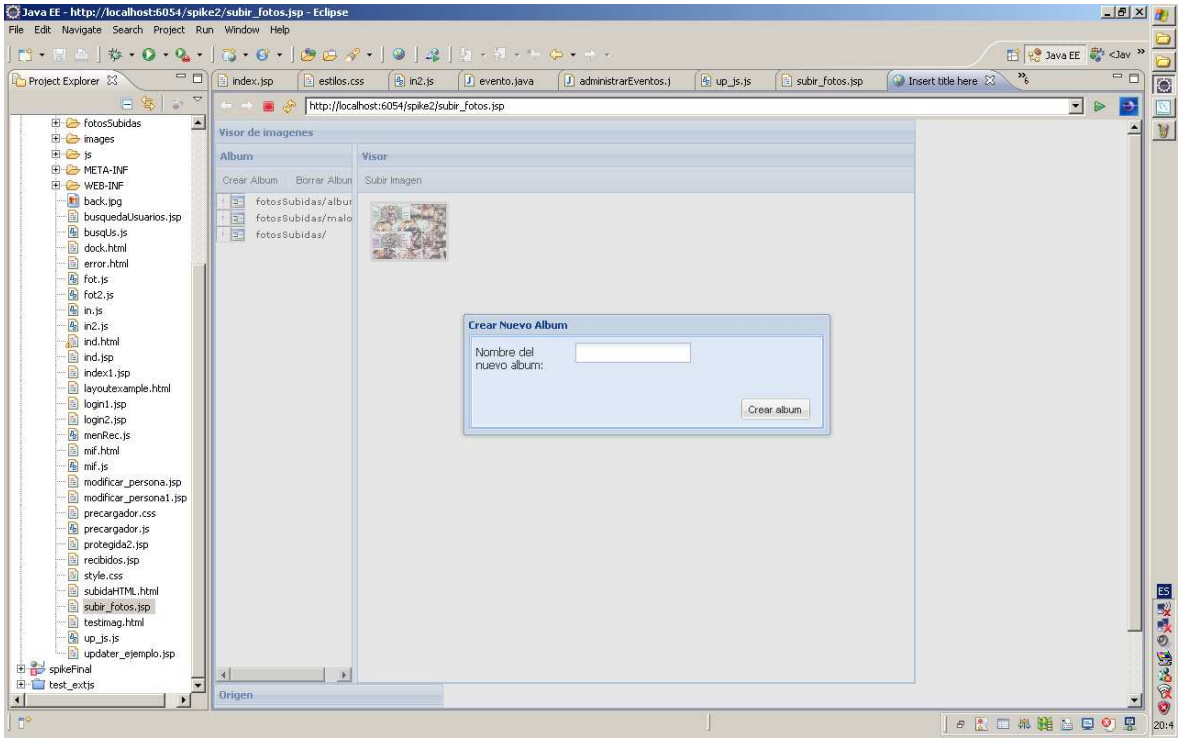

Figura 2.6 Captura de pantalla del módulo de subida de imágenes del segundo spike, compilado como módulo individual.

Varias de las funcionalidades construidas a partir de los ejemplos del primer spike, fueron haciendo notoria la necesidad de definir la capa de control, así como varios tipos de estructuras que la interface requería para mostrar la información traída desde la base de datos. Asimismo, los mensajes de prueba manejados desde los servlets, a través de la consola de la IDE Eclipse, ya no fueron suficientes para poder evaluar el correcto desempeño de la interface con los datos proveídos desde la capa de control. Por esta razón, tuvo que migrarse de navegador, reemplazando el navegador interno que la IDE Eclipse por Mozilla Firefox con el complemento Firebug, que permite evaluar parámetros varios en la interface, como tiempos de carga y paquetes de datos recibidos y enviados.

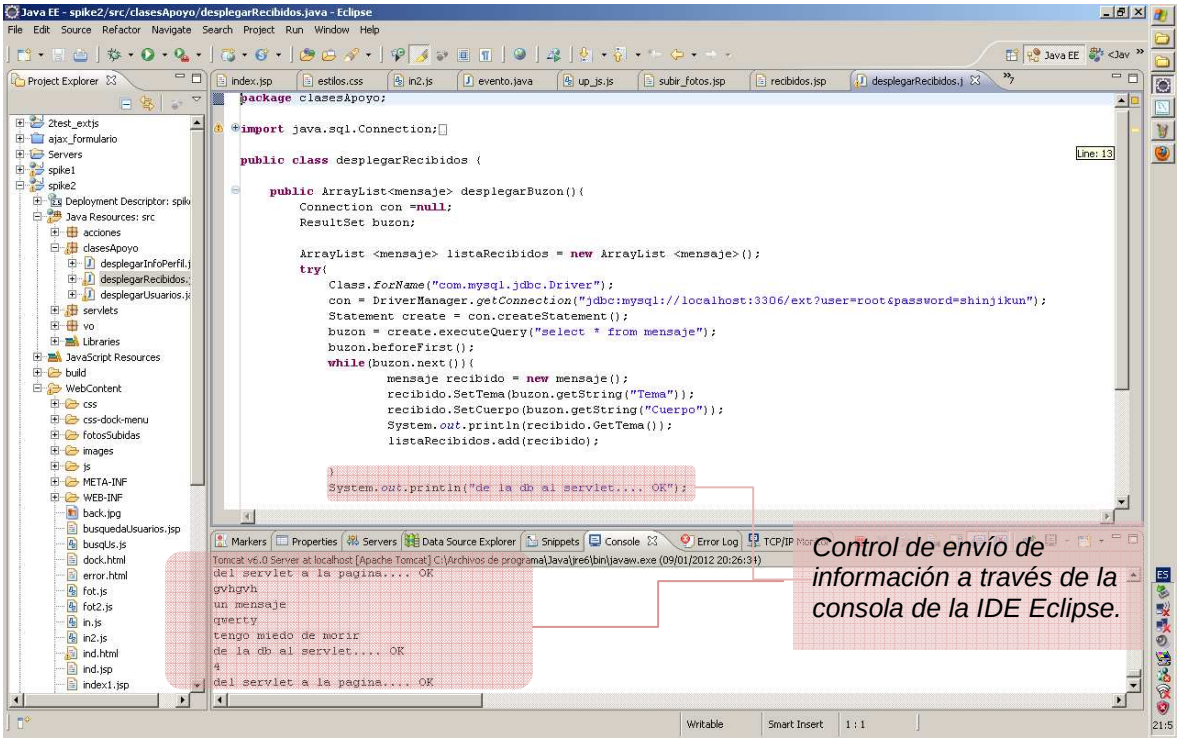

Figura 2.7 Captura de pantalla del servlet del módulo de despliegue de mensaje recibidos del segundo spike, con el control de flujo de información a través de la consola de la IDE Eclipse.

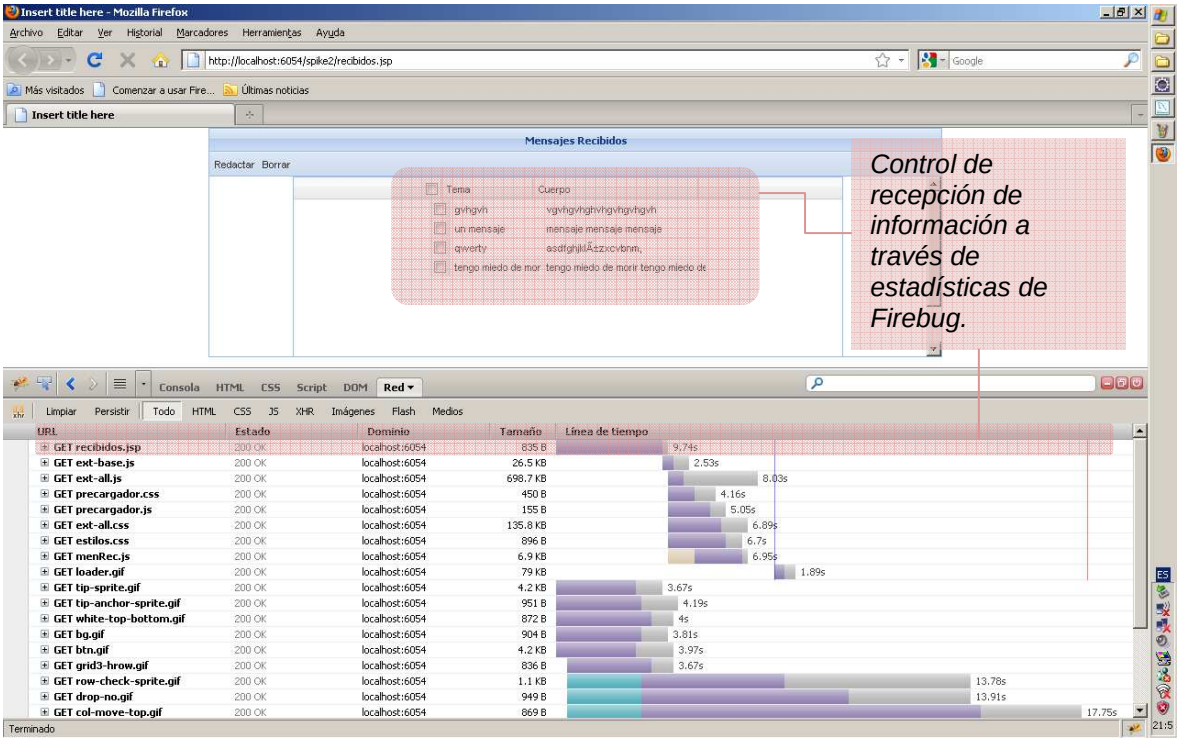

Figura 2.8 Captura de pantalla de la interface del módulo de despliegue de mensaje recibidos del segundo spike, con el control de flujo de información a través de la herramienta Firebug, de Mozilla Firefox.

3. ***Definir cual sería la tecnología seleccionada para la capa de control, para estandarizar los formatos de las llamadas que pudieran producirse entre la interface y la base de datos.*** En el mismo sentido, definir las estructuras de datos que se manejarán para la entrega de información por parte de la base de datos a la interface también se considera una tarea a cumplir. Para la construcción de los paquetes de información a intercambiarse entre la base de datos y la interface En una primera instancia del proyecto, se había optado por utilizar jsp y servlets indistintamente para la capa de control de datos; no obstante, las mismas jsp empezaron a presentarse adecuadas para el manejo de las interfaces, de modo que las servlets quedaron dispuestas para el manejo de la comunicación entre las jsp y la base de datos. Para el manejo de usuarios y control de ingreso, se manejara variables de sesión http tanto en los servlets como en las jsp, tomando uno de los campos de la tabla “usuario”.

   A continuación se presenta el código que tendrá el servlet de login de usuario, basado en el funcionamiento del jsp del primer spike:

```
package servlets;

import java.io.*;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpSession;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

importvo.usuario;
importacciones.administrarUsuario;

/**
 * Servlet implementation class loginUsuario
 */
public class loginUsuario extends HttpServlet {
                                                        private    static    final    long
serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public loginUsuario() {
        super();
        // TODO Auto-generated constructor stub
    }

/*** @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)*/

protected  void  doGet(HttpServletRequest  request,  HttpServletResponse  response)  throws
ServletException, IOException {
// TODO Auto-generated method stub
```

```
}

/*** @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)*/

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
// TODO Auto-generated method stub

response.setContentType("text/html; charset=iso-8859-1");  // Definir tipo de salida
PrintWriter out = response.getWriter();
String usLog_nombre = request.getParameter("nombre");
StringusLog_apellido = request.getParameter("apellido");
intcomprobarUs;

HttpSessionsesion = request.getSession(true);
usuario usLogin = new usuario();
usLogin.SetNombre(usLog_nombre);
usLogin.SetApellido(usLog_apellido);

administrarUsuarioprocesoLogin = new administrarUsuario();
comprobarUs = procesoLogin.localizarUsuario(usLogin);

if (comprobarUs==1)
{

sesion.setAttribute("id", usLog_nombre);
                                                        System.out.println("adquirido
atributo de sesion");

//redireccion con login AJAX
out.write("{success: true}");
}
elseif (comprobarUs==0)
out.write("{success: false, errors: { reason: 'Usuario o contraseña incorrectos. Inténtelo
de nuevo.' }}");

else

out.write("{success: false, errors: { reason: 'Uno o más campos en blanco. Inténtelo de
nuevo.' }}");

}
}
```

Objeto HTTPSession para validar al usuario e iniciar sesión

Por otra parte, la librería ExtJS presenta prestaciones atractivas en el manejo de datos, como envío tradicional de datos a servidores; sin embargo, para la gestión de datos asíncrona y sin refrescos de página, ExtJS propone a AJAX como alternativa por defecto para el envío de datos al servidor desde la interface. Además, es compatible con estructuras de datos JSON (JavaScript Object Notation), lo cual permite desplegar información en la interface traída desde la base de datos, en contraste con estructuras tradicionales como requests y líneas HTML renderizadas desde la capa de control. ExtJS utiliza almacenamientos o *stores* para recibir información de la capa de control (en el caso del prototipo, un servlet) para luego desplegar la información en un objeto definido por ExtJS.

```
Ext.onReady(function() {
Ext.state.Manager.setProvider(new Ext.state.CookieProvider());
Ext.QuickTips.init();

//Conexionpara store
var proxy = new Ext.data.HttpProxy({
url: 'mostrarBuzonRec'
    });
```

Variable proxy para recibir los datos desde el servlet.

```
//Store con JSONReader para buzon de recibidos
buzonRec = new Ext.data.Store({
proxy:proxy,
reader: new Ext.data.JsonReader({
fields: [{name: 'tema', mapping:'tema'},
{name: 'cuerpo', mapping:'cuerpo'}],
root: 'arreglo'
}),
autoLoad:true
});
```

Almacenamiento o "store" para ordenar los datos enviados por el servlet para desplegar en la interface

```
//Construccion  de columna de seleccion para buzon de recibidos
varsm = new Ext.grid.CheckboxSelectionModel();

//Construccion de funcion de seleccion de mensaje de buzon
varcellClickEvent = function(grid,rowIndex,columnIndex,cellIndex,e){
if(columnIndex==1){
Ext.MessageBox.alert('cell click','You just clicked on a cell!');
}
};

//Construcción del grid para despliegue de buzon de recibidos al cargar
página
vargrilla = new Ext.grid.GridPanel({
store: buzonRec,
selModel:sm,
columns:[sm,{header:  "Tema",  width:  100,  sortable:  true,  dataIndex:
'tema'},
          {header:  "Cuerpo",  width:  180,  sortable:  true,  dataIndex:
'cuerpo'}],
stripeRows: false,
height:250,
width:600,
listeners : {
cellclick : cellClickEvent
}

});

});
```

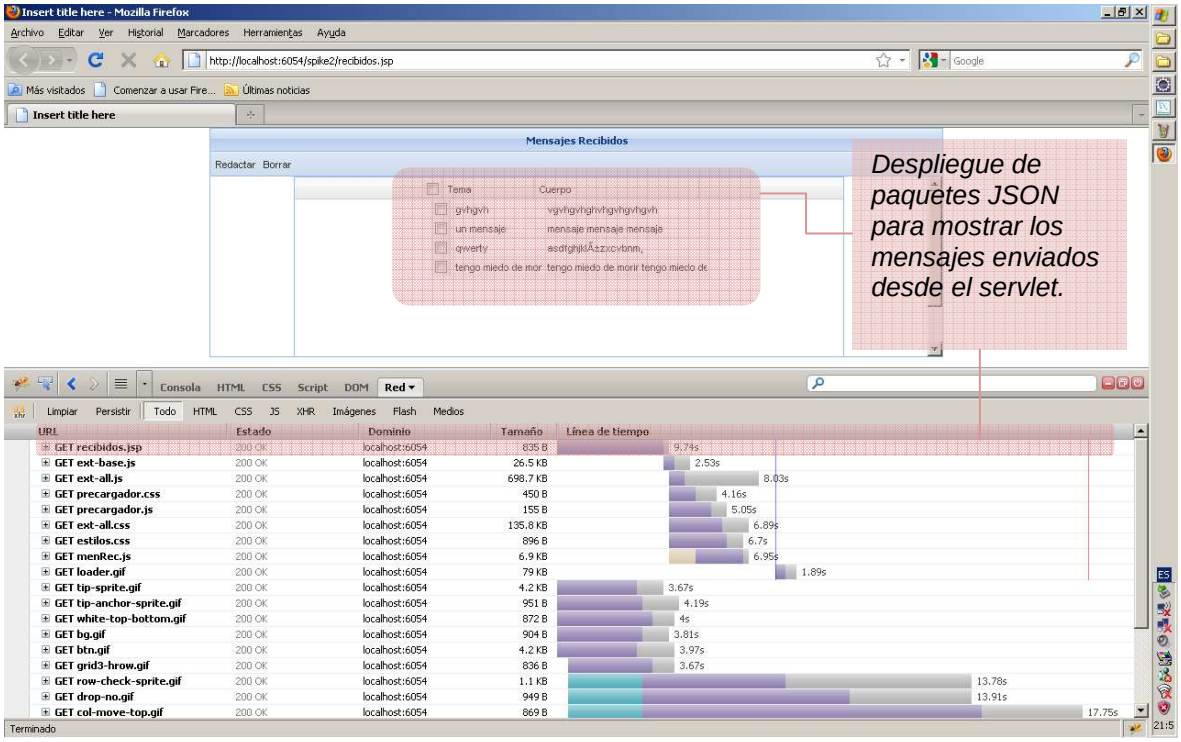

Figura 2.9 Captura de pantalla de la interface del módulo de despliegue de mensaje recibidos del segundo spike, mostrando la recepción de paquetes JSON enviados por el servlet.

Por lo tanto, la estructura del prototipo de la red social quedaría definida de la siguiente manera:

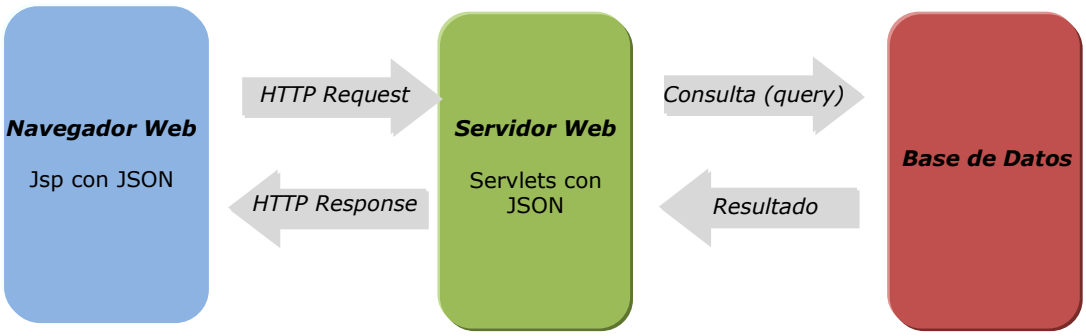

Figura 2.10 Estructura definitiva de la red social

4. *Por último,* ***definir la versión final de la interface del prototipo, para que la versión final del prototipo sea trabajada únicamente en las capas de control y de datos.-*** Luego de construir varias de las funcionalidades del prototipo, se definió la presentación de la interface general donde todas las funcionalidades se llevaran a cabo; a continuación

se presenta la versión final del sitio del prototipo de red social.

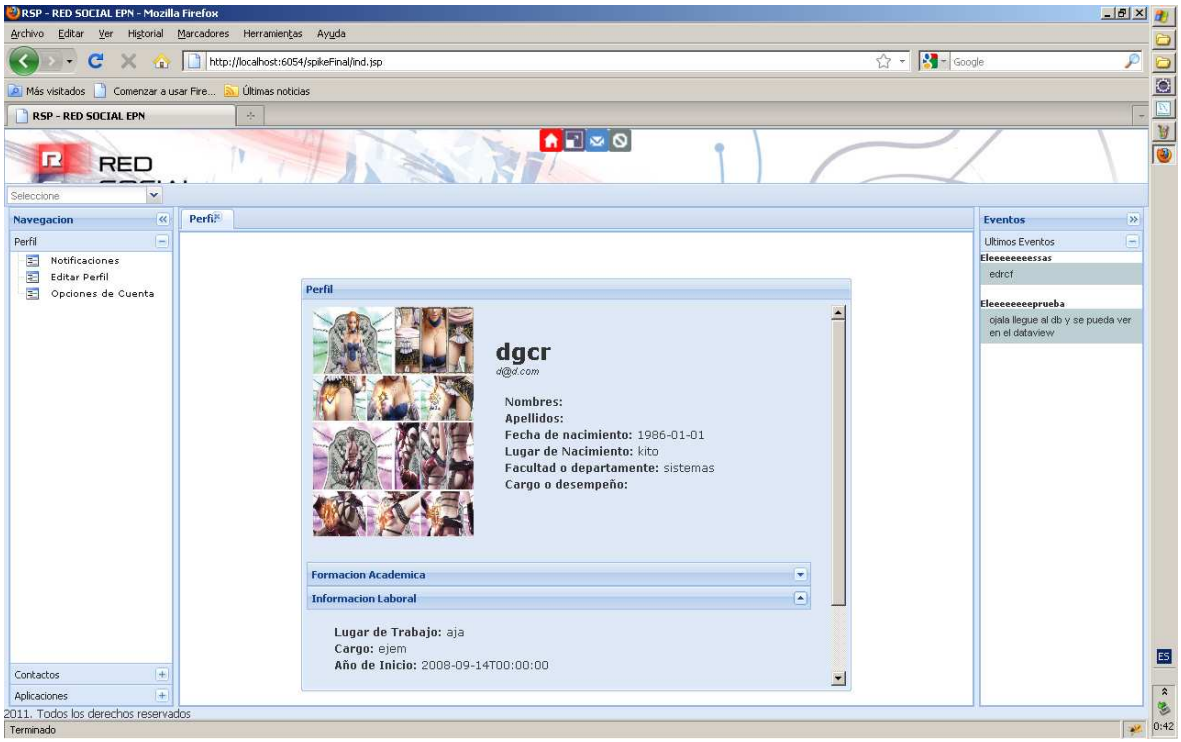

Figura 2.11 Captura de pantalla de la interface principal del prototipo, en su versión final.

2.4.3 SPIKE DEFINITIVO

Una vez definidos los módulos del sitio de red social, su interface definitiva y la tecnología para el intercambio de información entre la interface con el motor de datos, se procedió a definir el modelo de datos definitivo.

Como el sistema trabaja bajo una arquitectura MVC, es conveniente en este instante insistir que para los datos a manejarse dentro del sitio de red social, se definieron distintos objetos (encapsulados en clases JAVA) que conforman la parte del “Modelo” en la arquitectura del sistema, y posteriormente formaran parte del modelo de datos de nuestro sitio de red social:

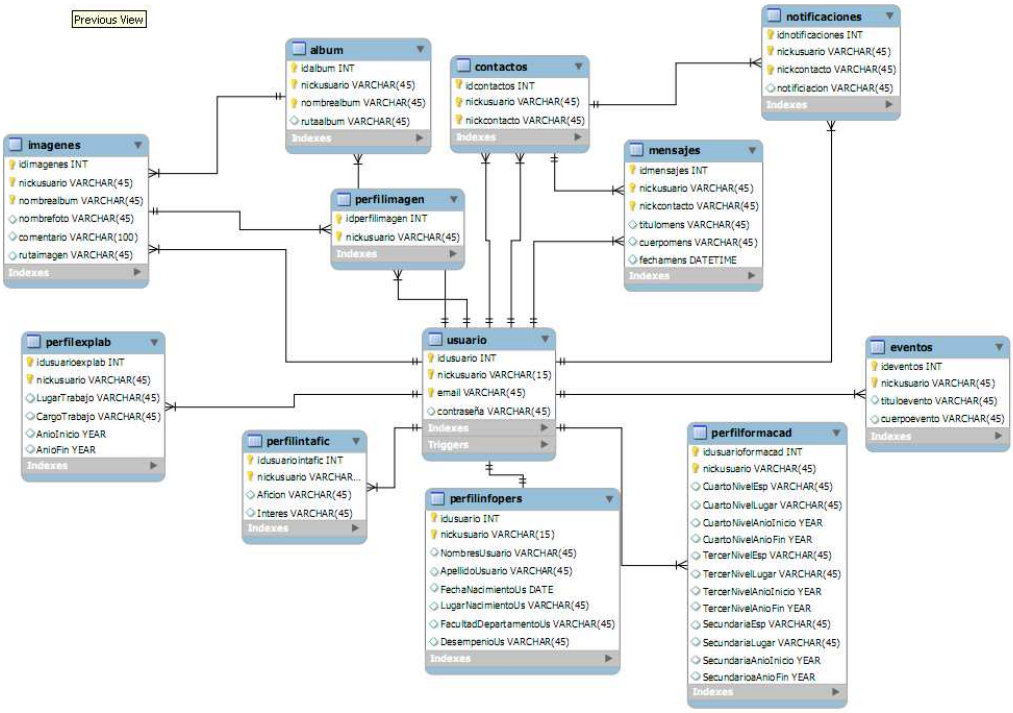

Figura 2.12 Modelo de datos del prototipo.

Utilizando las prestaciones que ofrece MySql Workbench, se esquematizó el modelo entidad-relación de la base de datos del sitio de red social, y mediante la funcionalidad de “ingeniería hacia adelante” (Forward Engineering), se generó la base de datos.

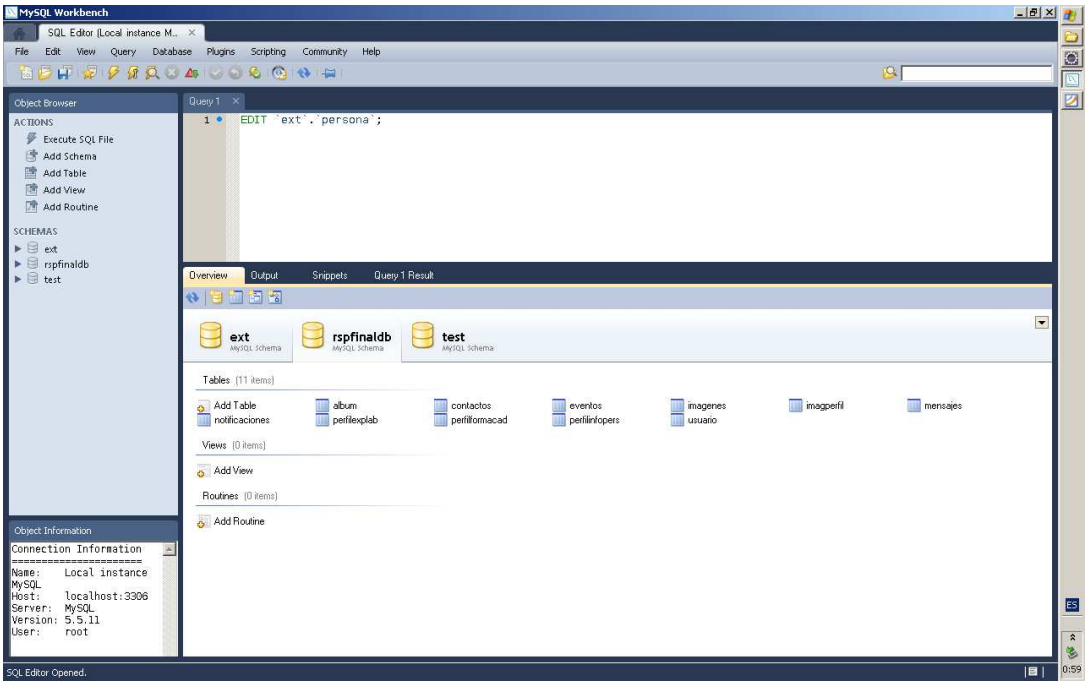

Figura 2.13 Base de datos del prototipo.

Las tablas de la base de datos recogen, principalmente, la siguiente colección de información:

MODELO DE DATOS	
Tablas que contienen información del usuario	Usuario, perfilinfopers, perfilinfoacad, perfilexplab, perfilimag, contactos.
Tablas que contienen información de elementos multimedia	Imagen, eventos, notificaciones, perfilimag
Tablas de despliegue de información general	Eventos, notificaciones, mensajes, contactos

Tabla 2.1 Tabla de contenido de modelo de datos

En la capa de control se redefinieron todos y cada uno de los servlets para que trabajen según el modelo de datos definido. El sitio de red social utiliza servlets para la gestión de información sobre las distintas tablas del modelo de datos.

En el segundo spike, los servlets de las funcionalidades del sitio de red social únicamente trabajaban para el despliegue de información, sin ningún control sobre este intercambio de datos. Para el spike final se consideró la validación de datos, a través de la capa de presentación, ya que ExtJS permite la validación de datos con pocas líneas de código. A continuación se presenta un ejemplo de cómo puede validarse un campo por el tipo de dato requerido e impidiendo que se envíe un campo en blanco:

```
{
xtype: 'textfield',
fieldLabel: 'Email',
name: 'emailnew',
allowBlank:false,
blankText:'Ingrese su correo electronico',
vtype:'email',
vtypeText:'Ingrese una direccion de correo electrónico c
}
```

Configuraciones “allowBlank”, que gestionan campos en blanco.

Configuraciones “vtype”, que gestionan tipos de datos a validar.

Por otra parte, los servlets tienen la tarea de comprobar una sola variable de sesión del usuario que ha ingresado al sistema y gestionar los datos, ya validados desde la capa de presentación.

Por último, se detallaron funcionalidades como el cierre de sesión y despliegue de urls externas relativas a la EPN para concluir con la construcción del prototipo.

2.5 PROYECCIONES DE MANTENIMIENTO Y ANÁLISIS POST-MORTEM DEL PROYECTO

Con el prototipo terminado y habiendo realizado las pruebas pertinentes tanto a nivel de desarrollo como en la fase final de producción del prototipo (los resultados de las pruebas se analizarán en el siguiente capítulo), se han determinado las siguientes proyecciones del mantenimiento del sitio de red social:

- **Implementación de un servidor dedicado:** La arquitectura del prototipo del sitio de red social tiene la siguiente forma:

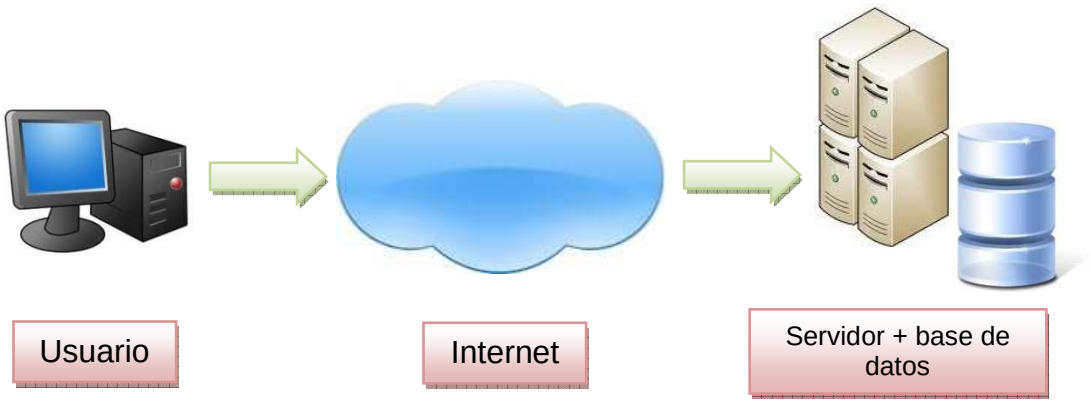

Figura 2.14 Arquitectura de red del prototipo

No obstante, el prototipo de red social esta levantado en su entorno de desarrollo, es decir, funciona bajo el servidor web que se utilizó para las pruebas y “debugging” del sitio, el cual presenta ciertas deficiencias al efectuar varias pruebas. Por lo tanto, para el correcto desempeño del sitio de red social en el tiempo, es necesario levantar la aplicación bajo un servidor dedicado, de manera que se optimice el rendimiento de la

aplicación y se cumpla a cabalidad con la forma de la estructura de la red.

- **Migración a una tecnología superior:** El prototipo está construido en la versión 3.3 de ExtJS; se seleccionó esta versión ya que hasta el cierre de edición del presente documento, posee considerable documentación, tanto oficial (tres libros de texto oficiales además de la documentación oficial de la API) como comunitaria (el foro de Sencha.com, stackoverflow.com). Este fenómeno no ocurre con la última versión, la 4.0, la cual tiene una sintaxis diferente y aun no tiene documentación variada. Sin embargo, esta versión ofrece soporte para dispositivos táctiles y móviles, por lo que si la aplicación tiene demanda y crece, debe tomarse en cuenta la posibilidad de migrar al sitio con la misma API, pero en una versión superior. Esto, claro, cuando exista el soporte documental adecuado y la aplicación misma requiera mencionada actualización.

- **Personal Dedicado:** Si la aplicación adquiere tamaño en las dos proyecciones anteriores, requerirá de más gente que trabaje sobre la aplicación. El prototipo fue desarrollado por una persona, sin embargo, en las pruebas se notó la necesidad de un dominio característico de la estructura de modelo de datos, por lo que un modelo de producción a futuro de la aplicación requerirá necesariamente de un DBA, además del codificador de la interface y la capa de control.

- **Integración del sistema con el servicio de correo interno de la EPN:** La aplicación, en su estado actual de prototipo, solo muestra la funcionalidad de sitio de red social, mostrando las herramientas que esta presenta en sus diferentes módulos. Sin embargo, un usuario puede tener varias cuentas en el sitio como direcciones de correo posea. El objetivo del sitio de red social es, principalmente, integrar a la comunidad politécnica; por lo tanto, los usuarios que la conforman deben ser miembros *reales* de la comunidad politécnica. En este sentido, se ha observado que varios miembros de la comunidad politécnica (profesores, personal administrativo) tienen su propia cuenta de correo electrónico bajo el dominio de la EPN, que es una

sola para cada miembro. Por esta razón, si el prototipo es llevado a producción, debe considerarse la posibilidad de integrarlo con el servicio de correo electrónico interno de la EPN, para que todos los miembros de la comunidad politécnica posean una cuenta de correo que les permita ingresar al sitio de red social, autenticándolos como miembros auténticos de la comunidad politécnica.

Se considera un tiempo de duración de fase de producción del proyecto de 18 a 24 meses, en donde se pondrá fin al proyecto, dando lugar al análisis post-mortem del proyecto, donde se analizarán los resultados obtenidos a nivel social y de integración de la comunidad politécnica.

CAPITULO 3: EVALUACIÓN DEL PROTOTIPO

3.1 PRUEBAS DEL PROTOTIPO

Las pruebas del prototipo se realizaron al final de cada uno de los tres spikes propuestos en la fase de desarrollo. Al final de cada iteración, arrojó como resultado un programa funcional que permitía proceder al desarrollo de una aplicación más compleja hasta alcanzar el modelo final del prototipo de red social planteado en los objetivos del proyecto.

Para la evaluación de los programas funcionales en cada iteración, se utilizaron tres herramientas:

- *Un browser o navegador:* El navegador permitió observar los diferentes comportamientos de la librería ExtJS sobre elementos HTML clásicos, así como la creación de objetos a partir de scripts basados en esta librería. Para los dos primeros spikes se utilizó el navegador interno del IDE Eclipse, mientras que para el spike final se utilizó Mozilla Firefox.
- *La consola del IDE Eclipse:* La consola de Eclipse, por otra parte, facilitó el control del flujo de información entre la base de datos y la capa de control, representada por los servlets, que fueron levantados en el IDE mencionado.
- *La herramienta Firebug, de Mozilla Firefox:* Esta herramienta es considerada un estándar para el “debugging” de aplicaciones desarrolladas con ExtJS, ya que permite el control de objetos y respuestas enviados desde la capa de control a la interface y viceversa, con evaluación de tiempos de respuesta, sin comprometer la seguridad de la aplicación.

Los tipos de pruebas realizadas en cada uno de los spikes fueron los que la metodología XP recomienda:

1. *Pruebas unitarias,*
2. *Pruebas de aceptación.*

Para el primer tipo de pruebas se consideraron las herramientas de evaluación previamente mencionadas y los resultados que arrojaron. Para el segundo tipo, se complementaron estos resultados más las historias de usuario para definir un resultado. Ambos tipos de pruebas estuvieron sujetos a detección de errores o “bugs”, para su correspondiente correctivo.

3.1.1 PRUEBAS UNITARIAS

Las pruebas unitarias realizadas en cada uno de los módulos en cada spike del prototipo permitieron evaluar varios aspectos en el desarrollo del prototipo:

- Cada prueba que se realizaba, se demostraba las cualidades y bondades de la librería ExtJS, como una API poderosa, capaz de generar interfaces enriquecidas con pocas líneas de código, y a su vez exigía al codificador a probar varias de las configuraciones, métodos y eventos de varios elementos que pueden crearse con la librería para la creación de los módulos del prototipo.
- Las pruebas unitarias ayudaron a escoger adecuadamente la tecnología para la capa de control, ya que la idea de un sitio levantado con RIAs se fundamenta en el flujo de información sin refrescos de página y tiempos de respuesta cortos.
- Por otra parte, el desarrollo modulo por modulo permitió un desarrollo “a la par” de cada una de las partes que conforman estos módulos: se puedo crear las clases java adecuadas para los objetos que contendrán los datos, los servlets que gestionan la información y las jsp que despliegan la información junto con los elementos definidos por la librería ExtJS.
- Las pruebas unitarias demostraron ser importantes en la evaluación del desempeño del servidor y el modelo de datos implementados en el prototipo, ya que a partir de dichas observaciones se determinaron varias consideraciones para el correcto desarrollo de un modelo de producción.

A continuación, se presentarán tres tablas que muestran las pruebas unitarias realizadas en cada uno de los spikes, los errores y bugs detectados, su corrección

y el resultado final que se obtuvo:

Primer Spike

Prueba	Errores/Bugs	Corrección	Resultado Final
Ejemplo ExtJS tomado de www.sencha.com	• Página en blanco • Elementos HTML renderizados sin estilos de ExtJS • Configuraciones y métodos de ExtJS no se ejecutan.	• Ubicación adecuada de las librerías *extjs-base.js y extjs-all.js* en una ruta existente. • Adjunción de los archivos CSS para estilizar objetos creados con ExtJS. • Corrección en la sintaxis de los elementos ExtJS para su correcto funcionamiento.	*Ejemplo ExtJS tomado de www.sencha.com correctamente desarrollado*
Ejemplo ExtJS desarrollado por el codificador	• Página en blanco • Configuraciones y métodos de ExtJS no se ejecutan. • Elementos ExtJS desplegados incorrectamente	• Corrección en la sintaxis de los elementos ExtJS para su correcto funcionamiento. • Corrección en los símbolos de agrupación (corchetes) en las funciones y definiciones de los elementos ExtJS. • Creación de archivo de extensión .js para declarar los elementos ExtJS e invocarlos a la página jsp a través de etiquetas `<script>.`	*Ejemplo ExtJS desarrollado por el codificador correctamente desarrollado*

Primera versión de la interface del prototipo	• Configuraciones y métodos de ExtJS no se ejecutan. • Elementos ExtJS desplegados incorrectamente.	• Corrección en la sintaxis de los elementos ExtJS para su correcto funcionamiento. • Creación de archivo de extensión .js para declarar los elementos ExtJS e invocarlos a la página jsp a través de etiquetas `<script>`.	*Primera versión de la interface del prototipo completamente implementado.*
Segunda versión de la interface del prototipo	• Nuevos elementos ExtJS (paneles, treepanels, ventanas emergentes, pestañas) desplegados incorrectamente. • Página en blanco • Despliegue incorrecto de urls externas en paneles ExtJS estándar. • Configuraciones y métodos de ExtJS no se ejecutan.	• Corrección en la sintaxis de los elementos ExtJS para su correcto funcionamiento. • Creación de archivo de extensión .js para declarar los elementos ExtJS e invocarlos a la página jsp a través de etiquetas `<script>`. • Uso del objeto ExtJS *iframe* para carga de urls externas dentro de elementos ExtJS.	*Segunda versión de la interface del prototipo correctamente implementado.*
Primera versión del modulo de login del sistema + integración con interface del prototipo	• Página en blanco luego del envío de datos en el formulario de login. • Despliegue de interface sin variables de sesión. • Despliegue incorrecto de los elementos de la interface.	• Uso de servlets para el envío de información y jsp para recuperación de valores de sesión. • Corrección en la sintaxis de los elementos ExtJS para su correcto funcionamiento.	*Primera versión del modulo de login del sistema implementada, con envío de formularios estándar.*

Tabla 3.1 Tabla de pruebas unitarias del Primer Spike

Segundo Spike:

Pruebas	Errores / Bugs	Corrección	Resultado Final
Módulo de subida (upload) de imágenes	• Imágenes subidas desde la interface no son creadas en la carpeta de destino. • Imágenes subidas al servidor no pueden ser vistas en la interface. • Creación de carpetas para álbumes de imágenes desde la interface no se ve reflejada en la ruta de destino.	• Implementación de los objetos *store* y *proxy* para conexión de objetos ExtJS con la capa de control. • Implementación de objetos JSON tanto en la capa de control como en la interface para el despliegue de datos remotos. • Implementación del objeto *dataview* para la visualización de objetos JSON. • Implementación de librerías *commons* y *fileupload* en el servlet de subida de imágenes para correcta subida y creación de archivos de imagen en la carpeta de destino.	*Módulo de subida (upload) de imágenes correctamente implementado, como módulo individual.*
Módulo de búsqueda de contactos	• Contactos de prueba en la base de datos no se muestran en la interface.	• Implementación de los objetos *store* y *proxy* para conexión de objetos ExtJS con la capa de control. • Implementación de objetos JSON tanto en la capa de control como en la interface para el despliegue de datos remotos.	*Módulo de búsqueda de contactos correctamente implementado, como módulo individual.*

Módulo de personalización de la interface principal del prototipo	• Estilos CSS para interface no se cargan correctamente • Lista de selección de temas para personalización de interface no despliega las opciones.	• Implementación de los objetos *store* y *proxy* para conexión de objetos ExtJS con la capa de control. • Adjunción de los archivos CSS de los nuevos temas para estilizar objetos creados con ExtJS. • Corrección en la sintaxis de los elementos ExtJS para su correcto funcionamiento.	*Módulo de personalización de la interface principal del prototipo correctamente implementado, como módulo individual.*
Módulo de mensajería	• Página en blanco luego del envío de datos en el formulario de envío. • Elementos enviados desde los servlets no se despliegan en la interface. • Nuevos elementos ExtJS (grids, paneles) para despliegue de buzón de mensajes se renderizan incorrectamente.	• Implementación de los objetos *store* y *proxy* para conexión de objetos ExtJS con la capa de control. • Implementación de objetos JSON tanto en la capa de control como en la interface para el despliegue de datos remotos. • Implementación del objeto *gridpanel* para el despliegue objetos JSON del buzón de mensajes. • Corrección en la sintaxis de los elementos ExtJS para su correcto funcionamiento.	*Módulo de mensajería correctamente implementado, como módulo individual.*

Módulo de Eventos	• Página en blanco luego del envío de datos en el formulario de envío. • Elementos enviados desde los servlets no se despliegan en la interface. • Nuevos elementos ExtJS (grids, paneles) para despliegue de buzón de mensajes se renderizan incorrectamente. • Los elementos no se actualizan periódicamente.	• Implementación de los objetos *store* y *proxy* para conexión de objetos ExtJS con la capa de control. • Implementación del objeto *dataview* para la visualización de objetos JSON. • Implementación de las configuraciones *setInterval()* y reload() para la carga periódica de elementos JSON. • Corrección en la sintaxis de los elementos ExtJS para su correcto funcionamiento.	*Módulo de eventos correctamente implementado, como módulo individual.*
Módulo de Notificaciones	• Página en blanco luego del envío de datos en el formulario de envío. • Elementos enviados desde los servlets no se despliegan en la interface. • Nuevos elementos ExtJS (grids, paneles) para despliegue de buzón de mensajes se renderizan incorrectamente.	• Implementación de los objetos *store* y *proxy* para conexión de objetos ExtJS con la capa de control. • Implementación del objeto *dataview* para la visualización de objetos JSON. • Implementación de las configuraciones *setInterval()* y reload() para la carga periódica de elementos JSON. • Corrección en la sintaxis de los elementos ExtJS para su correcto funcionamiento.	*Módulo de notificaciones correctamente implementado, como módulo individual.*

Módulo de actualización de perfil	• Página en blanco luego del envío de datos en el formulario de envío. • Elementos enviados desde los servlets no se despliegan en la interface. • Elementos ExtJS desplegados correctamente pero estéticamente inadecuados. • Elementos requieren de refresco de página para actualizarse.	• Implementación de los objetos *store* y *proxy* para conexión de objetos ExtJS con la capa de control. • Corrección en la sintaxis de los elementos ExtJS para su correcto funcionamiento. • Implementación del objeto *dataview* para la visualización de objetos JSON, con estilos CSS personalizados sobre los elementos del *dataview.* • Implementación de llamadas XHR (Ajax), provistas por ExtJS para redireccionamiento y actualización asíncrona de elementos ExtJS.	*Módulo de actualización de perfil correctamente implementado, como módulo individual.*
Versión final del módulo de login del sistema	• Página en blanco luego del envío de datos en el formulario de envío. • El envío estándar de datos para ingreso no permite redireccionar desde el servlet hacia la aplicación.	• Implementación de llamadas XHR (Ajax), provistas por ExtJS para redireccionamiento y actualización asíncrona de elementos ExtJS. • Corrección en la sintaxis de los elementos ExtJS para su correcto funcionamiento.	*Versión final del módulo de login correctamente implementado, con envío de formularios AJAX.*

Módulo de configuración de cuenta	• Página en blanco luego del envío de datos en el formulario de envío. • Elementos requieren de refresco de página para actualizarse.	• Implementación de los objetos *store* y *proxy* para conexión de objetos ExtJS con la capa de control. • Corrección en la sintaxis de los elementos ExtJS para su correcto funcionamiento. • Implementación de llamadas XHR (Ajax), provistas por ExtJS para redireccionamiento y actualización asíncrona de elementos ExtJS.	*Módulo de configuración de cuenta correctamente implementado, como módulo individual.*
Módulo de solicitud de contactos	• Página en blanco luego del envío de datos en el formulario de envío. • Elementos requieren de refresco de página para actualizarse.	• Implementación de llamadas XHR (Ajax), provistas por ExtJS para redireccionamiento y actualización asíncrona de elementos ExtJS. • Corrección en la sintaxis de los elementos ExtJS para su correcto funcionamiento.	*Módulo de solicitud de contactos correctamente implementado, como módulo individual.*

Tabla 3.2 Tabla de pruebas unitarias del Segundo Spike

Spike Final:

Pruebas	Errores / Bugs	Corrección	Resultado Final
Integración de módulos en versión final de interface	• Página en blanco. • Elementos ExtJS desplegados correctamente pero estéticamente inadecuados.	• Corrección en la sintaxis de los elementos ExtJS para su correcto funcionamiento.	*Integración de módulos en versión final de interface correctamente implementado.*
Integración de interface final + módulo de login con el modelo de datos definitivo	• Página en blanco. • Elementos ExtJS desplegados correctamente pero estéticamente inadecuados. • Respuestas tardías del servidor en proveer los datos. • Módulos desplegados correctamente pero sin la información respectiva.	• Corrección en la sintaxis de los elementos ExtJS para su correcto funcionamiento. • Creación de índices en el modelo de datos para que no exista inconsistencia en los datos manejados. • Implementación de los objetos *store* y *proxy* para conexión de objetos ExtJS con la capa de control. • Limpieza de cache de navegadores y directorios de trabajo del servidor.	*Integración de interface final + módulo de login con el modelo de datos definitivo, pero con errores.*

Tabla 3.3 Tabla de pruebas unitarias del Spike Final

3.1.2 PRUEBAS DE ACEPTACIÓN

Las pruebas de aceptación, según la metodología XP, permiten ver si las historias de usuario se han cumplido, y propone la realización de correctivos hasta que las historias de usuarios se cumplan[22].A partir de las 15 historias de usuario que se obtuvieron, se han determinado cuatro aspectos u objetivos a cumplirse, a partir de las características que el prototipo de sitio de red social debía tener, las cuales las historias de usuario pueden clasificarse:

[22]JOSKOWICZ, José. (2008). **Reglas y Prácticas en eXtremmeProgramming.** IngenieríaTelemática de la Universidad de Vigo, España

- *Tiempos de carga cortos:* Debe ser rápido en sus tiempos de carga y en su ejecución.

- *Personalización:* Debe ser personalizable en los aspectos individuales del usuario (ej: su perfil).

- *Enriquecimiento Visual:* Debe ser lo suficientemente interactivo para justificar su enriquecimiento visual, ofreciendo a una interfaz fácil de utilizar, amigable y en algún grado, conocida para el usuario.

- *Comunicaciones:* Debe estar diseñado para poder establecer comunicaciones con los miembros del espacio de trabajo.

En función de estos aspectos, con sus respectivos objetivos, se realizaron las pruebas de aceptación utilizando las mismas consideraciones que en las pruebas unitarias, es decir, detección de errores, correcciones y resultado final.

Tiempos cortos de carga

Pruebas	Errores / bugs	Corrección	Resultado Final
Ingreso al sistema a través del módulo de login	• Tiempos de carga mayores a 30 segundos. • Desconexión del modulo con el servidor (transacción abortada).	• Limpieza del cache del navegador y directorio de trabajo del servidor. • Reinicio del servidor al presentarse desconexión.	*Ingreso al sistema a través del modulo de login con una ocurrencia de desconexión del 35%*

Carga de la Interface Principal del Prototipo y sus elementos (perfil, eventos, notificaciones, buzón de mensajes)	• Tiempos de carga mayores a 30 segundos. • Elementos de la interface principal renderizados pero sin información remota.	• Limpieza del cache del navegador y directorio de trabajo del servidor. • Reinicio del servidor al presentarse desconexión. • Corrección en la sintaxis del código de los servlets, para su correcto funcionamiento.	*Carga de la Interface Principal del Prototipo y sus elementos, con una ocurrencia de desconexión del 30%*
Actualizaciones de elementos ExtJS del prototipo (perfil, eventos, notificaciones, buzón de mensajes)	• Información remota de los elementos ExtJS del prototipo no se actualiza.	• Limpieza del cache del navegador y directorio de trabajo del servidor. • Corrección en la sintaxis del código de los servlets, para su correcto funcionamiento.	*Actualizaciones de elementos ExtJS, con una ocurrencia de desconexión del 5%*

Tabla 3.4 Tabla de pruebas de aceptación referente a tiempos cortos de carga

Personalización

Pruebas	Errores / bugs	Corrección	Resultado Final
Actualización de datos del perfil de usuario	• Información remota de los elementos ExtJS del prototipo no se actualiza.	• Limpieza del cache del navegador y directorio de trabajo del servidor. • Reinicio del servidor al presentarse desconexión.	Actualización de datos del perfil de usuario *con una ocurrencia de desconexión del 5%*

Personalización del estilo CSS de la interface	• Estilos CSS para interface no se cargan correctamente	• Adjunción de los archivos CSS de los nuevos temas para estilizar objetos creados con ExtJS. • Corrección en la sintaxis de los elementos ExtJS para su correcto funcionamiento.	*Personalización del estilo CSS de la interface, cumplida sin margen de desconexión.*

Tabla 3.5 Tabla de pruebas de aceptación referente a personalización

Enriquecimiento Visual.

Pruebas	Errores / bugs	Corrección	Resultado Final
Efectos visuales y elementos ExtJS personalizados (ventanas emergentes, agregación dinámica de paneles, mensajes de carga).	• Página en blanco • Configuraciones y métodos de ExtJS no se ejecutan. • Elementos ExtJS desplegados incorrectamente • Elementos ExtJS desplegados correctamente pero estéticamente inadecuados.	• Corrección en la sintaxis de los elementos ExtJS para su correcto funcionamiento.	*Efectos visuales y elementos ExtJS personalizados correctamente implementados*
Implementación de JQuery para efectos visuales adicionales (slideshow en el login, dock de botones en la interface principal, mensaje de carga entre modulo de login e interface)	• Página en blanco • Elementos JQuery se despliegan como código HTML normal. • Elementos JQuery desplegados de manera incorrecta.	• Ubicación adecuada de la librería *jquery.js* en una ruta existente. • Actualización de la librería *jquery.js*a su versión más reciente. • Corrección en la sintaxis de los elementos ExtJS y JQuery para su correcto funcionamiento.	*Implementación de JQuery para efectos visuales adicionales, de manera adecuada.*

Actualizaciones asíncronas (AJAX)	• Información remota de los elementos ExtJS del prototipo no se actualiza.	• Limpieza del cache del navegador y directorio de trabajo del servidor. • Reinicio del servidor al presentarse desconexión.	*Actualizaciones asíncronas con una ocurrencia de desconexión del 5%*

Tabla 3.6 Tabla de pruebas de aceptación referente a enriquecimiento visual

Comunicaciones

Pruebas	Errores / bugs	Corrección	Resultado Final
Módulo de mensajería	• Página en blanco luego del envío de datos en el formulario de envío. • Elementos enviados desde los servlets no se despliegan en la interface. • Nuevos elementos ExtJS (grids, paneles) para despliegue de buzón de mensajes se renderizan incorrectamente.	• Implementación de los objetos *store* y *proxy* para conexión de objetos ExtJS con la capa de control. • Implementación de objetos JSON tanto en la capa de control como en la interface para el despliegue de datos remotos. • Implementación del objeto *gridpanel* para el despliegue objetos JSON del buzón de mensajes. • Corrección en la sintaxis de los elementos ExtJS para su correcto funcionamiento.	*Módulo de mensajería correctamente implementado, como módulo individual.*

Módulo de Eventos	• Página en blanco luego del envío de datos en el formulario de envío. • Elementos enviados desde los servlets no se despliegan en la interface. • Nuevos elementos ExtJS (grids, paneles) para despliegue de buzón de mensajes se renderizan incorrectamente. • Los elementos no se actualizan periódicamente.	• Implementación de los objetos *store* y *proxy* para conexión de objetos ExtJS con la capa de control. • Implementación del objeto *dataview* para la visualización de objetos JSON. • Implementación de las configuraciones *setInterval()* y reload() para la carga periódica de elementos JSON. • Corrección en la sintaxis de los elementos ExtJS para su correcto funcionamiento.	*Módulo de eventos correctamente implementado, como módulo individual.*
Módulo de Notificaciones	• Página en blanco luego del envío de datos en el formulario de envío. • Elementos enviados desde los servlets no se despliegan en la interface. • Nuevos elementos ExtJS (grids, paneles) para despliegue de buzón de mensajes se renderizan incorrectamente.	• Implementación de los objetos *store* y *proxy* para conexión de objetos ExtJS con la capa de control. • Implementación del objeto *dataview* para la visualización de objetos JSON. • Implementación de las configuraciones *setInterval()* y reload() para la carga periódica de elementos JSON. • Corrección en la sintaxis de los elementos ExtJS para su correcto funcionamiento.	*Módulo de notificaciones correctamente implementado, como módulo individual.*

Módulo de configuración de cuenta	• Página en blanco luego del envío de datos en el formulario de envío. • Elementos requieren de refresco de página para actualizarse.	• Implementación de los objetos *store* y *proxy* para conexión de objetos ExtJS con la capa de control. • Corrección en la sintaxis de los elementos ExtJS para su correcto funcionamiento. • Implementación de llamadas XHR (Ajax), provistas por ExtJS para redireccionamiento y actualización asíncrona de elementos ExtJS.	*Módulo de configuración de cuenta correctamente implementado, como módulo individual.*
Módulo de solicitud de contactos	• Página en blanco luego del envío de datos en el formulario de envío. • Elementos requieren de refresco de página para actualizarse.	• Implementación de llamadas XHR (Ajax), provistas por ExtJS para redireccionamiento y actualización asíncrona de elementos ExtJS. • Corrección en la sintaxis de los elementos ExtJS para su correcto funcionamiento.	*Módulo de solicitud de contactos correctamente implementado, como módulo individual.*

Tabla 3.7 Tabla de pruebas de aceptación referente a comunicaciones

3.2 COMPARACIÓN DE LOS RESULTADOS CON LOS OBJETIVOS ESTABLECIDOS

Tras haber evaluado el prototipo con las pruebas unitarias y de aceptación, y considerando la naturaleza de los aspectos u objetivos a cumplir se en el desarrollo del prototipo, se establece la siguiente comparación:

Objetivo a cumplir por el prototipo	Resultado obtenido
Debe ser rápido en sus tiempos de carga y en su ejecución.	Se obtiene una aplicación que muestra tiempos de carga que oscilan entre los 2 y los 30 segundos por módulo. Los módulos del prototipo se cargan simultáneamente pero hay una ocurrencia de desconexión con el servidor del prototipo del 27%.
Debe ser personalizable en los aspectos individuales del usuario	El prototipo posee dos módulos que permiten al usuario tener una experiencia personalizada: con el módulo de actualización de perfil puede modificar sus datos a voluntad y con el modulo de personalización de interface, el usuario puede cambiar el estilo CSS de la interface, a través de opciones que la aplicación provee.
Debe ser lo suficientemente interactivo para justificar su enriquecimiento visual, ofreciendo a una interfaz fácil de utilizar, amigable y en algún grado, conocida para el usuario.	Se obtiene una aplicación levantada sobre ExtJS, una extensión de JavaScript, con elementos web enriquecidos visualmente, además de la integración de elementos JQuery en la interface. La aplicación no requiere de refrescos de página para la actualización de información.
Debe estar diseñado para poder establecer comunicaciones con los miembros del espacio de trabajo.	El prototipo ofrece cuatro módulos para establecer comunicaciones entre los usuarios de la aplicación: mensajería, búsqueda y solicitud de contactos, eventos y notificaciones al perfil.

Tabla 3.8 Tabla de comparación con los objetivos propuestos con los resultados obtenidos

Considerando esta comparación, tres de los cuatro aspectos han sido cumplidos exitosamente por el prototipo; el objetivo de rapidez con tiempos cortos de carga se ve afectado debido a la limitación de hardware sobre el cual prototipo fue

levantado; no obstante, esta última deficiencia fue superada modificando ligeramente el entorno de ejecución del prototipo, ejecutando la aplicación con un despliegue (deployment) directo del prototipo como archivo WAR (Web ARchive) en el servidor web, en vez de ejecutarlo desde el IDE de desarrollo Eclipse.

3.3 ANÁLISIS DE COSTOS DE IMPLEMENTACIÓN DEL SISTEMA DE RED SOCIAL PARA LA EPN

Tras haber realizado las pruebas sobre el prototipo de sitio de red social y de comparar los resultados obtenidos en el desarrollo del prototipo con los objetivos inicialmente planteados, es necesario realizar un análisis de costos de implementación de la aplicación, ya que para la puesta en producción del sitio de red social requiere ciertas consideraciones, que se consideraron en el capítulo 2, en el análisis de mantenimiento y post mortem del proyecto:

- *Implementación de un servidor dedicado*
- *Migración a una tecnología superior*
- *Personal Dedicado*

De estas consideraciones, la que más importancia tiene es la referente a la implementación de un servidor dedicado que soporte, en este caso, aplicaciones JAVA y configuraciones para Apache Tomcat.

Si bien el prototipo, como toda aplicación puesta en su fase de producción, es susceptible de adquirir un tamaño considerable, no precisamente requiere de una tecnología en servidores semejante a un clúster o una granja de servidores; no obstante, por tratarse de una aplicación de tipo social, cuyos índices de acceso pueden alcanzar los 35 usuarios simultáneos (Tomcat soporta de 80 a 150 usuarios simultáneos), requiere una arquitectura determinada, sobre todo en el hardware del servidor.

En la actualidad, existen varias empresas que ofrecen servicio de hosting para aplicaciones JAVA, y dependiendo del desempeño que queramos para nuestra

aplicación, el precio irá variando. A mayor rendimiento, mayor costo mensual. Los precios pueden oscilar desde $2,45[23] hasta los $55[24].

Con respecto a la tecnología superior a implementarse, ExtJS como API es de distribución gratuita. Lo que tiene costos son los cursos de entrenamiento que Sencha realiza periódicamente, cuyo valor puede alcanzar los $3800 por un curso de 5 días. No obstante, para la versión 3.3, en la que el prototipo está construido, existe mucha documentación (oficial y comunitaria) para la implementación de elementos nuevos, si la aplicación en su fase de producción así lo requiera.

Para el desarrollo del prototipo, una sola persona se encargó de todas las partes del prototipo: interface, capa de control y modelo de datos. No obstante, si la aplicación entra en fase de producción y crece, es necesario que por lo menos dos personas se encuentren a cargo del proyecto: una que domine la librería ExtJS para la rápida implementación de elementos nuevos con sus respectivas configuraciones, métodos y eventos, y otra encargada de la capa de control para la gestión de nuevas acciones sobre un modelo de datos regularmente administrado. Para el desarrollador de la interface se considera un sueldo mensual de $700 a $900 a tiempo completo, mientras que para el DBA se considera una remuneración de entre $800 y $1100 por la misma carga de trabajo.

Por lo tanto, la proforma para la implementación del sitio de red social a nivel de producción posee los siguientes rubros mensuales:

Por concepto de	Costo Mensual (Promedio)
Hosting Java	$ 25.00
Sueldos Personal (2 personas)	$ 1750.00
API ExtJS	$ 0.00
TOTAL	$ 1775.00

Tabla 3.9 Tabla de costos de implementación del prototipo

[23]MOCHAHost**.** http://www.mejoramos.com/hosting-tomcat-jsp/
[24]MOCHAHost**.** http://www.mejoramos.com/hosting-tomcat-jsp/

CAPITULO 4: CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

- El desarrollo de aplicaciones web de índole social es posible, además de ser llevado a cabo a partir de múltiples opciones de arquitectura de software, así como las tecnologías en las diferentes capas de la aplicación. Las RIAs son adaptativas a las tecnologías de control de flujo de datos, ya que manejan paquetes de datos estándar, así como su desempeño implementado al poseer características visuales incrementadas.
- El prototipo de sitio de red social desarrollado ha demostrado ser funcional en relación a la mayoría de las historias de usuario propuestas. No obstante, también ha presentado ciertas deficiencias que deben ser consideradas en el momento en que el prototipo deba ser implementado ya a nivel de producción.
- Las comunicaciones entre los distintos miembros de la comunidad politécnica se encuentra bastante limitada; si bien existen los espacios necesarios tanto físicos como virtuales para establecer canales de comunicación, aun existe desconocimiento de actividades, grupos y eventos que aportan con el desarrollo de la EPN. En este sentido, el prototipo de red social propone abrir más canales de comunicación, para fomentar la integración de los miembros de la comunidad politécnica, a fin de lograr grupos multidisciplinarios de trabajo para beneficio de la EPN y de la sociedad.

4.2 RECOMENDACIONES

- Se recomienda escoger adecuadamente la plataforma hardware para el desarrollo de aplicaciones, sobre todo cuando estas aplicaciones requieran de elementos de desarrollo tales como servidores, middleware, y entornos de desarrollo que demanden desempeño alto en tiempo real.
- De igual manera, se recomienda desarrollar aplicaciones con flujo de información utilizando herramientas que nos permitan medir los tiempos de respuesta de las diferentes capas de las aplicaciones, así como el tipo de información que se transfiere, al igual que su formato. Hoy en día existen herramientas implementadas en los navegadores web que no solo nos permiten correcciones de las aplicaciones a nivel estético (HTML, CSS), sino también en intercambio de datos, lo que facilita el “debugging” de las aplicaciones, acortando tiempos de desarrollo.
- Por ultimo, se recomienda utilizar tecnología de desarrollo con suficiente y variada documentación, ya sea oficial o comunitaria. La documentación permite el estudio de errores o *bugs* clásicos en las tecnologías de desarrollo, lo que permite dar soluciones rápidas a problemas típicos, sin necesidad de una depuración intensiva de la aplicación en desarrollo.

GLOSARIO

EPN: Escuela Politécnica Nacional.

ExtJS: Extended Java Script. Biblioteca de JavaScript para el desarrollo de aplicaciones web interactivas usando tecnologías como AJAX, DHTML y DOM. Fue desarrollada por Sencha.

RIA: Rich Internet Application. Aplicaciones web que tienen la mayoría de las características de las aplicaciones de escritorio tradicionales. Estas aplicaciones utilizan un navegador web estandarizado para ejecutarse y por medio de complementos o mediante una máquina virtual se agregan las características adicionales.

JAVA: lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Servlets: Objeto que se ejecuta en un servidor o contenedor JEE, especialmente diseñado para ofrecer contenido dinámico desde un servidor web, generalmente HTML.

JSP: Java Server Page. Tecnología Java que permite generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo.

JSON: JavaScript Object Notation. Formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

BIBLIOGRAFÍA

- DUHL, Joshua. **Rich Internet Applications.** IDC White Paper. Noviembre del 2003.

- VARIOS AUTORES. **Rich Internet Applications: Design, Measurement, and Management Challenges.** Keynote White Paper. *http://www.keynote.com/company/resource_library/whitepapers.html*.

- NADA, Tom; HELWIG, Shawn. **Rich Internet Applications: Technical Comparison and Case Studies of AJAX, Flash and Java based RIA.** Best Practice Reports. UW E-Business Consortium. University of Wisconsin-Madison. 16 de noviembre del 2005

- NADA, Tom; HELWIG, Shawn. **Rich Internet Applications: Technical Comparison and Case Studies of AJAX, Flash and Java based RIA.** Best Practice Reports. UW E-Business Consortium. University of Wisconsin-Madison. 16 de noviembre del 2005

- FREEMAN, Linton C. **The Development of Social Network Analysis: A study in the sociology science.** Empirical Press. Vancouver, British Columbia. Canada. 2004

- ANONIMO. **History of Social Networking.** Obtenido de: *http://www.bighistory.net/history-of-social-networking/*. 02 de febrero de 2011.

- ALLEN, Christopher. **Tracing the Evolution of Social Software.** Obtenido de: *http://www.lifewithalacrity.com/2004/10/tracing_the_evo.html*. 13 de Octubre del 2004.

- BOYD, Danah m.; ELLISON, Nicole B. **Sitios de Redes Sociales: Definición, Historia y Conocimiento.** Journal of Computer-MediatedCommunication, 13(1), artículo 11. 2007 (traducción).

- BORTNIK, Sebastián. **Dudas y certezas sobre las redes sociales en las Empresas.** ESET Latinoamérica. Buenos Aires. 02 de agosto del 2010.

- PORRUA GARCÍA, Manuel. **Impacto de las redes sociales.** Dintel. Diciembre 2009/Enero 2010.

ANEXOS

ANEXO 1.- HISTORIAS DE USUARIOS SELECCIONADAS

1. NIVEL DE COMUNICACIÓN ENTRE LAS ENTIDADES DE LA FIS (tipo 1)
    a. *"Soy estudiante de primer semestre y vengo de provincia, y en las matriculas tuve problemas ya que de no ser por una amiga que me comentó que había que pagar una cuota en el 5to piso, me hubiera quedado sin matricula…"*
    b. *"Un problema de la comunicación en la facultad es que las carteleras no se actualizan con frecuencia. Todas las noticias van al correo y a veces llegan tarde y los avisos son muy pequeños..."*
    c. *"Sería interesante conocer la actividad de las células, para ver si nosotros también nos interesamos y participamos…"*
    d. *"… yo solo se que en la facultad, aparte de las clases, esta Imagen FIS y la célula de la IEEE. Nada más."*
    e. *"Sería chévere tener un espacio en donde profesores, estudiantes y autoridades pudiéramos comunicarnos sin tener que pedir citas, o hablar con las secretarias..."*

2. CRITERIOS DE FUNCIONALIDAD DE UNA APLICACIÓN WEB ENRIQUECIDA VISUALMENTE (tipo 2)
    a. *"Tres cosas: rapidez, poder cambiar a nuestro gusto y que sea fácil de usar."*
    b. *"Que tenga bastante opciones para poder modificar nuestra pagina a nuestro gusto…"*
    c. *"Debe ser bastante fluido, con animaciones, con Flash, que tenga la facilidad de manejarse como un escritorio de Windows…"*
    d. *"Que sea menos estático, con pantallas que se muevan y muchos gráficos…"*
    e. *"Que no se demore en cargar mucho, porque eso desespera…"*
    f. *"Una aplicación web buena debe reaccionar igual en cualquier maquina que se le abra…."*
    g. *"… además, la aplicación debe ser segura, para que no haya posibilidad de que nuestros datos sean manipulados por otros…."*
    h. *"Toda aplicación web debe ofrecer seguridad al usuario, de modo que no exista filtración de nuestros datos…."*

3. FACTORES DE SELECCIÓN DE UNA RED SOCIAL (tipo 3)
    a. *"A mi me gusta Facebook porque puede encontrarse amigos bien fácil: solo pones el correo para ver si tiene perfil…"*
    b. *"Una red social debe ayudarte a encontrar a tus amigos mas fácilmente, porque ya encontrándolos puedes ver sus fotos y saber lo que están haciendo…"*
    c. *"… por ejemplo, Facebook no me gusta mucho, porque la página es bien simple, no le puedes poner colores o fondo de pantalla…"*
    d. *"… los juegos de las redes sociales son bonitos pero demoran en cargarse, como el Farmville. A veces solo pasas ahí y te olvidas de otras cosas que el Facebook tiene…"*
    e. *"Debe estar acorde a los cambios y adaptarse a las nuevas tecnologías…."*

ANEXO 2: EJEMPLOS DE CÓDIGO FUENTE DEL PROTOTIPO

CÓDIGO FUENTE DE LA INTERFACE DEL PROTOTIPO

ARCHIVOS JSP

ind.jsp

```
<%
      //Con el siguiente codigo evitamos que el navegador guarde en
cache la pagina si cerramos sesion y ponemos
      //flecha atras

      response.setHeader("Cache-Control","no-cache");
      response.setHeader("Pragma","no-cache");
      response.setDateHeader("Expires",0);
%>

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ page import="vo.usuario"%>
<%@ page import="clasesApoyo.desplegarInfoPerfil"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<%@page import="java.util.*" session="true"%>
<html>


<%
        //Tomamos el valor de sesion validado en el modulo de login para
permitir el acceso a la página -->
        String nomb = (String)session.getAttribute("id");
        //Si no se pasa por el modulo de login, se desplegara un mensaje
de error
      if(nomb==null){
      response.sendRedirect("error.html");
            }
      //Si se pasa por el modulo de login, se desplegara la pagina
principal, conteniedo todos los modulos
        else{
      %>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=ISO-
8859-1">
        <meta http-equiv="Cache-Control" content="no-cache">
        <meta http-equiv="Pragma" content="no-cache">
        <meta http-equiv="Expires" content="Sat, 01 Dec 2001 00:00:00
GMT ">
        <title>RSP - RED SOCIAL EPN</title>

        <!-- ** CSS ** -->
        <!-- base library -->
        <link rel="stylesheet" type="text/css" href="css/ext-all.css" />
            <link href="style.css" rel="stylesheet" type="text/css" />
```

```
        <!-- overrides to base library -->

             <style type="text/css">
 html, body {
        font:normal 12px verdana;
        margin:0;
        padding:0;
        border:0 none;
        overflow:hidden;
        height:100%;
    }

    </style>
        <!-- ** Javascript ** -->

        <script type="text/javascript" src="js/jquery.js"></script>
            <script type="text/javascript"
src="js/interface.js"></script>

            <!-- ExtJS library: base/adapter -->
            <script type="text/javascript" src="js/ext-
base.js"></script>
            <script type="text/javascript" src="js/ext-all.js"></script>

        <!-- ExtJS library: all widgets -->

        <link rel="stylesheet" type="text/css" href="precargador.css">
            <script type="text/javascript"
src="precargador.js"></script>

        <script type="text/javascript"
src="js/ux/fileuploadfield/FileUploadField.js"></script>
      <script type="text/javascript" src="js/ux/DataView-
more.js"></script>
        <script type="text/javascript" src="js/miframe-
debug.js"></script>
        <script type="text/javascript"
src="js/styleswitcher.js"></script>
        <script type="text/javascript" src="js/examples.js"></script>
            <script type="text/javascript" src="in2.js"></script>
        <!-- overrides to library -->

        <!-- extensions -->

        <!-- page specific -->

            <link rel="stylesheet" type="text/css"
href="css/estilos.css" />
      <link rel="stylesheet" type="text/css"
href="js/ux/fileuploadfield/css/fileuploadfield.css"/>
      <link rel="stylesheet" type="text/css" href="css/data-view.css" />
      <link rel="stylesheet" type="text/css" href="css/examples.css" />
```

```
      <!-- Stilos para cambio de apariencia -->

      <link rel="stylesheet" type="text/css" title="blue"
href="css/xtheme-blue.css" />
      <link rel="stylesheet" type="text/css" title="gray"
href="css/xtheme-gray.css" />
      <link rel="stylesheet" type="text/css" title="access"
href="css/xtheme-access.css" />
      <link rel="stylesheet" type="text/css" title="yourtheme"
href="css/yourtheme.css" />

    </head>
    <body>
     <!--divs precargador -->
     <div id="mascara-carga"></div>
            <div id="carga">
                  <div class="indicador">Cargando...</div>
            </div>


 <!--dock menu JS options -->
      <script type="text/javascript">

      $(document).ready(
            function()
            {
                  $('#dock').Fisheye(
                        {
                              maxWidth: 10,
                              items: 'a',
                              itemsText: 'span',
                              container: '.dock-container',
                              itemWidth: 25,
                              proximity: 10,
                              halign : 'center'
                        })

            }
      );

      </script>

    <div id="north">

            <div class="dock" id="dock">
                        <div class="dock-container">
                              <a class="dock-item" href="#"
id="home"><img src="images/rsp/home_50.jpg" alt="Home"
/><span>Home</span></a>
                              <a class="dock-item" href="protegida.jsp"
id="perdock" target="perf11"><img src="images/rsp/perfil_50.jpg"
alt="Perfil" /><span>Perfil</span></a>
```

```
                            <a class="dock-item" href="#"><img
src="images/rsp/buzon_50.jpg" alt="Buzón" /><span>Buzón</span></a>
                            <a class="dock-item" id="salir"><img
src="images/rsp/salir_50.jpg" alt="Salir" /><span>Salir</span></a>
                        </div>
                </div>

    </div>
    <div id="west" class="x-hide-display"></div>
    <div id="perfil" class="x-hide-display"></div>
    <div id="edperf" class="x-hidden"></div>
    <div id="east" class="x-hide-display"></div>
    <div id="south"></div>
    </body>
    <%
        }
        %>
</html>
```

ARCHIVOS JS (ExtJS)

login.js

```
Ext.onReady(function() {
	Ext.state.Manager.setProvider(new Ext.state.CookieProvider());
	Ext.QuickTips.init();

	//FORMULARIO DE LOGIN
	var formLog = new Ext.form.FormPanel({
	  // formulario de Login con AJAX, manejando lamadas XHR
	  renderTo: 'formlogin',//<!-- donde pondremos el formulario -->
	  title: 'RSP Login',//<!-- titulo de la ventana -->
	  url:'loginUsuario',//para llamadas ajax, se especifica la url,
el standardSubmit es falso por defecto
	  frame:true,
	  width: 250,//<!-- ancho del formulario -->
	  heigth: 250,//<!-- ancho del formulario -->
	  items: [{//<!-- items hace referencia a los campos del
formulario es un vector -->
		xtype: 'textfield',//<!--xtype tipo del campo -->
		fieldLabel: 'Email',//<!--fieldLabel texto del campo
-->
		name: 'emailus',//<!--name nombre del campo -->
		allowBlank:false,
		blankText:'Ingrese su correo electronico',
		vtype:'email',
		vtypeText:'Ingrese una direccion de correo
electrónico correcta. Ej: usted@rsp.com'
	    },{
		xtype: 'textfield',//<!--xtype puede ser textfield,
datefield, timefield, numberfield, combo, textarea-->
		fieldLabel: 'Contraseña',
		inputType: 'password',
		name: 'passus',
		allowBlank:false,
		blankText:'Ingrese su contraseña'
	    }],
	    //<!-- en es es un vector precedido de la palabra
buttons: para especificar -->
	    //<!-- que este vector son los botones del formulario en
este caso el submit-->
	  buttons: [{
		text: 'Log in',
		formBind:true,
		handler: function() {
				//evento del boton de login manejando
llamadas de formnato JSON desde el servidor
			formLog.getForm().submit({
				method:'POST',
```

```
                                    success:function(){

                                    Ext.Msg.alert('Estatus',
'Identificación correcta', function(btn, text){
                                    //si el servidor envia succes:true
                                                if (btn == 'ok'){
                                                var redirect = 'ind.jsp';
                                                window.location =
redirect;
                                    }
                                          });
                                },
                                failure:function(form, action){
                                   if(action.failureType ===
Ext.form.Action.SERVER_INVALID){
                                    //si el servidor envia succes:false
mas los errores de invalidadcion
                                    obj =
Ext.util.JSON.decode(action.response.responseText);
                                    Ext.Msg.alert('Identificación
incorrecta', obj.errors.reason);
                                }else{
                                    Ext.Msg.alert('¡Atención!', 'Fallo
de conexión con el servidor de autenticación: ' +
action.response.responseText);
                                }
                                formLog.getForm().reset();
                            }


                             });

                             }


                }]
        });

      //FORMULARIO DE REGISTRO DE NUEVO USUARIO
      var formNew = new Ext.form.FormPanel({
        renderTo: 'formnuevo',
        url:'registarNuevoUsuario',
        frame: true,
        title: 'Registro de Usuarios',
        width: 250,
        heigth: 250,
        items: [{
                    xtype: 'textfield',
                    fieldLabel: 'Email',
                    name: 'emailnew',
                    allowBlank:false,
                    blankText:'Ingrese su correo electronico',
                    vtype:'email',
```

```
                    vtypeText:'Ingrese una direccion de correo
electrónico correcta. Ej: usted@rsp.com'
                },{
                    xtype: 'textfield',
                    fieldLabel: 'Nick RSP',
                    name: 'nicknew',
                    allowBlank:false,
                    blankText:'Ingrese su nombre de usuario'

                },{
                    xtype: 'textfield',
                    fieldLabel: 'Contraseña',
                    inputType: 'password',
                    name: 'passnew',
                    allowBlank:false,
                    blankText:'Ingrese su contraseña',
                    minLength:6,
                    minLengthText:'La contraseña debe contener minimo 6
caracteres.'

                }],
                //<!-- en es es un vector precedido de la palabra
buttons: para especificar -->
                //<!-- que este vector son los botones del formulario en
este caso el submit-->
        buttons: [{
                    text: 'Guardar',
                    formBind:true,
                    handler: function() {
                             formNew.getForm().submit({
                                   method:'GET',
                                   success:function(){

                                         Ext.Msg.alert('Registro
Exitoso', 'Gracias por unirse a RSP. Sus datos acaban de ser enviados a
su dirección de correo.');
                                         formNew.getForm().reset();
                                   },
                                   failure:function(form, action){
                                         if(action.failureType ===
Ext.form.Action.SERVER_INVALID){
                                               //si el servidor envia
succes:false mas los errores de invalidadcion
                                               obj =
Ext.util.JSON.decode(action.response.responseText);
                                               Ext.Msg.alert('Registro
Incorrecto', obj.errors.reason);
                                         }else{

      Ext.Msg.alert('¡Atención!', 'Fallo de conexión con el servidor de
autenticación: ' + action.response.responseText);
                                         }
                                   formNew.getForm().reset();
```

```
                                    }

                                });

                            }
                        }]
                });


});
```

CÓDIGO FUENTE DE LA CAPA DE CONTROL DEL PROTOTIPO

VALUE OBJECTS

usuario.java

```
package vo;

public class usuario {
	private String nickusuario;
	private String email;
	private String contrasena;
	private String imagenUs;
	private String tipoPerfil;

	public usuario(){

	}

	//setters and getters de los atributos del usuario

	public void SetNick(String nickusuario){
		this.nickusuario=nickusuario;
	}

	public void SetEmail(String email){
		this.email=email;
	}

	public void SetContra(String contrasena){
		this.contrasena=contrasena;
	}

	public String GetNick(){
		return this.nickusuario;
	}

	public String GetEmail(){
		return this.email;
	}

	public String GetContra(){
		return this.contrasena;
	}

	public void SetImagUs(String imagenUs){
		this.imagenUs=imagenUs;
	}
	public String GetImagUs(){
		return this.imagenUs;
	}

	public void SetTipoPerf(String tipoPerfil){
```

```
            this.tipoPerfil=tipoPerfil;
      }
      public String GetTipoPerf(){
            return this.tipoPerfil;
      }

}
```

ACCIONES

administrarUsuario.java

```
package acciones;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

import vo.usuario;
import vo.perfilInfoPers;
import vo.perfilFormAcad;
import vo.perfilExpLab;

public class administrarUsuario {
	public void registrarUsuario(usuario newUs){
		Connection con =null;
		try {
			Class.forName("com.mysql.jdbc.Driver");
			con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/rspfinaldb?user
=root&password=shinjikun");
			Statement create = con.createStatement();
			create.executeUpdate("INSERT INTO usuario(nickusuario,
email, contraseña)
VALUES('"+newUs.GetNick()+"','"+newUs.GetEmail()+"','"+newUs.GetContra()
+"')");
			System.out.println("Usuario registrado, desconectando
de base de datos...");
			con.close();
		}
		catch(Exception e){
			e.printStackTrace();
		}
	}


	public int localizarUsuario(usuario us){
		Connection con =null;
		ResultSet usBus;
		String emBus;
		String cnBus;
		int b=9;
		try {
			Class.forName("com.mysql.jdbc.Driver");
			con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/rspfinaldb?user
=root&password=shinjikun");
```

```
                Statement create = con.createStatement();
                usBus = create.executeQuery("select email,contraseña
from usuario where email='"+us.GetEmail()+"'");
                while(usBus.next())
                {
                  emBus= usBus.getString("email");
                  cnBus= usBus.getString("contraseña");
                  if(emBus.equals(us.GetEmail()) &&
cnBus.equals(us.GetContra()))
                        {
                              b=1;//si se cumple es 1
                        }
                  else
                           {
                              b=0; //sino es 0
                        }

             }
                System.out.println("Desconectando del servidor...");
                usBus.close();
                create.close();
                con.close();


          }
          catch(Exception e){
                e.printStackTrace();
          }
          return b;

     }

     public String nickUsuarioLocalizado(usuario us){
          Connection con =null;
          ResultSet usLoc;
          String nickUsLoc="";
          try {
                Class.forName("com.mysql.jdbc.Driver");
                con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/rspfinaldb?user
=root&password=shinjikun");
                Statement create = con.createStatement();
                usLoc = create.executeQuery("select nickusuario from
usuario where email='"+us.GetEmail()+"'");
                usLoc.next();
                nickUsLoc= usLoc.getString("nickusuario");
                System.out.println("Desconectando del servidor...");
                usLoc.close();
                create.close();
                con.close();


          }
```

```
            catch(Exception e){
                  e.printStackTrace();
            }
            return nickUsLoc;

      }

      //Operaciones de modificacion
      public void modificarusuario(perfilInfoPers infoUsmod, String us
){
            Connection con =null;
            try {
                  Class.forName("com.mysql.jdbc.Driver");
                  con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/rspfinaldb?user
=root&password=shinjikun");
                  Statement create = con.createStatement();
                  create.executeUpdate("UPDATE perfilinfopers set
NombresUsuario='"+infoUsmod.GetNombresUsuario()+"',
ApellidoUsuario='"+infoUsmod.GetApellidosUsuario()+"' where
nickusuario='"+us+"'");
                  System.out.println("Usuario modificado, desconectando
de base de datos...");
                  con.close();
            }
            catch(Exception e){
                  e.printStackTrace();
            }
      }

      public void modificarexplab(perfilExpLab expLabmod, String us ){
            Connection con =null;
            try {
                  Class.forName("com.mysql.jdbc.Driver");
                  con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/rspfinaldb?user
=root&password=shinjikun");
                  Statement create = con.createStatement();
                  create.executeUpdate("UPDATE perfilexplab set
LugarTrabajo='"+expLabmod.GetLugarTrabajo()+"',
CargoTrabajo='"+expLabmod.GetCargoTrabajo()+"',
AnioInicio='"+expLabmod.GetAnioInicio()+"' where nickusuario='"+us+"'");
                  System.out.println("Usuario modificado, desconectando
de base de datos...");
                  con.close();
            }
            catch(Exception e){
                  e.printStackTrace();
            }
      }

      public int contactoAmigo(String us, String cont){
            Connection con =null;
```

```
            ResultSet contBus;
            String contacto;
            int b=9;
            try {
                  Class.forName("com.mysql.jdbc.Driver");
                  con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/rspfinaldb?user
=root&password=shinjikun");
                  Statement create = con.createStatement();
                  contBus = create.executeQuery("select nickcontacto from
contactos where nickusuario='"+us+"' and nickcontacto='"+cont+"'");
                  while(contBus.next())
                  {
                    contacto= contBus.getString("nickcontacto");
                    System.out.println(cont);
                    if(contacto.equals(cont))
                          {
                                b=1;
                                System.out.println(cont+" es amigo");
                          }
                    else
                             {
                                b=0;
                                System.out.println(cont+" no es
amigo");
                          }

               }
                  System.out.println("Desconectando del servidor...");
                  contBus.close();
                  create.close();
                  con.close();


            }
            catch(Exception e){
                  e.printStackTrace();
            }
            return b;

      }

}
```

SERVLETS

desplegarInfPerfilJSON.java

```
package servlets;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.*;
import javax.servlet.http.HttpSession;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.util.*;

import org.apache.commons.lang.exception.NestableRuntimeException;
import org.apache.commons.logging.*;
import org.apache.commons.*;
import org.apache.commons.collections.map.ListOrderedMap;
import org.apache.commons.beanutils.*;

import net.sf.json.JSONObject;
import net.sf.json.JSONSerializer;
import net.sf.json.JSONArray;
import net.sf.ezmorph.Morpher;


import clasesApoyo.desplegarInfoPerfil;
import vo.usuario;
import vo.perfilInfoPers;
import vo.perfilFormAcad;
import vo.perfilExpLab;

/**
 * Servlet implementation class desplegarInfPerfilJSON
 */
public class desplegarInfPerfilJSON extends HttpServlet {
      private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public desplegarInfPerfilJSON() {
        super();
        // TODO Auto-generated constructor stub
    }

      /**
       * @see HttpServlet#doGet(HttpServletRequest request,
HttpServletResponse response)
```

```
       */
      protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
            // TODO Auto-generated method stub
            response.setContentType("text/json;charset=UTF-8");
            PrintWriter out = response.getWriter();
            HttpSession sesion = request.getSession(true);
            String idUsuario = (String)sesion.getAttribute("id");
            System.out.println(idUsuario);
            desplegarInfoPerfil traerPerf = new desplegarInfoPerfil();
            usuario userDB = traerPerf.desplegarPerfilUs(idUsuario);
            perfilInfoPers infoPersUs =
traerPerf.desplegarPerfilInfoPers(idUsuario);
            perfilFormAcad formAcadUs =
traerPerf.desplegarPerfilFormAcad(idUsuario);
            perfilExpLab infoLabUs =
traerPerf.desplegarPerfilExpLab(idUsuario);
            //PaqueteJSON para perfil
            JSONArray usuarioJSON = new JSONArray();
            JSONObject usrobjJSON = new JSONObject();
            //info del usuario
            usrobjJSON.put("nickUs", userDB.GetNick());
            usrobjJSON.put("mailUs", userDB.GetEmail());
            String rutaImagen = infoPersUs.GetImagPerfUs();
            if(rutaImagen!=null){
                  usrobjJSON.put("imgUs", infoPersUs.GetImagPerfUs());
            }
            else{
                  usrobjJSON.put("imgUs", "fotosSubidas/default.gif");
            }

            //info personal
            usrobjJSON.put("nombresUs", infoPersUs.GetNombresUsuario());
            usrobjJSON.put("apellidosUs",
infoPersUs.GetApellidosUsuario());
            usrobjJSON.put("fechaNacUs",
infoPersUs.GetFechaNacimientoUs());
            usrobjJSON.put("lugarNacUs",
infoPersUs.GetLugarNacimientoUs());
            usrobjJSON.put("facDeptUs", infoPersUs.GetFacDeptUs());
            usrobjJSON.put("desempUs", infoPersUs.GetDesempenioUs());
            usrobjJSON.put("intUs", infoPersUs.GetInteresUs());
            usrobjJSON.put("aficUs", infoPersUs.GetAficionUs());
            //academico
            usrobjJSON.put("cuartoEspUs",
formAcadUs.GetCuartoNivelEsp());
            usrobjJSON.put("cuartoLugUs",
formAcadUs.GetCuartoNivelLugar());
            usrobjJSON.put("cuartoIniUs",
formAcadUs.GetCuartoNivelAnioInicio());
            usrobjJSON.put("cuartoFinUs",
formAcadUs.GetCuartoNivelAnioFin());
```

```
            usrobjJSON.put("tercerEspUs",
formAcadUs.GetTercerNivelEsp());
            usrobjJSON.put("tercerLugUs",
formAcadUs.GetTercerNivelLugar());
            usrobjJSON.put("tercerIniUs",
formAcadUs.GetTercerNivelAnioInicio());
            usrobjJSON.put("tercerFinUs",
formAcadUs.GetTercerNivelAnioFin());
            usrobjJSON.put("segundoEspUs",
formAcadUs.GetSegundoNivelEsp());
            usrobjJSON.put("segundoLugUs",
formAcadUs.GetSegundoNivelLugar());
            usrobjJSON.put("segundoIniUs",
formAcadUs.GetSegundoNivelAnioInicio());
            usrobjJSON.put("segundoFinUs",
formAcadUs.GetSegundoNivelAnioFin());
            //laboral
            usrobjJSON.put("lugarTrabUs", infoLabUs.GetLugarTrabajo());
            usrobjJSON.put("cargoTrabUs", infoLabUs.GetCargoTrabajo());
            usrobjJSON.put("inicioTrabUs", infoLabUs.GetAnioInicio());
            usuarioJSON.add(usrobjJSON);

            System.out.println("pasado el usuario
"+usrobjJSON.toString());
            //out.write("{success: true, data:"+usrobjJSON+"}");
            out.write("{arregloPerfilUs:"+usuarioJSON+"}");

      }

      /**
       * @see HttpServlet#doPost(HttpServletRequest request,
HttpServletResponse response)
       */
      protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
            // TODO Auto-generated method stub
      }

}
```

SCRIPT DE LA BASE DE DATOS DEL PROTOTIPO

```
-- MySQL dump 10.13  Distrib 5.5.9, for Win32 (x86)
--
-- Host: localhost    Database: rspfinaldb
-- ------------------------------------------------------
-- Server version 5.5.11

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO'
*/;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Current Database: `rspfinaldb`
--

CREATE DATABASE /*!32312 IF NOT EXISTS*/ `rspfinaldb` /*!40100 DEFAULT
CHARACTER SET latin1 */;

USE `rspfinaldb`;

--
-- Table structure for table `album`
--

DROP TABLE IF EXISTS `album`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `album` (
  `idalbum` int(11) NOT NULL AUTO_INCREMENT,
  `nickusuario` varchar(45) NOT NULL,
  `nombrealbum` varchar(45) NOT NULL,
  `rutaalbum` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`idalbum`,`nickusuario`,`nombrealbum`),
  KEY `nickusuarioalbm` (`nickusuario`),
  KEY `nombrealbum` (`nombrealbum`),
  CONSTRAINT `nickusuarioalbm` FOREIGN KEY (`nickusuario`) REFERENCES
`usuario` (`nickusuario`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `album`
--
```

```
LOCK TABLES `album` WRITE;
/*!40000 ALTER TABLE `album` DISABLE KEYS */;
INSERT INTO `album` VALUES (1,'dgcr','album de
dgcr','fotosSubidas/dgcr/album de dgcr'),(2,'dgcr','otro
albumcito','fotosSubidas/dgcr/otro albumcito'),(5,'user2','album de
user2','fotosSubidas/user2/album de user2'),(6,'user3','album de
user3','fotosSubidas/user3/album de user3');
/*!40000 ALTER TABLE `album` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `contactos`
--

DROP TABLE IF EXISTS `contactos`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `contactos` (
  `idcontactos` int(11) NOT NULL AUTO_INCREMENT,
  `nickusuario` varchar(45) NOT NULL,
  `nickcontacto` varchar(45) NOT NULL,
  PRIMARY KEY (`idcontactos`,`nickusuario`,`nickcontacto`),
  KEY `nickusuariocont` (`nickusuario`),
  KEY `nickcontact2` (`nickcontacto`),
  CONSTRAINT `nickcontact2` FOREIGN KEY (`nickcontacto`) REFERENCES
`usuario` (`nickusuario`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `nickusuariocont` FOREIGN KEY (`nickusuario`) REFERENCES
`usuario` (`nickusuario`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `contactos`
--

LOCK TABLES `contactos` WRITE;
/*!40000 ALTER TABLE `contactos` DISABLE KEYS */;
INSERT INTO `contactos` VALUES (1,'dgcr','user2'),(2,'user2','dgcr');
/*!40000 ALTER TABLE `contactos` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `eventos`
--

DROP TABLE IF EXISTS `eventos`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `eventos` (
  `ideventos` int(11) NOT NULL AUTO_INCREMENT,
  `nickusuario` varchar(45) NOT NULL,
  `tituloevento` varchar(45) DEFAULT NULL,
```

```
  `cuerpoevento` varchar(200) DEFAULT NULL,
  PRIMARY KEY (`ideventos`,`nickusuario`),
  KEY `nickusuarioev` (`nickusuario`),
  CONSTRAINT `nickusuarioev` FOREIGN KEY (`nickusuario`) REFERENCES
`usuario` (`nickusuario`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `eventos`
--

LOCK TABLES `eventos` WRITE;
/*!40000 ALTER TABLE `eventos` DISABLE KEYS */;
INSERT INTO `eventos` VALUES
(1,'dgcr','ssas','edrcf'),(2,'dgcr','prueba','ojala llegue al db y se
pueda ver en el dataview'),(3,'dgcr','otro eventito','haver si refresca
el dataview'),(4,'dgcr','mensaje de prueba','ing
tenemaza'),(5,'dgcr','rueba','ing montenegro'),(6,'dgcr','evento en
apache','apache?');
/*!40000 ALTER TABLE `eventos` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `imagenes`
--

DROP TABLE IF EXISTS `imagenes`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `imagenes` (
  `idimagenes` int(11) NOT NULL,
  `nickusuario` varchar(45) NOT NULL,
  `nombrealbm` varchar(45) NOT NULL,
  `rutaimag` varchar(200) NOT NULL,
  `nombreimag` varchar(45) DEFAULT NULL,
  `comentario` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`idimagenes`,`nickusuario`,`nombrealbm`,`rutaimag`),
  KEY `nickusuarioimagen` (`nickusuario`),
  KEY `nombrealbumimag` (`nombrealbm`),
  KEY `rutaimag` (`rutaimag`),
  CONSTRAINT `nickusuarioimagen` FOREIGN KEY (`nickusuario`) REFERENCES
`usuario` (`nickusuario`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `nombrealbumimag` FOREIGN KEY (`nombrealbm`) REFERENCES
`album` (`nombrealbum`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `imagenes`
--

LOCK TABLES `imagenes` WRITE;
```

```
/*!40000 ALTER TABLE `imagenes` DISABLE KEYS */;
INSERT INTO `imagenes` VALUES (1,'dgcr','album de
dgcr','fotosSubidas/dgcr/album de
dgcr/captura.JPG','Foto1',NULL),(2,'dgcr','album de
dgcr','fotosSubidas/dgcr/album de dgcr/angela-
10.jpg','foto2',NULL),(3,'dgcr','otro albumcito','fotosSubidas/dgcr/otro
albumcito/wall_1copia.jpg\'',NULL,NULL);
/*!40000 ALTER TABLE `imagenes` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `mensajes`
--

DROP TABLE IF EXISTS `mensajes`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `mensajes` (
  `idmensajes` int(11) NOT NULL AUTO_INCREMENT,
  `nickusuario` varchar(45) NOT NULL,
  `nickcontacto` varchar(45) NOT NULL,
  `titulomens` varchar(45) DEFAULT NULL,
  `cuerpomens` varchar(45) DEFAULT NULL,
  `fechamens` date DEFAULT NULL,
  `estado` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`idmensajes`,`nickusuario`,`nickcontacto`),
  KEY `nickcontactomsg` (`nickcontacto`),
  KEY `nickusuariomsg` (`nickusuario`),
  CONSTRAINT `nickcontactomsg` FOREIGN KEY (`nickcontacto`) REFERENCES
`contactos` (`nickcontacto`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `nickusuariomsg` FOREIGN KEY (`nickusuario`) REFERENCES
`usuario` (`nickusuario`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `mensajes`
--

LOCK TABLES `mensajes` WRITE;
/*!40000 ALTER TABLE `mensajes` DISABLE KEYS */;
INSERT INTO `mensajes` VALUES (1,'dgcr','user2','prueba1','aki viendo si
funciona','2012-01-01','leido'),(2,'dgcr','user2','prueba2','segunda
prueba','2012-02-23','no
leido'),(4,'user2','dgcr','edrf','cdevfr','2012-02-27','no
leido'),(5,'user2','dgcr','wswd','hola....','2012-03-06','no leido');
/*!40000 ALTER TABLE `mensajes` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `notificaciones`
--
```

```
DROP TABLE IF EXISTS `notificaciones`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `notificaciones` (
  `idnotificaciones` int(11) NOT NULL AUTO_INCREMENT,
  `nickusuario` varchar(45) NOT NULL,
  `nickcontacto` varchar(45) NOT NULL,
  `notificiacion` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`idnotificaciones`,`nickusuario`,`nickcontacto`),
  KEY `nickusernot` (`nickusuario`),
  KEY `nickcontactnot` (`nickcontacto`),
  CONSTRAINT `nickcontactnot` FOREIGN KEY (`nickcontacto`) REFERENCES
`contactos` (`nickcontacto`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `nickusernot` FOREIGN KEY (`nickusuario`) REFERENCES
`usuario` (`nickusuario`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `notificaciones`
--

LOCK TABLES `notificaciones` WRITE;
/*!40000 ALTER TABLE `notificaciones` DISABLE KEYS */;
/*!40000 ALTER TABLE `notificaciones` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `perfilexplab`
--

DROP TABLE IF EXISTS `perfilexplab`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `perfilexplab` (
  `idusuarioexplab` int(11) NOT NULL AUTO_INCREMENT,
  `nickusuario` varchar(45) NOT NULL,
  `LugarTrabajo` varchar(45) DEFAULT NULL,
  `CargoTrabajo` varchar(45) DEFAULT NULL,
  `AnioInicio` varchar(20) DEFAULT NULL,
  `AnioFin` date DEFAULT NULL,
  PRIMARY KEY (`idusuarioexplab`,`nickusuario`),
  KEY `nickusuarioexp` (`nickusuario`),
  CONSTRAINT `nickusuarioexp` FOREIGN KEY (`nickusuario`) REFERENCES
`usuario` (`nickusuario`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `perfilexplab`
--

LOCK TABLES `perfilexplab` WRITE;
```

```
/*!40000 ALTER TABLE `perfilexplab` DISABLE KEYS */;
INSERT INTO `perfilexplab` VALUES
(1,'dgcr','','','',NULL),(4,'user2',NULL,NULL,NULL,NULL),(5,'user3',NULL
,NULL,NULL,NULL);
/*!40000 ALTER TABLE `perfilexplab` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `perfilformacad`
--

DROP TABLE IF EXISTS `perfilformacad`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `perfilformacad` (
  `idusuarioformacad` int(11) NOT NULL AUTO_INCREMENT,
  `nickusuario` varchar(45) NOT NULL,
  `CuartoNivelEsp` varchar(45) DEFAULT NULL,
  `CuartoNivelLugar` varchar(45) DEFAULT NULL,
  `CuartoNivelAnioInicio` date DEFAULT NULL,
  `CuartoNivelAnioFin` date DEFAULT NULL,
  `TercerNivelEsp` varchar(45) DEFAULT NULL,
  `TercerNivelLugar` varchar(45) DEFAULT NULL,
  `TercerNivelAnioInicio` date DEFAULT NULL,
  `TercerNivelAnioFin` date DEFAULT NULL,
  `SecundariaEsp` varchar(45) DEFAULT NULL,
  `SecundariaLugar` varchar(45) DEFAULT NULL,
  `SecundariaAnioInicio` date DEFAULT NULL,
  `SecundariaAnioFin` date DEFAULT NULL,
  PRIMARY KEY (`idusuarioformacad`,`nickusuario`),
  KEY `nickusuarioform` (`nickusuario`),
  CONSTRAINT `nickusuarioform` FOREIGN KEY (`nickusuario`) REFERENCES
`usuario` (`nickusuario`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `perfilformacad`
--

LOCK TABLES `perfilformacad` WRITE;
/*!40000 ALTER TABLE `perfilformacad` DISABLE KEYS */;
INSERT INTO `perfilformacad` VALUES
(1,'dgcr',NULL,NULL,NULL,NULL,'Ingeniero','EPN - Kito','2004-10-
12','2011-07-23','FIMA','Colegio Garka','1997-10-06','2003-07-
13'),(4,'user2',NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,N
ULL),(5,'user3',NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,N
ULL);
/*!40000 ALTER TABLE `perfilformacad` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `perfilinfopers`
```

```
--

DROP TABLE IF EXISTS `perfilinfopers`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `perfilinfopers` (
  `idusuario` int(11) NOT NULL AUTO_INCREMENT,
  `nickusuario` varchar(15) NOT NULL,
  `NombresUsuario` varchar(45) DEFAULT NULL,
  `ApellidoUsuario` varchar(45) DEFAULT NULL,
  `FechaNacimientoUs` date DEFAULT NULL,
  `LugarNacimientoUs` varchar(45) DEFAULT NULL,
  `FacultadDepartamentoUs` varchar(45) DEFAULT NULL,
  `DesempenioUs` varchar(45) DEFAULT NULL,
  `InteresUs` varchar(45) DEFAULT NULL,
  `AficionUs` varchar(45) DEFAULT NULL,
  `imagPerfil` varchar(200) DEFAULT NULL,
  PRIMARY KEY (`idusuario`,`nickusuario`),
  KEY `nickusuarioinfo` (`nickusuario`),
  KEY `imagusuarioperf` (`imagPerfil`),
  CONSTRAINT `imagusuarioperf` FOREIGN KEY (`imagPerfil`) REFERENCES
`imagenes` (`rutaimag`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `nickusuarioinfo` FOREIGN KEY (`nickusuario`) REFERENCES
`usuario` (`nickusuario`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `perfilinfopers`
--

LOCK TABLES `perfilinfopers` WRITE;
/*!40000 ALTER TABLE `perfilinfopers` DISABLE KEYS */;
INSERT INTO `perfilinfopers` VALUES (1,'dgcr','persona','','1986-01-
01','kito','sistemas','egresado','coser,cantar,comer',' el jarabe
tapatio, las morenas','fotosSubidas/dgcr/album de dgcr/angela-
10.jpg'),(4,'user2','Usuario','Dos','1988-12-
05','cuenca','electrica','estudiante','interes
1','aficion1',NULL),(5,'user3','Usuario','Tres','1980-11-
13','guayaquil','ambiental','profesor','interes2','interes2',NULL);
/*!40000 ALTER TABLE `perfilinfopers` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `solicitudes`
--

DROP TABLE IF EXISTS `solicitudes`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `solicitudes` (
  `idsolicitudes` int(11) NOT NULL AUTO_INCREMENT,
  `nickusuario` varchar(45) NOT NULL,
```

```
  `nicksolicitante` varchar(45) NOT NULL,
  PRIMARY KEY (`idsolicitudes`,`nickusuario`,`nicksolicitante`),
  KEY `nickussol` (`nicksolicitante`),
  KEY `nickconsol` (`nickusuario`),
  CONSTRAINT `nickconsol` FOREIGN KEY (`nickusuario`) REFERENCES
`usuario` (`nickusuario`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `nickussol` FOREIGN KEY (`nicksolicitante`) REFERENCES
`usuario` (`nickusuario`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `solicitudes`
--

LOCK TABLES `solicitudes` WRITE;
/*!40000 ALTER TABLE `solicitudes` DISABLE KEYS */;
INSERT INTO `solicitudes` VALUES (1,'dgcr','user3');
/*!40000 ALTER TABLE `solicitudes` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `usuario`
--

DROP TABLE IF EXISTS `usuario`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `usuario` (
  `idusuario` int(11) NOT NULL AUTO_INCREMENT,
  `nickusuario` varchar(15) NOT NULL,
  `email` varchar(45) NOT NULL,
  `contraseÃ±a` varchar(45) DEFAULT NULL,
  `tipoperfil` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`idusuario`,`nickusuario`,`email`),
  KEY `nickusuario` (`nickusuario`)
) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `usuario`
--

LOCK TABLES `usuario` WRITE;
/*!40000 ALTER TABLE `usuario` DISABLE KEYS */;
INSERT INTO `usuario` VALUES
(1,'dgcr','d@d.com','wsed','publico'),(4,'user2','user2@2.com','usuario2
','privado'),(5,'user3','test3@3.org','usuario3','publico');
/*!40000 ALTER TABLE `usuario` ENABLE KEYS */;
UNLOCK TABLES;
/*!50003 SET @saved_cs_client      = @@character_set_client */ ;
/*!50003 SET @saved_cs_results     = @@character_set_results */ ;
/*!50003 SET @saved_col_connection = @@collation_connection */ ;
```

```
/*!50003 SET character_set_client  = utf8 */ ;
/*!50003 SET character_set_results = utf8 */ ;
/*!50003 SET collation_connection  = utf8_general_ci */ ;
/*!50003 SET @saved_sql_mode       = @@sql_mode */ ;
/*!50003 SET sql_mode              =
'STRICT_TRANS_TABLES,STRICT_ALL_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERRO
R_FOR_DIVISION_BY_ZERO,TRADITIONAL,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTIT
UTION' */ ;
DELIMITER ;;
/*!50003 CREATE*/ /*!50017 DEFINER=`root`@`localhost`*/ /*!50003 TRIGGER
testref AFTER INSERT ON usuario
  FOR EACH ROW BEGIN
    INSERT INTO perfilinfopers SET nickusuario = NEW.nickusuario;
    INSERT INTO perfilformacad SET nickusuario = NEW.nickusuario;
    INSERT INTO perfilexplab SET nickusuario = NEW.nickusuario;

  END */;;
DELIMITER ;
/*!50003 SET sql_mode              = @saved_sql_mode */ ;
/*!50003 SET character_set_client  = @saved_cs_client */ ;
/*!50003 SET character_set_results = @saved_cs_results */ ;
/*!50003 SET collation_connection  = @saved_col_connection */ ;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2012-04-15 10:02:05
```