Generador de redes sociales con datos sintéticos

Memoria para la tesina del máster de Ingeniería de Computadores, DISCA

Autor:	**Miguel Ángel Garijo Quintana**
Email del autor:	**migaqui@ei.upv.es**
Tutor:	**Ana Pont Sanjuán**
Co Tutor:	**José Antonio Gil Salinas**

Índice

1. Introducción

Hoy en día, el término “Red Social” se ha convertido en un término muy habitual. Dejando de lado el significado que hace referencia a la representación estructural de la sociedad, se definen las redes sociales en internet como servicios o aplicaciones web que permiten la interacción entre usuarios que crean, comparten e intercambian contenidos.

La importancia en internet de estos servicios está fuera de discusión, ya que la gran mayoría de usuarios de internet las conocen y un porcentaje muy elevado las utiliza a diario. Como muestra, la mayoría de las empresas o instituciones que tienen presencia en internet, se publicitan a través de las redes sociales para llegar a mayor número de usuarios. Por lo tanto, ya no solo hablamos de una actividad lúdica, sino también comercial.

El comienzo de uso de estos servicios es relativamente cercano, ya que las principales redes sociales apenas tienen unos diez años de vida. Es por esto que todavía no se trata de un tipo de aplicación bien definido, ya que está en proceso de creación, renovándose constantemente. Además, debido a que un número muy elevado de accesos se realizan desde diferentes dispositivos móviles mediante aplicaciones muy diversas, este servicio muestra un nuevo grado de complejidad.

Entrando en un aspecto técnico, el soporte físico y lógico para los servicios de redes sociales tendrá unos requerimientos diferentes respecto a los de los sitios web habituales. No obstante, se han ido destinando en numerosas ocasiones servidores de uso común a soportar estos servicios, sin tener en cuenta ni poder estimar, los recursos que se iban a necesitar. En otras palabras, llegó antes la utilización, que el poder estudiar las necesidades.

Para los tipos de aplicaciones web más conocidas y con un tiempo de uso más longevo, o simplemente que desde sus inicios han tenido una finalidad más comercial, existen varias herramientas para analizar y conocer el soporte que se necesitará, permitiendo adelantarse a las limitaciones que se puedan encontrar. Para los servicios que estamos analizando, estas herramientas no existen. Las herramientas de las que hablamos son los *benchmarks* [4], que son técnicas utilizadas para medir el rendimiento de un sistema o componente del mismo.

Por lo tanto, el objetivo de éste estudio, consiste en aportar herramientas y métodos para obtener un estudio y análisis del soporte que necesitará un servicio de red social. Así se podrá tener una estimación de uso de recursos previo a la instalación y explotación. Concretamente, en este trabajo se desarrolla un banco de pruebas que nos permitirá disponer de una base con la que estudiar y evaluar una red social.

El propósito de este banco de pruebas es generar la base para procesos de simulación de uso de la red social. Si bien el desarrollo de la carga de trabajo (workload) queda fuera del ámbito de esta tesina, el resultado final de este trabajo será obtener una red social totalmente funcional con contenidos generados y cargados de forma sintética. Esto junto con la carga de trabajo adecuada permitirá disponer

de un *benchmark* para realizar estudios de evaluación de prestaciones en entornos prácticamente reales. Lo que permitirá analizar el comportamiento del servidor tanto a nivel lógico como físico al ejecutar simulaciones de uso con procesos externos que ataquen al servicio creado.

2. Las Redes sociales

Se ha comenzado hablando acerca de las redes sociales, pero no se ha definido realmente en que consiste una red social. Pese a que todos conocemos ejemplos de éstas, se va a describir qué características se consideran las más significativas.

Para empezar, existen varios tipos de redes sociales, diferenciados según los contenidos o el tipo de perfil de usuario al que están destinadas. Algunos de los tipos y ejemplos de las mismas son los siguientes:

- Proyectos colaborativos (Wikipedia)
- Blogs y microblogs (Twitter)
- Comunidades de contenidos (YouTube)
- Lugares de intercomunicación social (Facebook)
- Mundos de juegos virtuales (World of Warcraft)
- Mundos de sociedad virtual (Second Life)

2.1. Características de las redes sociales

Cada red social tiene unas características que la define y la hacen diferente a otras, pero en un nivel más básico, todas entre sí comparten muchas de las funcionalidades por lo que considerarlas redes sociales. A continuación se detallan las que se aceptan por la mayoría de autores como características básicas de las redes sociales:

1. **Identidad**

   Parece evidente, que cualquier sistema de red social, debe tener identificación de usuarios, es decir, definir de forma inequívoca cada uno de los usuarios. Además, esta información, en un caso ideal debe dar suficiente información sobre la identidad y necesidades del usuario que hay detrás, para poder cumplir las funciones del sistema o servicio.

2. **Relaciones entre usuarios**

   En la mayoría de los casos, se podrán establecer relaciones entre las identidades de una forma no jerárquica. Es decir, cada uno de los usuarios puede ser “amigo” o “seguidor” de otros usuarios, sin poder ni necesitar la intervención de un tercero. Estas relaciones pueden ser

unidireccionales o bidireccionales, es decir, un usuario puede ser seguidor de un segundo usuario sin que éste lo siga, o puede que los dos usuarios se sigan entre sí.

3. Conversaciones o mensajería

Entre los diferentes usuarios, ya sean relacionados o no, existirá la posibilidad de conversar entre sí. En algunos casos de forma pública y en otros de forma privada, según las posibilidades y características de la red social en cuestión.

4. Publicaciones

Estas conversaciones públicas, se pueden considerar publicaciones, ya que pueden ser simplemente mensajes que se escriben sin un destinatario en concreto, y algún o algunos usuarios pueden responder.

Dentro de las publicaciones, tienen gran potencial los enlaces a otras webs o redes sociales con contenidos de interés, o simplemente contenidos de textos que el usuario desea compartir.

5. Compartición de contenidos

Además de texto, se ofrece la posibilidad de publicar archivos con algún tipo de contenido. Estos archivos pueden ser de varios tipos, pero por lo general, los contenidos más frecuentemente compartidos suelen ser imágenes. En estos archivos se suele poder hacer referencia a otros usuarios para que tengan accesible y reciban notificación de éstos.

6. Otras funcionalidades habituales

Existen otras características, que aun no siendo básicas, tienen un uso bastante extendido y habitual. Algunas de ellas son:

- Búsqueda de usuarios: Los usuarios hacen uso de un motor de búsqueda de usuarios, y en ocasiones motores de recomendaciones de usuarios, para establecer una relación con ellos a nivel de red social.
- Notificaciones: De alguna forma, cada uno de los usuarios puede ver los nuevos mensajes recibidos y las menciones o referencias que se han hecho.
- Encuestas o votaciones: Se ofrece la posibilidad de que un usuario lance una encuesta digital sobre un grupo definido de usuarios.

2.2. Tipos de uso de la red social

Por lo general, las aplicaciones de red social están abiertas a todos los usuarios de internet. Es por eso que los usuarios pueden tener perfiles de uso muy diferentes, y los contenidos publicados pueden ser muy variados, y con distribuciones de presencia muy dispares. Por ejemplo, los usuarios pueden publicar contenidos muy extensos o muy cortos, y puede haber usuarios con muchas publicaciones o que apenas hacen uso del sistema.

A continuación se presenta una posible definición de agrupaciones de tipos de usuario en una red social:

Usuario social

Se identifica como un usuario con un número aleatorio de relaciones con otros usuarios (relación de amistad). Este supuesto perfil de usuario consultará las publicaciones de sus relaciones, y a su vez publicará contenidos que serán consultados por sus relaciones. Casi con total seguridad, se tratará de un usuario particular.

Lector de contenidos

Se trata de un usuario, también con un número aleatorio de relaciones con otros usuarios, pero que por lo general no aporta contenidos. No obstante, al igual que el usuario social, accede habitualmente y lee contenidos de otros usuarios de cualquier otro perfil. En este caso, también se puede considerar que el usuario va a ser un usuario particular.

Generador de contenidos

Se puede definir como un usuario que en la mayoría de los casos, se dedica a publicar contenidos, y por lo general extensos. Normalmente, no va a consultar los contenidos de los usuarios con los que tiene relación en la red social. En un caso práctico, se puede ceñir a una empresa que decide tener presencia en redes sociales con objetivos publicitarios. Estas publicaciones pueden ser anuncios publicitarios, o simplemente contenidos de interés que aporten visitas al sitio web en el que se publicitan u ofrecen servicios.

Lógicamente, existirán muchos más perfiles de usuario, pero las agrupaciones a rasgos generales, son las descritas en estos tres grupos. Así pues, la distribución de perfiles de usuario, con diferentes números de relaciones, diferente cantidad de contenidos y diferentes tipos de acceso, se puede considerar aleatoria. Calcular el número de cada tipo de perfiles de usuario sería posible, pero para esto, ya se debería haber tenido que estar explotando el sistema durante un tiempo, provocando problemas por las posibles carencias del entorno.

2.3. Repercusiones en el sistema

El acceso de un único usuario, no va a ser para nada problemático en el sistema. Sin embargo, varios cientos de accesos simultáneos de diferentes usuarios, ya puede llegar a generar que alguno de los recursos se sature provocando lentitud en el servicio, incluso la caída del mismo.

Los recursos que se deben tener en cuenta en un entorno de servicio web con las características propias de una red social serán muy similares a los de un sitio web dinámico común, pero caracterizado por el tipo de uso que haga cada usuario. Como se ve, siempre está el juego el comportamiento aleatorio de los usuarios.

Por ejemplo, un perfil de usuario que dedica mucho tiempo a publicar y no a consultar publicaciones, no va a consumir demasiados recursos de acceso a disco o bases de datos, ya sean locales o distribuidas. Sin embargo, va a ir añadiendo contenidos y ocupando almacenamiento en disco. Si los contenidos publicados son archivos de cualquier tipo, también se va a consumir un ancho de banda en la red durante la carga de dichos archivos.

Por otra parte, los perfiles de usuario que consultan muchas publicaciones de diferentes usuarios con los que tiene relación, o simplemente contenidos públicos, no va a consumir espacio de almacenamiento. A cambio, va a generar carga en los procesos de acceso a disco o base de datos, y como consecuencia, consumo de ancho de banda de la red.

La tabla 1 resume el nivel de consumo de recursos por cada tipo de usuario.

Tabla 1- Consumo de recursos por perfil de usuario

	Almacenamiento en Disco	Inserciones BBDD	Consultas BBDD	Red
Generador de contenidos	Alto	Alto	Bajo	Medio
Usuario social	Medio	Medio	Medio	Medio
Lector de contenidos	Bajo	Bajo	Alto	Alto

3. Estado del arte

Dado que las aplicaciones basadas en la Web que tanto se están imponiendo en la sociedad están en aumento y en especial las Redes Sociales, es evidente que es interesante conocer qué necesidades de recursos se van a necesitar de antemano para evitar la saturación parcial o total del servicio.

A diferencia de un sitio web común o una web con contenidos dinámicos gestionados por una persona o equipo de personas en las que se conoce de antemano el contenido que va a manejar, una red social puede crecer de forma descontrolada debido a la aparición de nuevos usuarios, al uso que ellos hagan de la misma o simplemente, a un *boom* de popularidad. Por este motivo entre otros, disponer de un entorno de red social para estudiar y analizar el impacto que puede tener sobre una infraestructura determinada es de gran importancia hoy en día.

Son varias las soluciones o proyectos que se están llevando a cabo respecto a la creación de datos sintéticos y el análisis de prestaciones de las redes sociales. Se pueden distinguir dos aproximaciones: generación de datos sintéticos de redes sociales mediante un banco de pruebas y la simulación del funcionamiento de la red social con datos ya pre-cargados.

El caso que estamos tratando se centra más en el ámbito de la generación de datos sintéticos de forma que se consiga crear una base de datos de red social apta para la ejecución de *benchmarks* que sean capaces de tomar la información necesaria para analizar el comportamiento del sistema y las necesidades software y hardware que más interese mejorar para el rendimiento.

Los diferentes casos de estudio que nos podemos encontrar, tratan de encontrar un modelo de generación de datos ficticios para lograr una carga lo más real posible. No obstante, ninguno ofrece una solución completa. A continuación se detallan algunos de los casos de estudio que existen:

SONETOR

SONETOR [1] es un generador de tráfico sintético para redes sociales, con la finalidad de modelar estadísticamente la interacción de los usuarios en la red social mediante la generación de secuencias de actividad. El modelado de generación de tráfico utiliza un grafo de interacción entre usuarios, la interacción definida por el tiempo de inactividad entre acciones y un modelo dependiente de las acciones previas realizadas por el usuario. Haciendo uso de estos grafos, genera unas trazas sintéticas de actividades de los usuarios, consiguiendo así un patrón de comportamiento a analizar.

El objetivo de este *benchmark* es poder estudiar el impacto de una red social sobre la forma y

frecuencia de acceso a los contenidos. Mediante las leyes de Zipf[3] se genera la distribución de la probabilidad de acceso a un contenido, obteniendo así la popularidad de los mismos. A partir de la popularidad, se plantean mejoras de acceso a datos para *datacenters* o redes de contenidos, ya sean centralizadas o distribuidas.

A pesar de su componente estadística el análisis es interesante, pero se ha detectado la carencia de poder determinar el grado de datos a analizar. En el estudio, se utiliza un conjunto de datos estático definido con un determinado número de usuarios y relaciones entre los mismos. La posibilidad de definir el número de usuarios de forma directa y de forma parametrizable la cantidad y el tipo de contenidos que va a tener cada usuario es muy interesante, ya que va a permitir modelar diferentes redes sociales, con mayor o menor popularidad. En otras palabras, permite confeccionar un escenario a medida del caso de estudio que se desee llevar a cabo.

Social Network Intelligence Benchmark

SNI Benchmark [2] es un *benchmark* creado a partir de la necesidad de obtener una forma de simular y generar datos de análisis de redes sociales. La iniciativa que motivó a su creación, se debe a que no existía un modelo bien definido de estos simuladores, en contraposición a los simuladores de sistemas basados en Web con modelos de datos mejor establecidos.

El trabajo de este estudio pasa por la creación de unos datos sintéticos sobre los que realizar las simulaciones. Según se propone, el número de usuarios será definido en la creación de los datos. Además, el conjunto de contenidos creados escalará de forma directa con el número de usuarios. Es decir, a mayor número de usuarios, mayor número de contenidos. Dicha información se tomará aleatoriamente de diccionarios de datos seleccionados para la creación de usuarios.

También las relaciones entre usuarios tendrán un papel importante en la cantidad final de datos. Estas relaciones de considerarán simétricas, es decir, los usuarios serán "amigos" entre sí. Se propone que el número de publicaciones de cada usuario se obtendrá mediante un cálculo que tendrá como parámetros el tiempo transcurrido desde la última publicación, y el número de usuarios con los que se está relacionado.

Se puede ver que en este trabajo ya se está cubriendo la necesidad de generar un modelo de datos, que se podrá definir en cierta medida respecto al tamaño final de los datos. El estudio que en este trabajo se cubre, genera en cierta medida un modelo de datos similar, pero con algunos matices diferentes. Según SNI[2], la cantidad de contenidos vendrá determinado en cierta medida por la cantidad de usuarios existentes. En cambio, el trabajo que se ha llevado a cabo en esta tesina, deja a libre elección del usuario definir la cantidad máxima de contenidos de cada tipo.

4. Generación sintética del frontend del banco de pruebas

Vistas las necesidades de existen a la hora de modelar una aplicación de Red Social para la simulación del funcionamiento, se propone un generador de datos sintéticos. Para ello, se ha utilizado una solución de redes sociales y una serie de aplicaciones que poblaran la base de datos de la red social.

A continuación se va a detallar el sistema que se ha instalado y se propone para obtener el entorno que se busca como objetivo en este trabajo:

4.1. El motor: elgg

Se ha elegido un motor Open Source de uso gratuito para crear la red social que hará de modelo u objeto de análisis. Se trata del proyecto *elgg*[5]. Como se ha dicho, es un motor de código abierto preparado para funcionar de forma inmediata sobre servidores web Apache con soporte a PHP y motor de bases de datos MySQL. Apenas necesita modificar unos pocos parámetros respecto a la instalación por defecto para que esta red social esté a punto para funcionar, permitiendo así una puesta en marcha casi inmediata.

La versión que se ha utilizado se trata de la actual *release* del proyecto en el momento de comenzar la preparación de esta herramienta: elgg - 1.8.17.

Los requisitos básicos que deben estar disponibles en el sistema son:

- Servidor web Apache con módulo “mod_rewrite” cargado.
- PHP 5 instalado como módulo de Apache y cargado.
- Biblioteca GD y Freetype de PHP 5.
- Biblioteca JSON de PHP 5.
- Biblioteca XML de PHP 5.
- Soporte *Multibyte String* para PHP 5.
- MySQL 5 o superior.

Se recomienda también realizar las siguientes modificaciones en la configuración base de PHP 5 modificando el fichero de configuración php.ini:

- Ampliar la memoria de los hilos de ejecución, de 8M a 12M.

```
memory_limit = 12M
```

- Ampliar el tamaño máximo de los ficheros que se permite cargar, de 2M a 10M.

```
upload_max_filesize = 10M
```

Este motor funciona a base de módulos también de código abierto, pudiéndose activar y desactivar, instalar e incluso desarrollar nuevos módulos. Esta característica es un punto a favor de la elección del motor, ya que permite dotar de las características deseadas a nuestra red social, incluso si se trata de una idea que no ha sido desarrollada todavía por nadie.

Con la correcta activación e instalación de estos módulos se consiguen muchas de las características que en este documento se recogen como esenciales para una red social. Así pues, se ha podido generar una red social válida para el estudio que se desea llevar a cabo.

De todas las características de las que se pueden encontrar en una red social, actualmente nos centramos en un subconjunto de ellas, cubriendo las que la mayoría de autores identifican como elementales para una red social. Son las siguientes:

- Autenticación de usuarios
- Relaciones entre usuarios
- Publicaciones en formato texto
- Publicaciones en formato texto con enlaces externos
- Publicación de contenidos en forma de imágenes

Se ha tenido que instalar el módulo *Tidypics Photo Gallery* en su versión 1.8.1 para conseguir el soporte de galerías fotográficas y así cubrir la característica de publicación de imágenes. El resto de características tienen soporte nativo en la aplicación, únicamente habiendo sido necesario desactivar las funcionalidades no deseadas.

Los módulos que necesariamente deben estar activos son:

- Tidypics Photo Gallery
- Blog
- File

- HTMLawed
- Invite Friends
- Likes
- Members
- Message Board
- Messages
- Notifications
- Profile
- Search
- Ads Rotator

Los módulos que se han tenido que desactivar:

- User Validation by Email

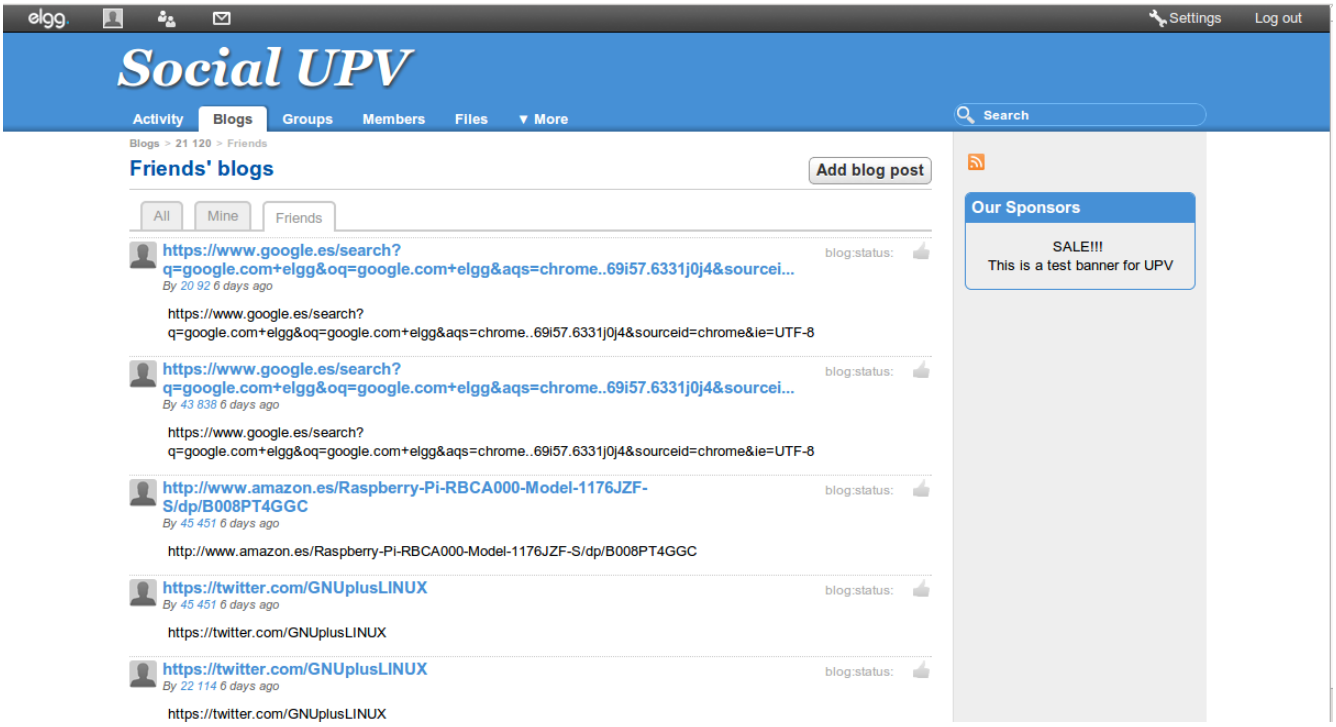

Figura 1-Página principal de la red social con datos sintéticos

4.2. La base de datos

Como se indica en el punto 4.1, el motor funciona haciendo uso de una base de datos MySQL. Aquí se almacenarán todas actividades y eventos realizados por los diferentes usuarios. Del mismo modo, también se almacenarán los contenidos generados, a excepción de las imágenes, que se almacenarán manteniendo el formato en el sistema.

El listado de tablas que se generan con la instalación es el siguiente:

- elgg_users_apisessions
- elgg_users_entity
- elgg_users_sessions
- elgg_entities
- elgg_entity_subtypes
- elgg_entity_relationships

- elgg_metadata
- elgg_metrastrings
- elgg_river
- elgg_sites_entity
- elgg_system_log
- elgg_access_collections
- elgg_access_collection_membership
- elgg_annotations
- elgg_api_users
- elgg_config
- elgg_datalists
- elgg_geocode_cache
- elgg_goups_entity
- elgg_hmac_cache
- elgg_objects_entity
- elgg_private_settings
- elgg_sites_entity

De todas ellas, las que puede tener más importancia conocer son las que alojarán los datos básicos de cada usuario, los datos publicados, y las que registran la actividad de la aplicación (log).

Tablas de usuario

elgg_users_entity: Almacena e identifica cada uno de los usuarios de la Red Social. Entre otros, almacena un *id*, el *username* de acceso, la contraseña encriptada, idioma principal del usuario, si está desautorizado y si es administrador. Se describe la tabla en la Tabla 2:

Tabla 2 - Tabla elgg_users_entity

Campo	Tipo	Tamaño estimado (bytes)
Guid	Bigint	8
Name	Text	7 (aprox.)
Username	Varchar(128)	8 (aprox.)
Password	Varchar(32)	14 (aprox.)
Salt	Varchar(8)	9
Email	Text	15 (aprox.)
Language	Varchar(6)	3
Code	Varchar(32)	33
Banned	Enum	1
Admin	Enum	1
Last_action	Int	4
Prev_last_action	Int	4
Last_login	Int	4
Prev_last_login	Int	4
		Total por registro: 115 bytes

Tablas principales de datos publicados

elgg_entities y elgg_entity_subtypes: Mediante el campo *id*, identifican cada uno de los módulos activos de la aplicación. Este *id* servirá para relacionar los contenidos con cada módulo en el que se deberá mostrar. Se listan los campos en la Tabla 3 y 4 respectivamente.

Tabla 3 - Tabla elgg_entities

Campo	Tipo	Tamaño estimado (bytes)
Guid	Bigint	8
Type	Enum	1
Sybtype	Int	4
Owner_guid	Bigint	8
Site_guid	Bigint	8
Container_guid	Bigint	8
Access_id	Int	4
Time_created	Int	4
Time_updated	Int	4
Last_action	Int	4
Enabled	Enum	1
		Total por registro: 54 bytes

Tabla 4 - Tabla elgg_entity_subtypes

Campo	Tipo	Tamaño estimado (bytes)
Id	Int	4
Type	Enum	1
Subtype	Varchar(50)	10 (aprox.)
Class	Varchar(50)	8 (aprox.)
		Total por registro: 23 bytes

elgg_objects_entity: En esta tabla se crean los objetos. En muchas ocasiones, los campos *title* y *description* se van a encontrar vacíos, ya que el único objetivo suele ser asignar un *guid* para relacionarlos en las otras tablas. Se detallan los campos en la Tabla 5.

Tabla 5 - Tabla elgg_objects_entity

Campo	Tipo	Tamaño estimado (bytes)
Guid	Bigint	8
Title	Text	0
Description	Text	0
		Total por registro: 8 bytes

elgg_entity_relationship: Esta tabla se utiliza para establecer la relación entre los diferentes objetos o entidades de la aplicación. Se aplicará la relación definida en el campo *relationship* entre los objetos *guid_one* y *guid_two*. Estos objetos serán los definidos en la tabla *elgg_objects_entity*. Se detallan los campos en la Tabla 6.

Tabla 6 - Tabla elgg_entity_relationship

Campo	Tipo	Tamaño estimado (bytes)
Id	Int	4
Guid_one	Bigint	8
Relationship	Varchar(50)	10 (aprox.)
Guid_one	Bigint	8
Time_created	Int	4
		Total por registro: 34 bytes

elgg_metastrings: Almacena las cadenas de texto de contenidos de cada publicación. No todo son contenidos directos, sino que también se almacenan etiquetas que sirven a la aplicación para hacer referencia otros datos internos. Se detalla la tabla en la Tabla 7.

Tabla 7 - Tabla elgg_metastrings

Campo	Tipo	Tamaño estimado (bytes)
Id	Int	4
String	Text	51
		Total por registro: 55 bytes

elgg_metadata: Relaciona la referencia a los contenidos (por lo general, *elgg_metrastrings*), los módulos a los que pertenece (*elgg_entities*), el identificador de usuario al que pertenece (*elgg_users_entity*) y la visibilidad que debe tener en la aplicación. Sirve de unión entre todas las tablas afectadas para mostrar contenidos. Se detalla la tabla en la Tabla 8.

Tabla 8 - Tabla elgg_metadata

Campo	Tipo	Tamaño estimado (bytes)
Id	Int	4
Entity_guid	Bigint	8
Name_id	Int	4
Value_id	Int	4
Value_type	Enum	1
Owner_guid	Bigint	8
Access_id	Int	4
Time_created	Int	4
Enabled	Enum	1
		Total por registro: 38 bytes

Tablas de registro

elgg_river y elgg_system_log: Registran todos los accesos, acciones y publicaciones que realiza un usuario. Esta información está en un formato determinado para que la aplicación muestre en el panel de

actividad del *frontend* las últimas acciones que se han llevado a cabo. Se listan los campos en la Tabla 9 y 10 respectivamente.

Tabla 9 - Tabla elgg_river

Campo	Tipo	Tamaño estimado (bytes)I
Id	Int	4
Type	Varchar(8)	8 (aprox.)
Sybtype	Varchar(32)	8 (aprox.)
Action_type	Varchar(32)	9 (aprox.)
Access_id	Int	4
View	Text	30 (aprox.)
Subject_guid	Int	4
Object_guid	Int	4
Annotation_id	Int	4
Posted	Int	4
		Total por registro: 79 bytes

Tabla 10 - Tabla elgg_system_log

Campo	Tipo	Tamaño estimado (bytes)
Id	Int	4
Object_id	Int	4
Object_class	Varchar(50)	13 (aprox.)
Object_type	Varchar(50)	13 (aprox.)
Object_subtype	Varchar(50)	13 (aprox.)
Event	Varchar(50)	7 (aprox.)
Performed_by_guid	Int	4
Owner_guid	Int	4
Access_id	Int	4
Enabled	Enum	1
Time_created	Int	4
Ip_address	Varchar(15)	9 (aprox.)
		Total por registro: 23 bytes

4.3. Generando contenido sintético

Para generar los datos sintéticos que van a poblar la base de datos, se han desarrollado una serie de pequeños procesos (*scripts* en *bash*) que se encargarán de generar esta carga y almacenarla en la base de datos. Estos ejecutables son independientes para cada módulo, pudiéndose lanzar estos de forma separada o conjunta mediante otro proceso que los agrupe.

La metodología general utilizada para generar los datos, consiste en producir un volumen de datos aleatorio acotado por unos parámetros definidos por el usuario. Así pues, se puede decir que la generación de datos es semi-aleatoria y definida por el usuario. Esto permite generar una base de datos del volumen y las características que se deseen definir en función del objetivo de caso de simulación y posterior análisis, pero manteniendo la característica del comportamiento aleatorio de los diferentes

usuarios.

A continuación se va a detallar como actúa cada uno de los procesos:

1. Usuarios

La generación de usuarios es la única parte que no se ve afectada por el concepto "aleatorio". Se entiende que dependiendo del número de usuarios, se va a definir en gran medida la talla de la base de datos que se desea generar, motivo por el cual no genera usuarios de forma aleatoria.

Se desea generar usuarios con nombre y apellidos, con un correo electrónico determinado y una contraseña de acceso. Además, a partir del nombre y apellidos se va a generar un nombre de usuario único mediante el que se podrá realizar el *login*. Dado que se necesitaría una base de datos de datos personales infinita para generar tantos usuarios como se desee, se ha optado por sintetizar estos datos mediante números enteros.

Como parámetros de entrada se deberán definir dos enteros que generarán usuarios a modo de matriz, N y A. El número total de usuarios que se obtendrá será el resultado de multiplicar N por A. De este modo, la forma de sintetizar la matriz en datos personales es la siguiente:

- Nombres y apellidos: se crearán A apellidos para cada uno de los N nombres. Así pues, se encontrarán como nombre los números enteros de 0 a N, y como apellidos los enteros de 0 a A. Por ejemplo, se podría encontrar el usuario con nombre completo "234 729".
- Nombre de usuario único: para cada usuario, se generará la cadena resultante de concatenar la letra "u" junto con el entero que representa el nombre, el carácter ".", y el entero que representa el apellido. La expresión sería: uNi.Aj. Siguiendo el ejemplo: "u234.729"
- Correo electrónico: dado que ya se dispone de una cadena inequívoca, se utilizará como parte del usuario del correo. Ésta es el nombre de usuario generado. A esta cadena se le concatenará el carácter "@" y un dominio. El dominio viene preestablecido a "upv.es". El ejemplo que se está explicado: u234.729@upv.es
- Contraseña: en este caso, no se necesita ninguna característica especial, nada más que se cumplan los requisitos mínimos que establece el motor. Este requisito es que tenga al menos seis caracteres. Dado que no se necesita ninguna seguridad al no tratarse de un entorno con información sensible, se ha optado por predefinir a todos los usuarios la cadena: "123456".

Se muestra a continuación en la Figura 2 un ejemplo de *login* sobre el sitio web resultante:

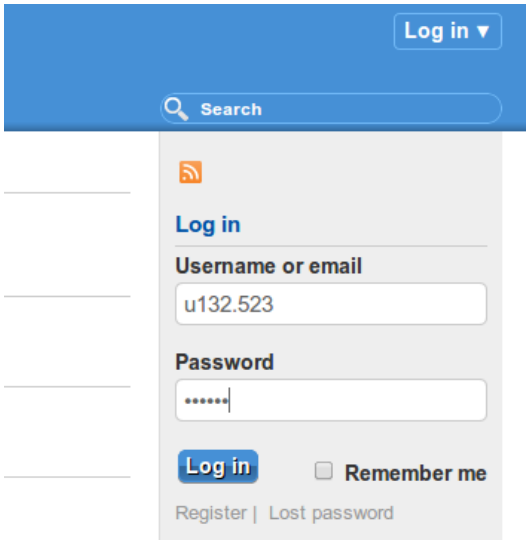

Figura 2 - Ventana de login con usuario de ejemplo

Conociendo en las tablas que se escribe, el tamaño de datos por registro de cada tabla, y el número de usuarios se puede realizar una estimación del crecimiento de la base de datos al ejecutar el proceso. Se detalla la estimación por cada usuario creado en la Tabla 11.

Tabla 11 - Tamaño de datos por usuario creado

Tabla	Número de registros	Tamaño estimado (bytes)
elgg_entities	2	108
elgg_entity_relationship	1	34
elgg_objects_entity	1	8
elgg_users_entity	1	115
elgg_metadata	1	38
		Total por registro: 303 bytes

2. Relaciones

En este módulo ya entra en juego la aleatoriedad, ya que en un entorno real no se puede conocer

la cantidad de relaciones que cada usuario va a tener con los otros usuarios. Se entiende que un usuario puede tener entre 0 y R_{max} relaciones. El motivo es que puede tratarse de un usuario sin apenas actividad que quizás no haya llegado a crear relaciones, o por el contrario tratarse de un usuario con una actividad considerable. El concepto de "semi-aleatoriedad" también entra en juego ahora, ya que se debe facilitar un único parámetro que define el máximo número de usuarios que se espera que tenga un usuario.

Una vez definido el máximo, se va a crear para cada uno de los usuarios un número de relaciones con otros usuarios entre 0 y R_{max}, siendo R_{max} el máximo definido. El valor obtenido dentro del rango será totalmente aleatorio. Las relaciones de amistad generadas serán bidireccionales, es decir, un usuario A será "amigo" de B, y a su vez, B será "amigo" de A.

De forma similar a la creación de usuarios, se detalla el espacio de datos en base de datos que se consumirá por cada relación de amistad bidireccional. Mostrado en la Tabla 12.

Tabla 12 - Tamaño de datos por relación entre usuarios

Tabla	Número de registros	Tamaño estimado (bytes)
elgg_entity_relationship	2	34 Total por registro: 68 bytes

3. Publicaciones de texto

Las publicaciones de texto también son semi-aleatorias. Dado un número máximo de publicaciones, se van a generar entre 0 y P_{max} publicaciones de texto. Con este modelo se simulará usuarios que generan mucho contenido, usuarios que apenas generarán y todos los casos intermedios.

El modo de funcionamiento del generador de publicaciones de texto, consiste en recorrer un archivo de texto con varios *megabytes* de contenido en texto plano. Se va a tomar un número fijo de 10 palabras para generar un título, y un número aleatorio entre 50 y 150 palabras para generar el contenido del post o publicación de texto. Las 25 primeras generarán un extracto que se mostrará antes de entrar en el contenido en sí.

Dado que toda la información de una publicación de texto se guardará en la base de datos, quizás sea éste el caso que más interese conocer el consumo de espacio en disco por publicación de datos. De forma homóloga a la creación de usuarios y de relaciones, se detalla este consumo en la Tabla 13. Sin embargo, se trata del caso más difícil de estimar, dado que el tamaño del registro de la tabla elgg_objects_entity, que se cargará con el título y el contenido de la

publicación, dependerá del número de palabras obtenido aleatoriamente. Además, estas palabras tendrán una cantidad de caracteres muy variado. Así pues, este tamaño se va mostrar como la variable $T_{publicación}$, y se obtendrá como sigue:

$T_{publicación} = N_{caracteres\_titulo} + N_{caracteres\_post}$

Siendo:

$N_{caracteres\_titulo}$ = Número de caracteres totales del título generado aleatoriamente

$N_{caracteres\_post}$ = Número de caracteres totales del contenido generado aleatoriamente

Tabla 13 - Tamaño de datos por publicación de texto creada

Tabla	Número de registros	Tamaño estimado (bytes)
elgg_entities	1	54
elgg_objects_entity	1	$8 + T_{publicación}$
elgg_metastrings	1	55
elgg_metadata	1	38
		Total por registro: $155 + T_{publicación}$

4. Publicación de enlaces

Para las publicaciones de enlaces, se va a utilizar el mismo formato que para publicaciones de texto, es decir, se van a publicar *posts* con enlaces como contenido. Estos enlaces se van a tomar de una batería de enlaces predefinida, que apuntarán a sitios web o servicios web de uso habitual. El número de enlaces que se publicará para cada usuario, vendrá determinado por el parámetro de entrada E_{max}, de forma que se generarán entre 0 y E_{max} publicaciones de enlaces externos.

Para obtener una estimación del tamaño de publicaciones de enlaces, haría falta conocer la el número medio de caracteres que tienen los enlaces. Si definimos este valor como $M_{caracter\_link}$, la tabla que define el tamaño estimado en disco por enlace sería la Tabla 14.

Tabla 14 - Tamaño de datos por enlaces externos

Tabla	Número de registros	Tamaño estimado (bytes)
elgg_entities	1	54
elgg_objects_entity	1	$8 + M_{caracter\_link}$ Total por registro: $62 + M_{caracter\_link}$

5. Publicación de imágenes

Una vez más, se va a generar una carga aleatoria pero limitada por un número máximo de publicaciones. Así pues, para cada uno de los usuarios registrados en el sistema se va a calcular un número aleatorio comprendido en el rango entre 0 y Imax, siendo Imax el número máximo de publicaciones que un usuario puede tener. Por lo tanto, el único parámetro que se deberá facilitar, es el número máximo de imágenes que un usuario puede publicar.

En el momento de la creación de datos, se ha barajado la idea de mantener una relación entre las relaciones de amistad que posee un usuario, y la cantidad de publicaciones de éste. Ésta es una características que se ha llevado a cabo en otros estudios, como es el caso de la parte de generación de carga de SONETOR[1]. Sin embargo, se ha descartado, dado que no se considera tan relevante la cantidad de contenidos asignados a cada usuario como la cantidad en sí de contenidos del servicio de red social. Además, es bien conocido que un usuario puede tener un comportamiento totalmente aleatorio, pudiendo tener un gran número de "amigos" pero no generar contenidos, o teniendo muy pocos amigos y generar mucho contenido, incluso público para todos los usuarios. Aquí entraría en juego los tres perfiles de usuarios descritos en el punto 2.2.

En este caso, y pese a que la imagen se almacena en disco sin hacer uso de la base de datos, también cabe tener en cuenta el consumo en MySQL. Este consumo va a ser de los más altos dentro de la aplicación, dado que se debe hacer referencia entre diferentes objetos en los que se define todo el acceso a la imagen. Quedan los detalles de inserciones en cada tabla y el consumo que conlleva en la Tabla 15 expuestos.

Tabla 15 - Tamaño de datos por usuario creado

Tabla	Número de registros	Tamaño estimado (bytes)
elgg_entities	3	162
elgg_entity_relationship	1	34
elgg_objects_entity	2	16
elgg_metastrings	5	275
elgg_metadata	6	228
		Total por registro: 715 bytes

Con todo, al final se va a conseguir una base de datos poblada con un número determinado de usuarios, y una cantidad de contenidos ceñidos a las necesidades que el *benchmark* que se desea ejecutar.

4.3. Trabajando con bases de datos existentes

El proceso de generación de datos puede ser un proceso costoso en cuanto al tiempo se refiere. Dependiendo del número de usuarios que se quiera definir, y de los parámetros de máximos insertados para la creación de cada tipo de datos, puede acabar generando procesos largos y con muchas repeticiones. Por el modelo de inserción de datos, las consultas se realizan de forma independiente en diferentes transacciones, propiciando una distribución más cercana al funcionamiento real de una red social.

Para conseguir que la red social que se crea para realizar la carga de datos sintéticos tenga una estructura concreta y bien conocida, se parte de una inyección inicial de datos. Estos datos, se han generado a partir de un modelo base y se deberán cargar.

Por estos motivos y por cualquiera que el usuario pueda necesitar, se ha creado una utilidad para gestionar este guardado y regeneración de datos, a modo de *backup*. Así pues, se ofrece la posibilidad de guardar, y cargar. Se detalla a continuación el funcionamiento del *script* de *bash* generado para este fin: `backup.sh`

Cabe destacar que el usuario con el que se ejecute este proceso deberá tener permiso de lectura y escritura en el directorio */opt* y en el directorio de publicación de contenidos web, por lo general, */var/www*.

Se deberá modificar para modificar la definición de parámetros de acceso de la base de datos. Las constantes que se deberá modificar son:

- DBNAME: nombre configurado para la base de datos (por defecto, elgg)
- DBUSER: nombre del usuario que tiene permiso de acceso de escritura y lectura en la base de datos
- DBPASS: contraseña de acceso del usuario definido en el parámetro DBUSER
- DBHOST: dirección de acceso a la base de datos (habitualmente, 127.0.0.1)

Una vez definido, se deberá ejecutar a través de una terminal junto con dos parámetros:

- action:
    - "save": para generar un nuevo paquete de backup
    - "load": para cargar en el servidor un paquete de backup existente
- filename: si la action es "save", indicará el nombre de paquete que se generará; si la action es "load", indicará el nombre del paquete proporcionado (sin extensión) para cargar el servidor.

Guardado de datos

Este proceso, copia los archivos fuente de la interfaz web de la red social, que por lo general se encontrarán en el directorio "/var/www/elgg". Hace lo mismo con el directorio en el que se encuentran los archivos de datos utilizado para la aplicación, como pueden ser archivos temporales, archivos, etc. El destino de la copia es un directorio temporal generado en el mismo directorio en el que se encuentra y ejecuta el proceso. El nombre de este directorio será el definido en el segundo parámetro.

Para la base de datos, se realiza un volcado a un archivo de texto plano en formato sql, mediante el procedimiento *mysqldump*. Este archivo se guarda también en el directorio temporal.

Finalmente, se empaqueta y comprime el directorio generando un archivo "tgz" con el nombre facilitado y añadiéndole la extensión. La ubicación, será el directorio de ejecución de la utilidad en cuestión.

Ejemplo de uso: `./backup.sh save filename`

- `filename`: No deberá contener la extensión. Se añadirá automáticamente

Carga de datos

En este caso, por simplicidad en el procedimiento, se recomienda ubicar el archivos que contiene el *backup* en el mismo directorio que la utilidad que se va a utilizar. También se puede ubicar en cualquier otra ruta, pero se debe facilitar el nombre con la ruta, y tener en cuenta los permisos de escritura en dicho directorio. El nombre (o ruta) del archivo se debe insertar como segundo parámetro sin escribir la extensión, que deberá ser "*.tgz*".

El proceso comenzará por descomprimir el paquete facilitado en el mismo directorio en el que se está ejecutando el proceso, creando un directorio temporal con el nombre del paquete. A continuación, se ejecuta el archivo *sql* con la base de datos y se carga en el sistema de bases de datos configurado. Se continúa copiando los directorios de datos y de fuentes, asignando los permisos necesarios para que se pueda ejecutar sin problemas la aplicación.

Para finalizar, se limpia el directorio temporal manteniendo el archivo con el *backup* comprimido, para poderlo ejecutar de nuevo si fuera necesario.

Ejemplo de uso: `./backup.sh load filename`

- `filename: No deberá` contener la extensión. El archivo deberá tener la extensión ".tgz"

Uso del proceso

Una vez conocido el funcionamiento del proceso de carga de datos, falta ver los usos que se le pueden dar:

- Se deberá utilizar de forma casi obligatoria el proceso tras la instalación del motor, dado que no tendrá datos cargados. Para ello, se recomienda vaciar la base de datos que genera el proceso de instalación de "elgg". Con este fin se utilizará el *script* de vaciado de base de datos.

  A continuación, se realizará una carga del paquete que se proporciona como carga básica, es decir, sin datos. Este paquete únicamente tendrá la configuración inicial. Para ello, se utilizará el proceso backup.sh, en su opción "load" y proporcionando como parámetro el nombre del paquete. Al final el proceso, se tendrá una instalación básica.

  Para finalizar, se deberá ejecutar cada uno de los procesos que se han detallado en este

documento para obtener el volumen de datos que se desee.

- Una vez se ha generado el escenario de red social deseado, es muy posible que se desee guardar para volver a recuperarlo tras realizar experimentos o simulaciones. Por lo tanto, una vez finalizados los procesos de llenado, se deberá ejecutar un guardado de la base de datos. Para ello se utilizará el proceso backup.sh en su opción "save" proporcionando un nombre para el archivo que se generará con los datos. Se debe tener en cuenta que dependiendo de los datos cargados, el archivo generado puede ser muy pesado y emplear mucho tiempo en generarlo.

- Si se ha modificado el escenario inicial y se desea volver al generado, o bien se desea replicar una instalación, solo se necesitará ejecutar el proceso backup.sh en modo load, de la misma forma que en el primer punto, donde se realiza una carga de solo la estructura. Se debe tener en cuenta que la base de datos se debe vaciar mediante el script de vaciado.

5. Administración

Una vez instalado y configurado el entorno de red social junto con los datos cargados, estaremos ante un entorno completo, que se va a necesitar administrar tanto para visualizar y controlar los datos cargados, como para modificarlos.

El método de administración del entorno es mediante el mismo *frontend*, siendo solamente necesario hacer *login* con un usuario de tipo administrador. Por defecto, en la carga de datos limpia se define un usuario para esta finalidad. Las credenciales son:

- Nombre de usuario: **admin**
- Contraseña: **123456**

Además, cuando se realiza la carga de usuarios, el primero que se creará (u0.0) se definirá como administrador. Las credenciales de este usuario seguirá el mismo patrón que el definido para todos los usuarios creados con el *script* de carga:

- Nombre de usuario: **u0.0**
- Contraseña: **123456**

Una vez accedido con un usuario administrador, se tendrá que acceder al panel, haciendo uso del enlace "Administration" que se encuentra en la parte superior derecha de las páginas web de la red social. La Figura 3 muestra el enlace mostrado en la pantalla inmediata al proceso de login cuando un usuario es de tipo administrador:

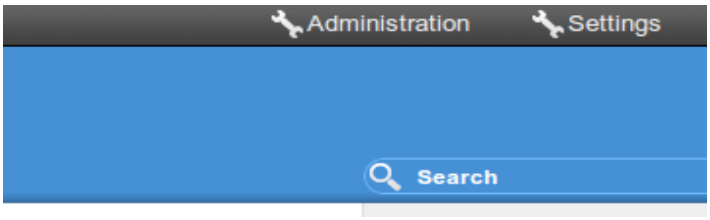

Figura 3- Acceso al panel de administración

Con esto se accederá a la página de inicio de administración. Este panel es bastante sencillo. En la parte derecha se encuentra un menú desplegable con todas las secciones que se pueden administrar. En la parte central van apareciendo los paneles con los que gestionar o visualizar cada apartado.

Una de las características que puede ser de especial interés, es la parte de administración de *plugins* instalados. Mediante este panel, se va a poder activar y desactivar los módulos instalados de forma

simple. Se muestra en la Figura 4 este panel, que también es una muestra válida de la vista de administración.

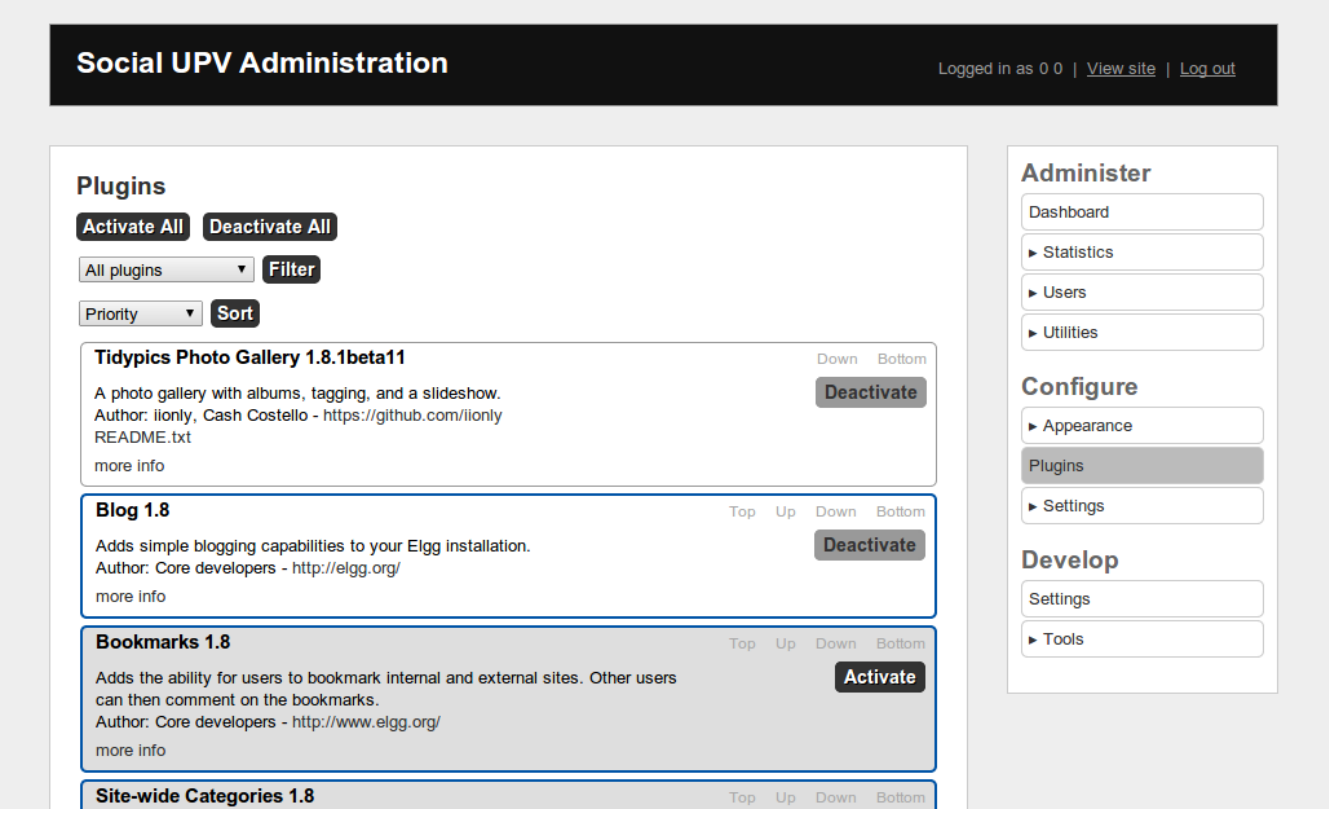

Figura 4- Administración de ELGG, plugins.

El propio motor de la red social dispone de una amplia documentación en la que se detalla toda la gestión de la red social generada. Para consultarla nada más hay que acceder a la página de documentación de administración de ELGG [6] que se puede encontrar en la red.

6. Estado

La vista web de cualquier usuario será sencilla en cuanto a estilos de apariencia se refiere. Sería completamente personalizable, pero teniendo en cuenta que el objetivo de esta tesina es obtener un banco de pruebas para procesos de análisis, queda en un segundo plano la dedicación de esfuerzos en obtener una apariencia atractiva.

Para acceder, no habrá más que hacer login como se describe en la sección 4.2 y se muestra en la Figura 2.

La forma que tendrá el *frontend* de toda la red social es el siguiente:

- Barra de navegación superior. En la que se encuentra en la parte izquierda el acceso al perfil del usuario, las relaciones de amistad y los mensajes privados. En la parte derecha se encuentran los enlaces de administración del perfil del usuario con el que se ha accedido, y en enlace para salir de la sesión en curso.

- Cabecera. Muestra un menú para navegar por las diferentes funcionalidades activadas. También dispone de un formulario de un único campo para realizar búsquedas de contenidos dentro de la red social.

- Menú dinámico. A la derecha del cuerpo, hay un menú que va cambiando dependiendo de la sección en la que se encuentre la navegación. En el entorno que se ha montado, también se muestra un pequeño *frame* que hace de *banner*. Este banner es un módulo que se puede configurar, al igual que el resto, desde la parte de administración, descrita en el punto 5.

- Cuerpo de la web. En el centro se mostrará la información relativa a la funcionalidad a la que se haya accedido. Por lo general, esta vista dispone de pestañas para moverse entre diferentes agrupaciones de los contenidos.

Como ejemplo, se muestra a continuación algunas de las diferentes vistas que se podrán encontrar en el *benchmark* que se genera tras llevar a cabo la carga mediante un backup o tras la ejecución de los diferentes procesos de generación de datos.

Publicación de enlaces

A continuación se muestran las publicaciones de enlaces. Como se ha comentado, la forma de crear estos enlaces es mediante la generación de *posts* cuyo contenido son enlaces externos. En la Figura 5 se puede ver un listado de *links* de los amigos del usuario registrado:

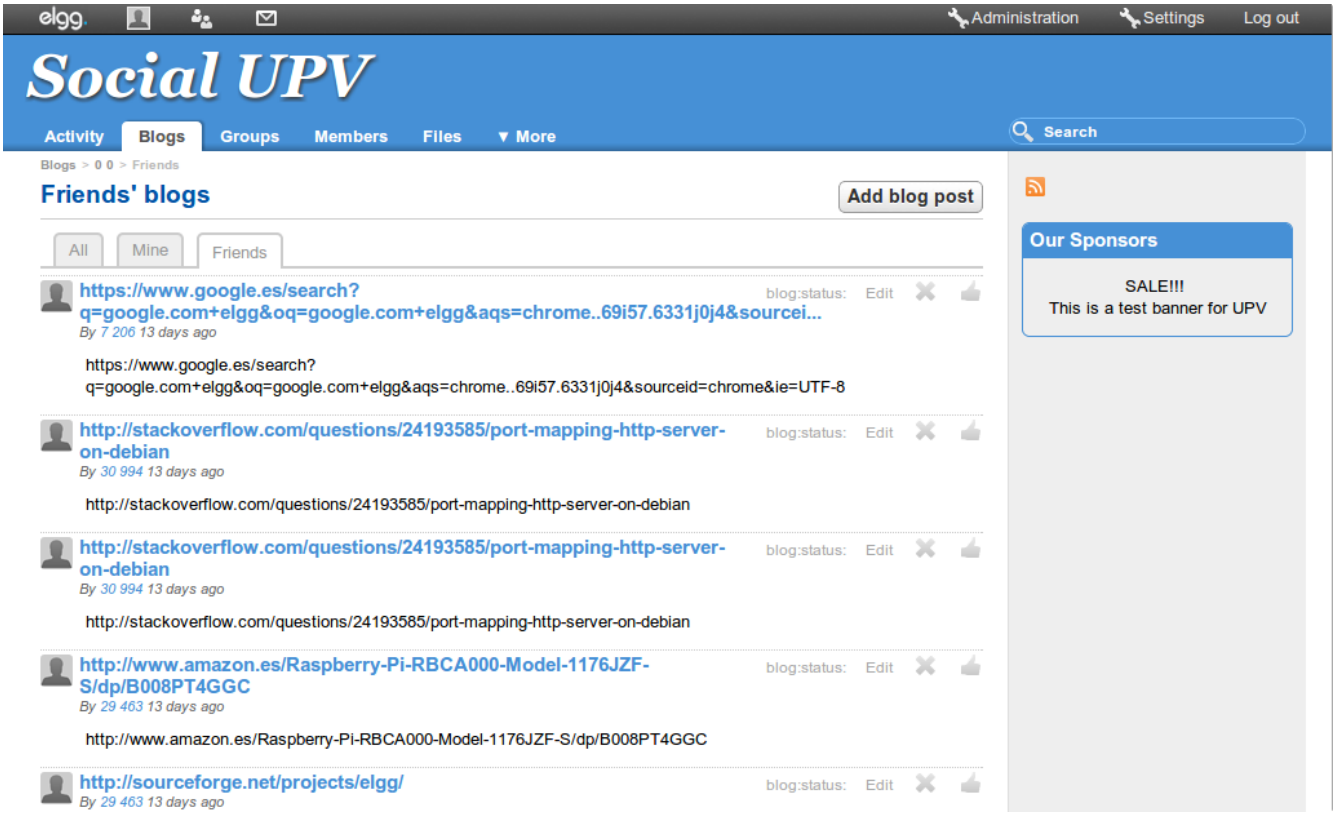

Figura 5- Blogs de enlaces

Publicación de texto

De igual forma que en el punto anterior, se listarán los blogs con contenido de texto. Al acceder a la visualización de cualquiera de ellos, la vista que se mostrará será la de la Figura 6.

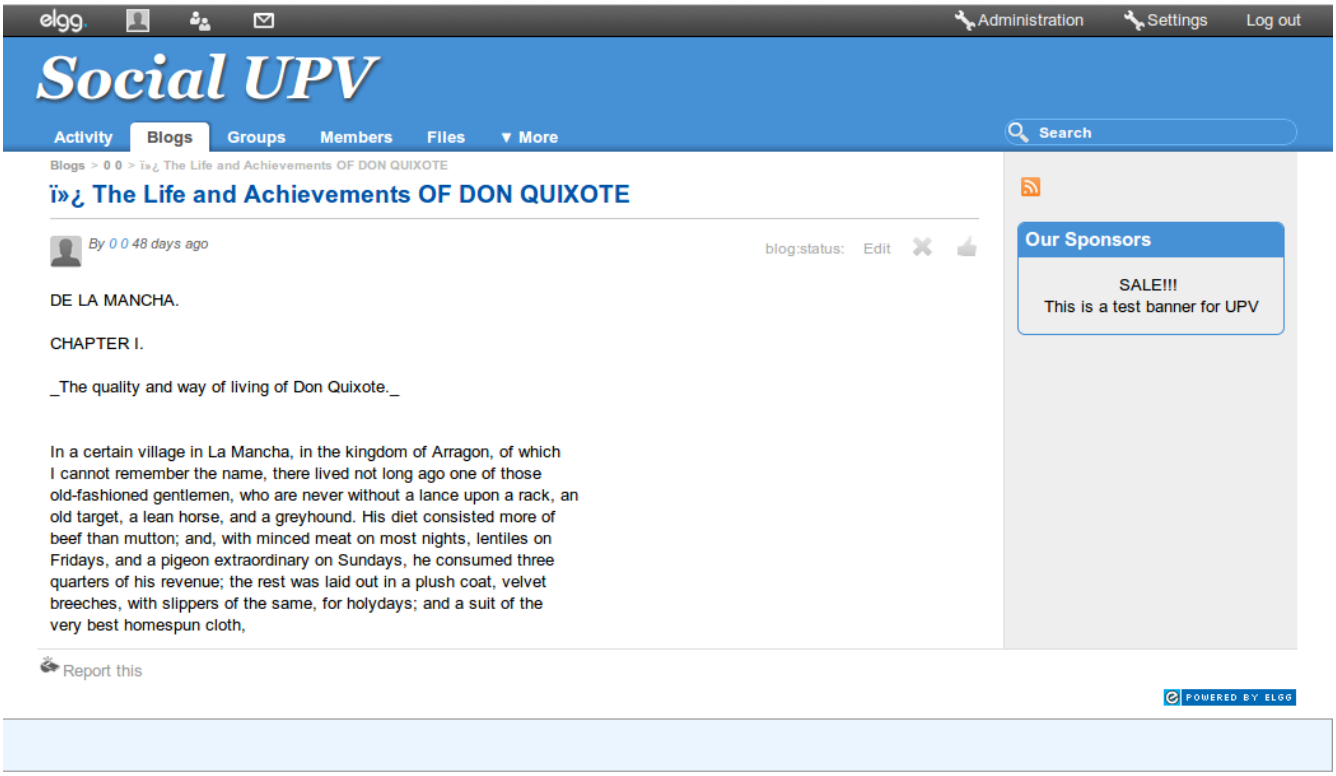

Figura 6 - Vista de publicaciones de texto

Publicación de imágenes

Para listar las imágenes, deberá acceder al enlace de “Photos”, que se encuentra en el menú de la cabecera, al desplegar el enlace “More”. La vista que se obtendrá será la mostrada en la Figura 7:

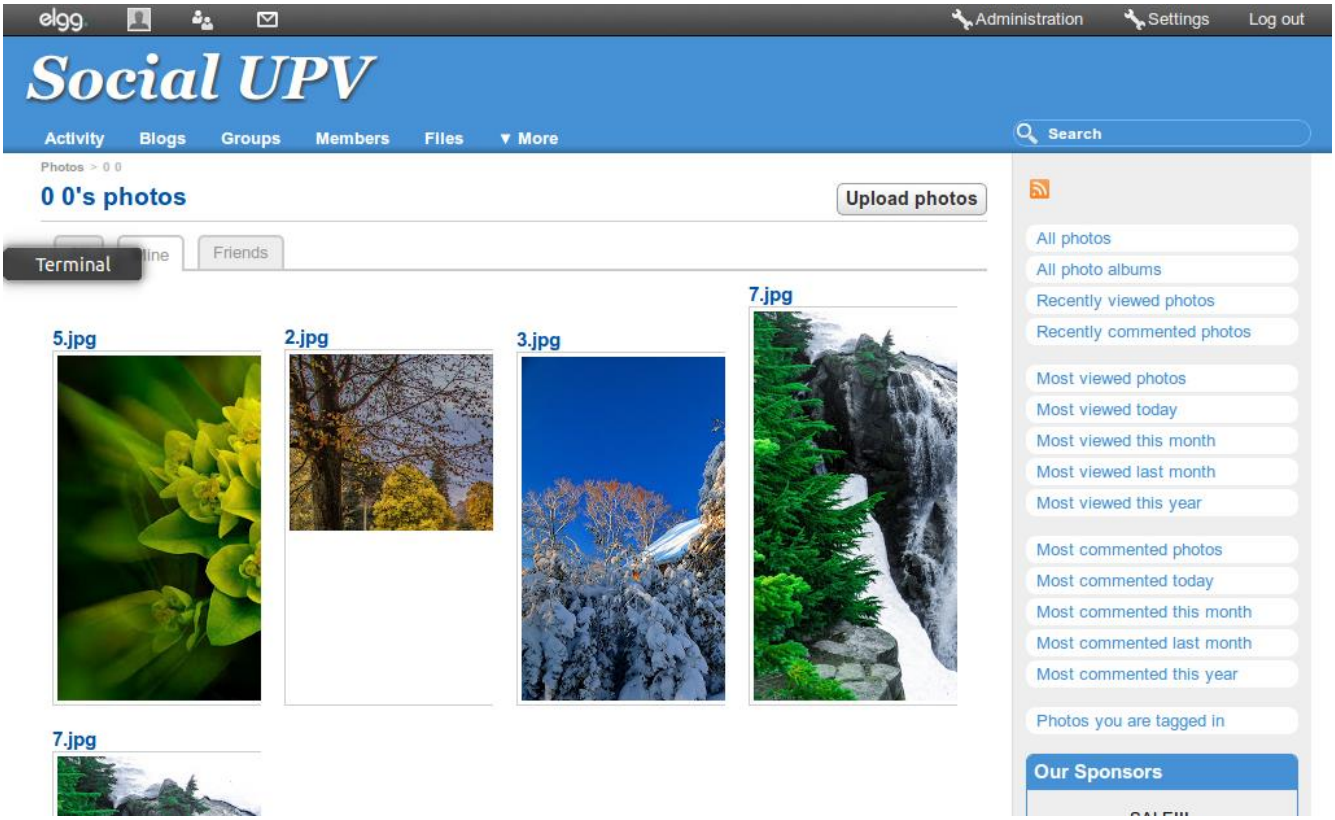

Figura 7- Vista de imágenes

Relaciones de amistad entre usuarios

Mediante la barra de navegación, se puede acceder al panel de relaciones de amistad. Por defecto se accederá al listado de amigos del usuario registrado, como se puede ver en la Figura 8.

Haciendo uso del menú de la derecha, se puede acceder a otros listados, como por ejemplo, al "Friends of" mostrado en la Firgura 9. En él se muestran los usuarios que tienen al perfil registrado añadido a la lista de amigos.

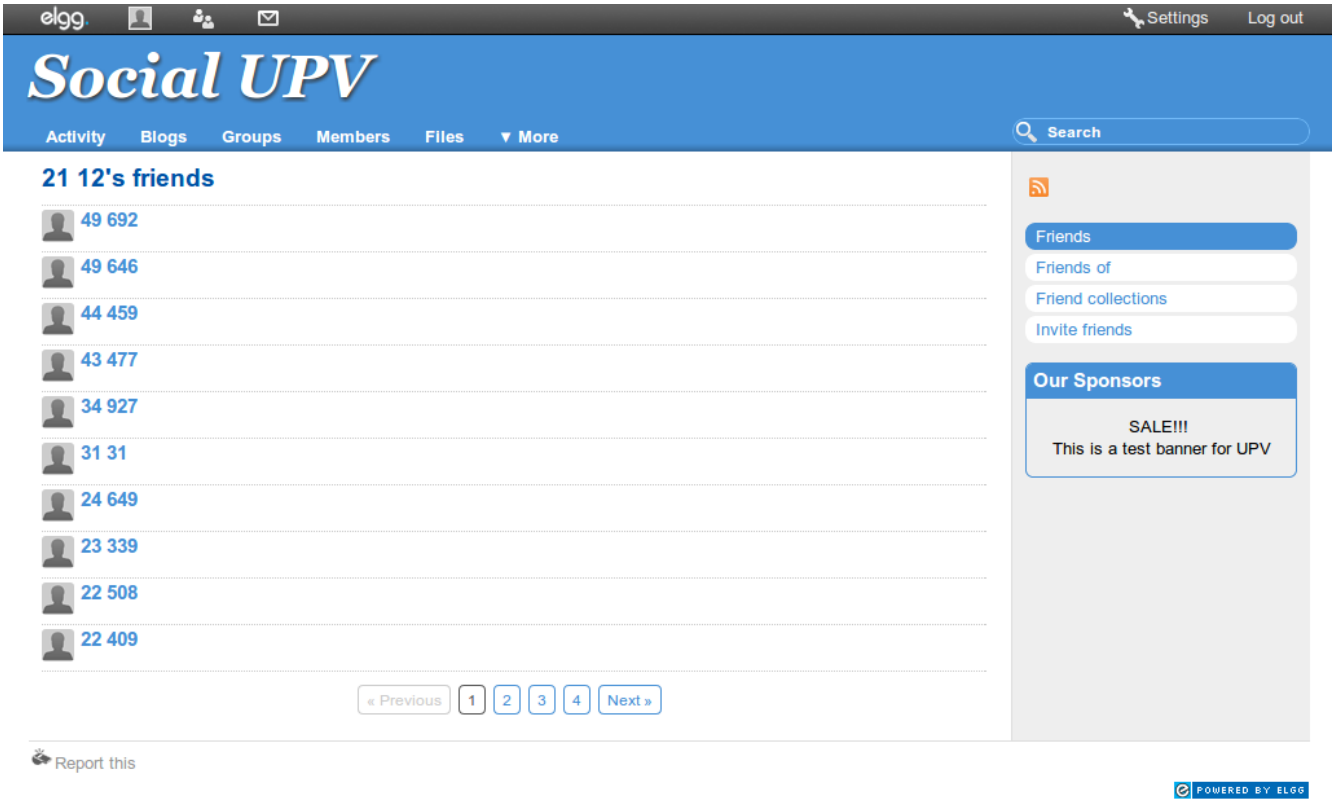

Figura 8- Amigos de un usuario

Figura 9- Usuarios de los que se es amigo

7. Conclusiones y trabajo futuro

Con todo lo expuesto, se pretende defender que la generación de un entorno de red social con datos previos cargados es una tarea esencial para la correcta ejecución de un *benchmark* para aplicaciones de este tipo. Como en un principio se expone, la intención es mejorar la actual oferta de herramientas que sirven para simular entornos de aplicaciones de redes sociales. Por lo que con esta tesina se pretende proporcionar el valor añadido de poder personalizar y el escalar en la medida que se desee el entorno base generado con datos sintéticos. Para este fin, se ofrece la personalización mediante la adición o extracción de módulos para adaptar el comportamiento a cualquier modelo que se desee analizar; y la escalabilidad a mediante los módulos de generación de carga, ya que se permite generar la cantidad y el tipo de datos deseados.

Con el trabajo que se ha realizado, tal y como se ha ido explicando, se obtiene una red social funcional y totalmente cargada con datos sintéticos, en resumen, un banco de pruebas para la ejecución de procesos de carga y rendimiento. Esto va a permitir utilizar herramientas que generen carga de trabajo para extraer datos del comportamiento del sistema, y así poder realizar estudios que permitan diseñan de forma analítica un entorno óptimo para las necesidades que se desea cubrir. Así, se pretende evitar costes y problemas de redimensionado o modificación cuando el servicio ya está en funcionamiento.

Así pues, esto sirve de motivación para continuar con la parte de análisis. Esta parte debería simular comportamientos aleatorios de los usuarios, siguiendo el patrón de este documento, de forma parametrizable. Es decir, poder definir para las simulaciones tiempo de navegación, número de accesos a datos o ambos. De esta forma, también se podrían generar casos de test con grados de acceso similares a los previstos según la generación de carga. No obstante, esta parte se deja abierta a futuros trabajos, al igual que este, con la intención de ampliar una oferta que todavía no está lo suficiente desarrollada.

8. Referencias

[1] SONETOR: a Social Network Traffic Generator

http://www.loria.fr/~bernardc/icc2014-bernardini-silverston-festor.pdf

[2] Social Network Intelligence Benchmark
http://www.w3.org/wiki/Social_Network_Intelligence_BenchMark

[3] Zipf's lay

http://en.wikipedia.org/wiki/Zipf's_law

[4] Benchmark

http://es.wikipedia.org/wiki/Benchmark

[5] ELGG

http://elgg.com

[6] Documentación de Administración de ELGG

http://docs.elgg.org/wiki/Main_Page#Administration

9. Referencias a figuras