Escuela Politécnica Superior

Departamento de Informática

Proyecto de fin de carrera.

ADAPTACIÓN Y DESPLIEGUE DE RED SOCIAL EN ALTA DISPONIBILIDAD

Generador de Redes Sociales Geoposicionadas

Por Arturo Ros Bravo

Tutelado por el Dr. Antonio Berlanga de Jesús

SEPTIEMBRE DE 2015

UC3M

Colmenarejo

Abstracto.

Dado el aumento de las opciones de infraestructura para servicios web y la competencia agresiva en precios tanto en el proveedor de servicio como en la demanda del mismo, se hace necesario un replanteamiento de los modelos de despliegue de servicios web.

Este proyecto tiene como objetivo dar una visión del panorama en servicios actual y venidero encuadrándolo con las tecnologías de servicios web actuales. Tomando todos los casos de implementación en consideración y con la intención de dar una guía en función del producto y las necesidades que este tenga.

Durante el proyecto se expondrá un posible caso de desarrollo de una red social geoposicionada y como podría desarrollarse la adaptación a un entorno de alta disponibilidad.

Palabras clave.

Redes sociales, IaaS, PaaS, SaaS, cloud, web services, alta disponibilidad, autoescalado, geoposicionamiento.

Agradecimientos.

Quería dar mis más sinceras gracias a muchas personas, pero resumiré un poco:

Javi y Fer, por contar conmigo para llevar a cabo esta idea.

Antonio Berlanga, por la paciencia a lo largo de mis ausencias.

Mis padres, Paloma y Víctor, por todo el apoyo, ánimo y paciencia a lo largo de estos años.

Guille, Adrián, Carlos, Alberto, Nacho y Alí por esos momentos que han hecho que llegue hasta este momento y lugar.

Y en resumen, a la UC3M, por sus profesores, sus clases, su filosofía y su gente, que a lo largo de estos años me ha enriquecido tanto en tantas cosas.

Contenido.

Tabla de Ilustraciones.

Tablas.

I. Introducción general.

1. Propósito.

El objetivo de este proyecto es la puesta en marcha de una red social geoposicionada. Dado que el núcleo de la red social ya se gestó en el proyecto de fin de carrera de Javier Iglesia Sabio "Diseño y desarrollo del software del servidor para un sistema de gestión de redes sociales geoposicionadas". En este proyecto se tratarán tanto la mejora del front-end de administración, como la de la arquitectura desplegada para dar soporte a todo el entramado.

Este proyecto entra dentro de un conjunto de tres partes que formarían un Generador de Redes Sociales Geoposicionadas. La primera, desarrollada por Javier Iglesia Sabio, incluiría el núcleo y back-end del servicio. La segunda, desarrollada por Fernando Tomé Bernal, la parte de front-end en dispositivos móviles. Y, finalmente, una última parte de front-end para la administración y despliegue final del producto. Todas las partes quedaron definidas y estructuradas durante la concepción del proyecto, con el objetivo de que fuesen presentables independientemente, sin necesidad de las secciones de cualquiera de los otros dos compañeros.

El deseo al realizar este proyecto es dar un ejemplo base de cuál debería ser la implementación final de la red social. Tanto a nivel de hardware, como de estructura de servidores. También mostrando que camino hay que seguir con el objetivo de crecer en el desarrollo de la plataforma manteniendo un consumo de recursos más óptimo.

El cliente objetivo para este proyecto, es aquel que busque desplegar una red social cerrada o abierta que pueda controlar de forma personal tanto para clientes externos como para uso interno. En este proyecto de fin de carrera no se entrará a valorar todos los posibles usos que se le pueda dar a la plataforma, pero si se darán algunos ejemplos con el objetivo de ilustrar mejor algunos casos.

2. Estructura de la memoria.

Estos son los puntos que componen la estructura de este documento y que detallan como se ha llegado al resultado final del proyecto.

- Descripción del contexto actual de las tecnologías aplicadas.

La intención es la de contextualizar el proyecto en el estado del arte de las áreas implicadas en su desarrollo. Por ello se describirá el panorama actual (2015) en arquitectura de servicios, hosting y plataformas en la nube.

- Selección de la implementación adecuada para el servicio.

Tras la exploración de las opciones y alternativas se elegirá la forma más adecuada en función de los requisitos y los costes que puedan suponer. Siempre teniendo en cuenta la escalabilidad del servicio y la opción a que terceros puedan hacer uso de él si se desea.

- Detalle del proceso de implementación.

Una vez seleccionados todos los pormenores, se explicará al detalle el proceso de implementación y adaptación realizado.

- Documentación extra para futuros desarrollos.

Con el objetivo de dar soporte y documentación a futuras iteraciones en el servicio, se documentará todo dato que pueda ser relevante y útil para cualquier persona que decida continuar con este proyecto.

3. Descripción del proyecto.

El proyecto pretende ofrecer una vista sencilla y directa sobre un proyecto que se compone de dos partes. Una inicia de desarrollo web y una segunda de optimización de recursos en infraestructuras en la nube.

La parte de desarrollo web es necesaria para dar una visión de cómo el desarrollo de front-ends puede ser ideado o diseñado con el objetivo de encapsularse, y poder dar opción a trabajar de forma aislada sin necesidad de coexistir con el back-end en la misma máquina, red o incluso plataforma. Dar una visión de cómo las tecnologías de direccionamiento, de almacenamiento y de entrega de contenidos ha avanzado hasta abrir unas opciones a los desarrolladores muy amplias.

Por supuesto, como parte de las premisas del proyecto se basa en la optimización de recursos a nivel de infraestructura, las decisiones con respecto al front-end de administración han sido en la dirección de requerir el mínimo uso de funciones de servidor que no sean peticiones a la API. Como extra a esta premisa surgió la inquietud de poder desarrollar un front-end que pudiera ser desplegado en cualquier servidor web (Apache, NGINX, IIS, Tomcat,…) y esto podrá sustentarse gracias a la API REST. API diseñada en el principio del proyecto tomando ejemplo de muchas de las APIs públicas que se pueden ver por internet (twitter, flickr, OneDrive, Dropbox,…). Gracias a ese ejemplo se podrá desarrollar lo que veremos más adelante que nos permite crear un front-end que utilice exclusivamente HTML5, CCS3, JS y JQuery para poder mostrar los retornos de la API REST de la red social geoposicionada.

La parte de implementación y despliegue es la que más se centra en las ventajas de los nuevos servicios que proporciona "La Nube". Los hosting PHP/ASP, los Datacenter particulares de empresas y los grandes proveedores a medida han llevado el peso de todos los niveles de desarrollo en los últimos 10-15 años. Ahora, la tecnología de virtualización de hardware y el hardware existente se han juntado para proporcionar una infraestructura que pueda tener lo mejor de esos tres sectores. Una infraestructura que puede crecer y decrecer tanto en vertical como en horizontal. Donde actualmente el mayor reto es saber diseñar bien el código para que se adapte mejor a los servicios más jugosos y poder conseguir un retorno de la inversión aún mayor. Y por esto, la

segunda y más importante parte del proyecto se centra en este aspecto.

Crear la plataforma más óptima para el producto y que esté preparada para poder ajustarse a las necesidades cambiantes de internet. Ese es el objetivo a marcarse en la segunda parte del proyecto y para ello estudiaremos las distintas alternativas que ofrece el mercado. En la actualidad, el poder lanzar una aplicación está al alcance de cualquiera que tenga conocimientos y una tarjeta de crédito. La única barrera para mantenerse, es saber que infraestructura montar para no ahogarse en costes. Por suerte, las grandes empresas del sector cada día ofrecen nuevos servicios en todos los estratos de la arquitectura, y esto da la facilidad al cliente de poder encontrar el producto que mejor se adapte a sus necesidades. Crear instancias docker, máquinas virtuales completas, servicios de CDN, o almacenamiento de datos de uso inmediato o con retador. Las posibilidades y opciones crecen cada día y con ello las oportunidades de mejorar el servicio y rentabilizarlo aún más.

Pero para rentabilizarlo hay que tener una idea o en este caso, la idea de licenciar la plataforma. Y en este caso, la segunda opción sería la más interesante dadas las altas opciones de personalización que tiene el conjunto del proyecto. La idea final de ejecución sería una en la que un cliente busque dar forma a una red social personal, cerrada o abierta, que lleve su imagen de marca y proporcione servicios a sus empleados o clientes. Para ello esta red social tiene una particularidad muy ventajosa, es como una página en blanco. Tanto por la parte del front-end, altamente y fácilmente personalizable, como por la sencillez de inserción de plugins al back-end.

Con todo esto sobre la mesa, los objetivos que marcan al proyecto son estos:

- Creación de un front-end para uso administrativo del sistema.
    - Independiente de tecnologías de servidor.
    - Usando los estándares más actuales de desarrollo web.
    - Incorporando herramientas que aporten valor a la experiencia del usuario.
    - Explotando las bondades de las API REST públicas.

- Analizar y establecer una arquitectura básica para la plataforma.
    - Minimizando recursos y permitiendo que estos puedan escalar fácilmente.
    - Explorar futuros servicios de conveniencia.

4. Objetivo personales.

Como parte del proceso de realización de este proyecto, hay un deseo de explorar y conocer nuevas tecnologías, servicios o ideas que me puedan ser en cualquier momento de utilidad en el futuro de mi actividad laboral. Por ello los objetivos personales marcados para este proyecto de fin de carrera son los siguientes:

- Estudio y aprendizaje de Arquitecturas de red.
- Estudio y aprendizaje de lenguajes de programación y diseño web
    - HTML5 y CSS3
    - JavaScript y jQuery
- Estudio y aprendizaje de comunicación con APIs REST y RESTFULL.
- Estudio y comprensión de servicios de virtualización.
    - Estudio y comprensión de los distintos servicios.
    - Estudio de los pros y contras de cada servicio.

II. Estado del Arte.

El objetivo de esta sección es presentar el contexto actual de las tecnologías implicadas en el desarrollo del sistema descrito. Para ello esta sección se divide en dos partes; la primera trata el contexto y la situación actual de las aplicaciones en la nube y la segunda se centra en la situación actual y en los recursos disponibles para la creación de Servicios Web.

1. Arquitectura de alta disponibilidad en la nube.

Los últimos años han supuesto un gran auge en la deslocalización de los entornos de producción e incluso de desarrollo. El incremento de las capacidades de cálculo de los procesadores y la reducción de los requisitos en hardware por parte de los Sistemas Operativos, han permitido que las grandes empresas del sector hayan dado un gran salto cualitativo. Han pasado de un nivel de abstracción de la clásica estructura de Datacenter-Servidor a uno de Virtualización en todos los niveles.

Este cambio, en apariencia poco relevante, ha proporcionado a los proveedores la capacidad de dar unos servicios de alta disponibilidad con una facilidad pasmosa, gracias a las tecnologías de virtualización que se han desarrollado en esta última década. Tecnologías como *shadow copy* o los clúster de hypervisores dan al proveedor la capacidad de dar un servicio de alta disponibilidad con un impacto parea el usuario casi inapreciable.

El usuario se ve beneficiado a su vez gracias a la infraestructura del proveedor, que es capaz de ofrecer distintos servicios que, en algunos casos, no se limitan a los del propio proveedor sino a servicios de terceros. Esta gran variedad de servicios ya diseñados y optimizados para las arquitecturas que estas infraestructuras ofrecen, da una riqueza de opciones comparable a un servicio personalizado e individualizado, pero con unos costes a largo plazo infinitamente menores.

1.1. Tipos de Servicios en la Nube.

Todo lo que en el último lustro se ha venido a denominar como "La Nube" es, en realidad, algo que siempre se ha dejado a definiciones vagas pero que en última instancia conduce a tres conceptos: La infraestructura como un servicio, el producto como un servicio y el Software como servicio (siendo este último del que más frecuentemente se trata cuando se habla de "La Nube").

1.1.1. IaaS.

Iniciales de *Infraestructure as a Service*, hacen referencia a un concepto que se trata principalmente en el B2B para ofrecer servicios de infraestructura a casi cualquier nivel a un cliente. Con este servicio se busca deslocalizar, replicar geográficamente, o suplir la falta de infraestructura propia para lanzar un producto.

Entre los proveedores más conocidos de este tipo de servicio se encontrarían Amazon Web Services (AWS) y Microsoft Azure, siendo seguidos por Google Cloud, Century Link, VMWare y tantas otras.

1.1.2. PaaS.

Iniciales de *Plataform as a Service*, hacen referencia al producto intermedio entre el IaaS y el SaaS cuyo ejemplo más clásico sería el servicio de Hosting como en el caso de Wordpress. Los PaaS permiten ejecución de código personal, pero siempre con unas restricciones, ya que no se logra nunca un nivel de control como el que se puede encontrar en un IaaS, sino uno más limitado a configuraciones de PHP, o acceso a bases de datos.

PaaS es un servicio intermedio que no requiere tener conocimientos de servidor y está indicado a aquellos que carecen de la preparación técnica necesaria o del tiempo de implementarlo.

1.1.3. SaaS.

Iniciales de *Software as a Service*, hacen referencia al concepto más conocido del Software que se suministra desde plataformas online. iCloud, OneDrive, Dropbox, Creative Suite de Adobe, todos estos son ejemplos de aplicaciones que se ofrecen a

consumidores de cualquier tipo, tanto empresas como particulares. Estos y otros software siguen un patrón que es lo que les define como SaaS: Ofrecer un Software de forma remota o local de un servicio y que también requiera de forma parcial o total acceso a internet para realizar funciones propias e inherentes del servicio.

1.2. Servicios Mixtos.

Aunque estos tres tipos de servicios puedan parecer disgregados, en la actualidad muchas empresas están dando en estos momentos los tres niveles de servicio bajo el mismo paraguas. Los más conocidos son los referenciados previamente en el apartado de IaaS.

Estos proveedores han desarrollado una infraestructura muy potente y poco a poco la van enriqueciendo con servicios PaaS o SaaS. Dando infinidad de opciones a los desarrolladores para que trabajen con total comodidad para poder elegir cosas tan simples como qué tipo de base de datos usar. Todo con el objetivo de que el hándicap que limite al producto sea el propio equipo de desarrollo y no de la plataforma donde esté desplegado.

1.3. Top del mercado.

Como ya se ha mencionado en el punto sobre IaaS el mercado de la infraestructura está bastante poblado, pero solo dos empresas copan ahora en todos los aspectos los rankings de este sector:

- Amazon Web Services.
- Microsoft Azure.

Ambas empresas son las que cuentan con mayores centros de datos y con mayor número de localizaciones a lo largo del globo. Además de contar con la mayor variedad de servicios en todos los niveles.

Ilustración 1 Gartner: Magic Quadrant for Cloud Infrastructure as a Service, Worldwide report, Lydia Leong et al, publicado el 18 de mayo de 2015

Ambos ofrecen servicios muy parecidos dada la amplia gama que tienen, pero también hay diferencias y en ellas se ve lo mejor de cada servicio.

1.3.1. Diferencias entre AWS y Azure.

Las diferencias de ambos servicios serán divididas en dos, por calidad de servicio y por oferta de servicios.

1.3.1.1. Diferencias a nivel de calidad de servicio.

Se han hecho muchos análisis a lo largo de los años que inclinaban la balanza siempre entre alguno de ellos, incluso en ocasiones a google cloud. Pero en el fondo las opciones de despliegue más variadas son las que al final dan peso en favor de una opción u otra.

La necesidad de tener más opciones a la medida, y con ello optimizar la relación cómputo/gasto es muy importante para los servicios web. Pero aquí ya entraríamos en la siguiente parte.

1.3.1.2. Diferencias en oferta de servicios.

La oferta de servicios es la que puede marcar la diferencia cuando el cliente lo que busca es soluciones a medida. Tanto AWS como Azure cuentan con grandes herramientas con las que ayudar a sus clientes a crear su infraestructura ideal. Desde almacenamiento de datos (S3 y Blobs), máquinas virtuales (EC2 y VMs), elementos de red (route 53, ACS), microservicios (EBS, webapps) o soluciones de terceros (Cisco, Oracle).

Este sector se ve muy enriquecido en AWS ya que ofrecen muchos niveles de granularidad y no solo en tamaño de las máquinas virtuales, sino en el tipo: máquinas completas, con gráficas CUDA, contenedores (docker), almacenamiento de contenido estático,…

Por su parte Azure da además de una granularidad de opciones nada desdeñable, herramientas para poder combinar o sincronizar las infraestructuras locales o *"on promise"* con las de la nube. Permitiendo así soluciones híbridas que en ciertos entornos con requisitos explícitos de seguridad e inventariado, como en el caso de los datos personales de nivel 3, ser más factibles y asumibles para una empresa a nivel de costes.

1.4. Conclusiones.

El alto grado de complejidad en el volumen de opciones ha hecho que de nuevo la decisión de cómo y dónde, desarrollar y desplegar un producto, sea más una decisión donde la mayor parte del peso recae en los departamentos de ingeniería de una empresa.

La gran variedad de opciones en todos los aspectos es una gran noticia y sobretodo que se encuentren no solo dos actores, sino más intentando pujar por la predilección del cliente. Esa competencia en el fondo, es un beneficio para todos.

2. Servicios web.

El despliegue desde su inicio ha sido siempre progresivo. Desde esa pequeña caja negra con una luz roja que conectaba las universidades de California, Stanford y Utah, hasta lo que ahora mismo es internet ha pasado casi medio siglo. En este tiempo internet ha crecido no solo en usuarios, ni tráfico, sino también en servicios. Servicios que a día de hoy nos permiten desde mirar el tiempo hasta poder vigilar la salud de nuestro hijo en el hospital en tiempo real.

Para nuestro caso, nos centraremos en los medios para ofrecer servicios web.

2.1. Definición de servicio Web.

Definir que es un servicio web es algo complicado en medida a todos los diferentes servicios que existe. Desde la W3C han hecho un esfuerzo acuñando la siguiente definición:

> *"(…) conjunto de aplicaciones o de tecnologías con capacidad para interoperar en la Web."*

Los servicios web tienen diferentes posibles implementaciones y tecnologías implicadas.

2.2. Tecnologías implicadas.

Las diferentes tecnologías e implementaciones que pueden verse en servicios web son las siguientes.

2.2.1. HTTP.

HTTP (Hypertext Transfer Protocol) es el protocolo de aplicación (nivel de capa OSI 7) empleado para llevar información y establecer medios de interacción entre los diversos sistemas que dan forma a la World Wide Web. Con un desarrollo tutelado por la Internet Engineering Task Force (IETF) y el World Wide Web Consortium (W3C), HTTP es el protocolo que da soporte a la red como herramienta global de comunicación.

El funcionamiento de la versión actual del protocolo (HTTP/1.1) queda descrito en el RFC 2616 (Junio de 1999).

Actualmente se encuentra en estado de Borrador el estándar HTTP/2, que incluye mejoras presentadas por Google en SPDY o de Microsoft en *Speed+Movility*.

2.2.2. Lenguajes de Servidor

Cuando se va a desarrollar Servicios Web, las opciones a elegir un lenguaje de programación para la parte del servidor son muchas, y aunque, en la teoría cualquier lenguaje podría servir para crear servicios web, existen lenguajes que ofrecen enfoques más prácticos y directos para este tipo de servicios. Los lenguajes empleados para desarrollar Servicios Web se pueden clasificar en dos grande grupos:

- Lenguajes Compilados
- Lenguajes Interpretados

Lenguajes Compilados

Los lenguajes compilados ofrecen mayor seguridad antes ciertos fallos, dado que al pasar por el proceso de compilación se detectan errores de código que en lenguajes interpretados pueden no ser detectados. Estos lenguajes ofrecen además, una mayor velocidad de ejecución al ser el resultado final, instrucciones más cercanas al código máquina.

Algunos ejemplos comunes de lenguajes compilados empleados comúnmente en Servicios Web son:

- C++
- .NET
    - C#
    - Visual Basic
- JAVA

Lenguajes Interpretados.

Los lenguajes interpretados (o de scripting) ofrecen menos velocidad de ejecución, pero curvas de aprendizaje más

accesibles y son más ágiles a la hora de introducir cambios en sistemas que ya están en funcionamiento.

Interpretados vs. Compilados.

Es interesante comentar que los lenguajes más utilizados a la hora de desarrollar Servicios Web suelen ser lenguajes Interpretados, esto se debe a que durante muchos años, la velocidad de ejecución no ha sido el cuello de botella de estos sistemas, quedando eclipsada por el problema de las velocidades de transmisión de peticiones y respuestas a través de la red. Hoy en día el cambio de enfoque hacia la explotación masiva de máquinas virtuales o "*containers*" está cambiando eso.

También juega un papel importante en este campo el hecho de que los sistemas no compilados hacen que resulte más sencillo modificar partes concretas del código de un determinado servicio.

2.2.3. Lenguajes *de Cliente*

En la parte del Cliente, cuando se trata de aplicaciones web utilizadas desde un navegador, se hace un poco más extraño hablar de lenguajes de programación, ya que durante mucho tiempo todo el trabajo de algoritmia y código ha estado concentrado en la parte del servidor, quedando para el usuario la tarea de interpretar y mostrar los datos recibidos usando algún lenguaje de marcado (por ejemplo HTML o XML).

En la actualidad merece una mención especial JavaScript en la parte del cliente ya que este lenguaje ha ido creciendo en relevancia y sofisticación a la hora de dar soporte a la lógica de la aplicación ejecutada en el cliente. En el último lustro JavaScript ha visto un auge en el uso de este lenguaje para código ejecutado en servidor con Node.js con el objetivo de dar soporte a programas de red altamente escalables. Hay que añadir que gracias a JavaScript surgieron tecnologías de gran versatilidad en el lado cliente del front-end como:

- AJAX (Asynchronous JavaScript And XML).
- jQuery.

Cuando se trata de partes Cliente contenidas en dispositivos móviles cada plataforma concreta ofrece su lenguaje nativo de

desarrollo, siendo los más comunes en la actualidad los siguientes:

- Objetive C o Swift (más reciente) en dispositivos iOS.
- JAVA en dispositivos Android.

2.3. Estilos Arquitectónicos para Servicios Web.

Los Servicios Web pueden desarrollarse conforme a distintas arquitecturas generales, en este apartado se detallan los criterios que ayudan a distinguirlas y las arquitecturas más utilizadas en el presente. A la hora de diferenciar estilos arquitectónicos hay dos preguntas que nos marcan las principales diferencias:

1. De qué manera comunica el cliente al servidor sus intenciones, es decir, como distingue el servidor las distintas peticiones (Method Information).
2. De qué manera comunica el cliente al servidor sobre el contexto en el que quiere operar (Scoping Information).

Respondiendo a estas preguntas a continuación se presentan algunas de las arquitecturas más extendidas.

2.3.1. RESTful, Resource Oriented.

En las arquitecturas REST, orientadas a recursos, la información sobre el método solicitado se detalla en el mensaje HTTP y la información sobre los datos sobre los que se quiere operar se detalla en la URI (Uniform Resource Identifier) a la que se realiza la petición. A la hora de determinar la información del método solicitado (Method Information) las arquitecturas REST se basan en los verbos de petición (o acciones) de HTTP.

- POST
- GET
- PUT
- DELETE

Para determinar sobre que parte del sistema se quiere operar (Scoping Information) se utilizan las URI (Uniform Resource Identifier) como medio de identificación de cada uno de los miembros del sistema web sobre los que se puede trabajar.

Para que una arquitectura se considere RESTful debe cumplir estos dos requisitos: el verbo HTTP empleado debe correspondese con la acción solicitada y el recurso ha de identificarse unívocamente con una URI. En caso contrario el sistema no puede ser considerado estrictamente REST.

2.3.2. Remote Procedure Call (RPC).

En las arquitecturas RPC el servidor recibe peticiones que contienen toda la información sobre el método solicitado y sobre el contexto en el que se quiere aplicar (Method y Scoping).

Esta arquitectura no utiliza los verbos HTTP como una herramienta para proveer al servidor de información sobre el método que solicita el cliente. Toda la información, tanto de Método como de Contexto viaja en el cuerpo de la petición HTTP, en este caso HTTP actúa únicamente como un protocolo de transporte.

2.3.3. Arquitecturas hibridas.

Se consideran arquitecturas hibridas aquellas que no responden de manera estricta a todos los requisitos de ninguna de las dos arquitecturas previamente descritas.

Un ejemplo de esto son las arquitecturas que utilizan URIs que identifican recursos de modo univoco pero que también contienen información sobre la acción solicitada; también los sistemas que utilizan los verbos HTTP para identificar la acción solicitada pero que encapsulan dentro del cuerpo de la petición HTTP la identificación del recurso solicitado.

III. Desarrollo.

1. Planteamiento del problema.

El cometido de este proyecto en el desarrollo global de las tres partes que conforman este generador de Redes Sociales Geoposicionadas, es desarrollar una web para administración y desplegar en una plataforma que sea capaz de dar un servicio eficaz y escalable a la aplicación en todas sus capas, tanto a nivel front-end como back-end.

Una de las premisas heredadas del desarrollo del back-end es que es necesaria una máquina que corra PHP y symfony junto a una base de datos MYSQL.

A continuación se recopilarán los casos de uso y requisitos asociados a la web de administración de la red social y tras ello entraremos en las cuestiones de diseño de la arquitectura.

2. Casos de Uso. Administrador – Servidor.

2.1. CU-01: Administrador crea Usuario.

Escenario principal de éxito:

1. El Administrador accede al sistema a través del Panel de Administración.
2. El administrador selecciona la opción “Crear Usuario”.
3. Selecciona el tipo de Usuario a crear (Administrador o Cliente).
4. Introduce los datos necesarios para crear el Usuario.
5. El Panel de Administración envía al Servidor la petición.
6. El Servidor confirma la validez de los datos.
7. Se recibe una notificación del éxito de la operación.

Extensiones:

6a. El Servidor rechaza los datos recibidos.

a) Fin del proceso.

6b. Ya existe un Usuario coincidente con el que se desea crear (Conflicto).

a) Fin del proceso.

2.2. CU-02: Administrador Modifica el perfil de un Usuario.

Escenario principal de éxito:

1. El Administrador accede al sistema a través del Panel de Administración.
2. El administrador selecciona la opción "Modificar Usuario".
3. Selecciona el perfil que quiere modificar.
4. Introduce los datos que desea modificar.
5. El Panel de Administración envía al Servidor la petición.
6. El Servidor confirma la validez de los datos.
7. Se recibe una notificación del éxito de la operación.

Extensiones:

6a. El Servidor rechaza los datos recibidos.

a) Fin del proceso.

2.3. CU-03: Administrador Elimina un Usuario.

Escenario principal de éxito:

1. El Administrador accede al sistema a través del Panel de Administración.
2. El administrador selecciona la opción "Eliminar Usuario".
3. Selecciona el perfil que quiere eliminar.
4. El Panel de Administración envía al Servidor la petición.

5. El Servidor confirma la validez de los datos.
6. Se recibe una notificación del éxito de la operación.

Extensiones:

5a. El Servidor rechaza los datos recibidos.

a) Fin del proceso.

2.4. CU-04: Administrador Inspecciona perfil de Usuario.

Escenario principal de éxito:

1. El Administrador accede al sistema a través del Panel de Administración.
2. El administrador selecciona la opción "Inspeccionar Usuario".
3. Selecciona el perfil que quiere inspeccionar.
4. El Panel de Administración envía al Servidor la petición.
5. El Servidor confirma la validez de los datos.
6. Se reciben los datos del Usuario solicitado.

Extensiones:

5a. El Servidor rechaza los datos recibidos.

a) Fin del proceso.

2.5. CU-05: Administrador Crea Mensaje.

Escenario principal de éxito:

1. El Administrador accede al sistema a través del Panel de Administración.
2. El administrador selecciona la opción "Crear Mensaje".
3. Introduce los datos del Mensaje.
4. Introduce la ubicación del Mensaje o fija Área del Mensaje.
5. El Panel de Administración envía al Servidor la petición.
6. El Servidor confirma la validez de los datos.
7. Se recibe una notificación del éxito de la operación.

Extensiones:

6a. El Servidor rechaza los datos recibidos.

a) Fin del proceso.

6b. La ubicación introducida no coincide con ningún Área.

a) Resultado vacío.
b) Fin del proceso.

2.6. CU-06: Administrador Modifica Mensaje.

Escenario principal de éxito:

1. El Administrador accede al sistema a través del Panel de Administración.
2. El administrador selecciona la opción "Modificar Mensaje".
3. Selecciona el Mensaje que quiere modificar.
4. Introduce los datos que desea modificar.
5. El Panel de Administración envía al Servidor la petición.
6. El Servidor confirma la validez de los datos.
7. Se recibe una notificación del éxito de la operación.

Extensiones:

6a. El Servidor rechaza los datos recibidos.

a) Fin del proceso.

2.7. CU-07: Administrador Elimina Mensaje.

Escenario principal de éxito:

1. El Administrador accede al sistema a través del Panel de Administración.
2. El administrador selecciona la opción "Eliminar Mensaje".
3. Selecciona el Mensaje que quiere eliminar.
4. El Panel de Administración envía al Servidor la petición.

5. El Servidor confirma la validez de los datos.
6. Se recibe una notificación del éxito de la operación.

Extensiones:

5a. El Servidor rechaza los datos recibidos.

b) Fin del proceso.
c)

2.8. CU-08: Administrador Inspecciona Mensaje.

Escenario principal de éxito:

1. El Administrador accede al sistema a través del Panel de Administración.
2. El administrador selecciona la opción "Inspeccionar Mensaje".
3. Selecciona el Mensaje que quiere inspeccionar.
4. El Panel de Administración envía al Servidor la petición.
5. El Servidor confirma la validez de los datos.
6. Se reciben los datos del Mensaje solicitado.

Extensiones:

5a. El Servidor rechaza los datos recibidos.

a) Fin del proceso.

2.9. CU-09: Administrador Actúa sobre un Mensaje.

Escenario principal de éxito:

1. El Administrador accede al sistema a través del Panel de Administración.
2. El administrador selecciona la opción "Actuar sobre Mensajes".
3. Selecciona el Mensaje sobre el que se quiere "Actuar".
4. Selecciona la acción concreta que quiere realizar.
5. El Panel de Administración solicita al Administrador los datos que sean necesarios para realizar la Acción seleccionada.

6. El Panel de Administración envía al Servidor la petición.
7. El Servidor confirma la validez de los datos.
8. Se recibe una notificación del éxito de la operación.
9. El Panel de Administración muestra el nuevo estado del Mensaje tras la Acción.

Extensiones:

6a. El Servidor rechaza los datos recibidos.

a) Fin del proceso.

6b. La Acción solicitada devuelve un error

a) Fin del proceso.

2.10. CU-10: Administrador Crea Área.

Escenario principal de éxito:

1. El Administrador accede al sistema a través del Panel de Administración.
2. El administrador selecciona la opción “Crear Área”.
3. Introduce los datos del Área.
4. Introduce la ubicación del Área.
5. El Panel de Administración envía al Servidor la petición.
6. El Servidor confirma la validez de los datos.
7. Se recibe una notificación del éxito de la operación.

Extensiones:

6a. El Servidor rechaza los datos recibidos.

a) Fin del proceso.

2.11. CU-11: Administrador Modifica Área.

Escenario principal de éxito:

1. El Administrador accede al sistema a través del Panel de Administración.
2. El administrador selecciona la opción “Modificar Área”.
3. Selecciona el Área que quiere Modificar.
4. Introduce los datos que desea modificar.
5. El Panel de Administración envía al Servidor la petición.
6. El Servidor confirma la validez de los datos.
7. Se recibe una notificación del éxito de la operación.

Extensiones:

6a. El Servidor rechaza los datos recibidos.

a) Fin del proceso.

2.12. CU-12: Administrador Elimina Área.

Escenario principal de éxito:

1. El Administrador accede al sistema a través del Panel de Administración.
2. El administrador selecciona la opción “Eliminar Área”.
3. Selecciona el Área que quiere eliminar.
4. El Panel de Administración envía al Servidor la petición.
5. El Servidor confirma la validez de los datos.
6. Se recibe una notificación del éxito de la operación.

Extensiones:

5a. El Servidor rechaza los datos recibidos.

a) Fin del proceso.

2.13. CU-13: Administrador Inspeccionar Área.

Escenario principal de éxito:

1. El Administrador accede al sistema a través del Panel de Administración.
2. El administrador selecciona la opción “Inspeccionar Área”.
3. Selecciona el Área que quiere inspeccionar.
4. El Panel de Administración envía al Servidor la petición.
5. El Servidor confirma la validez de los datos.
6. Se reciben los datos del Área solicitada.

Extensiones:

5a. El Servidor rechaza los datos recibidos.

a) Fin del proceso.

3. Requisitos.

En este apartado se enumeran los requisitos identificados para la web de administración de la red social.

El conjunto de requisitos describe las características y atributos de la web del proyecto, omitiendo la minuta de la implementación.

ID: RF-01	TIPO: Funcional	Fuente: CU-01
Título: Crear Cuentas		
El Administrador tiene que poder crear Cuentas de Usuario, tanto para el rol de Cliente como para el rol de administrador, usando la aplicación de administración del sistema.		

ID: RF-02	TIPO: Funcional	Fuente: CU-02
Título: Modificar Cuentas		
El Administrador tiene que poder modificar los datos de las cuentas, tanto para el rol de Cliente como para el rol de administrador, usando la aplicación de administración del sistema.		

ID: RF-03	TIPO: Funcional	Fuente: CU-03
Título: Eliminar Cuentas		
El Administrador tiene que poder eliminar del sistema Cuentas de Usuario, tanto para el rol de Cliente como para el rol de administrador, usando la aplicación de administración del sistema.		

ID: RF-04	TIPO: Funcional	Fuente: CU-04
Título: Inspeccionar Cuentas		
El Administrador tiene que poder acceder a la información de las Cuentas de Usuario usando la aplicación de administración del sistema.		

ID: RF-05	TIPO: Funcional	Fuente: CU-10
Título: Crear Áreas		
El Administrador tiene que poder Crear Áreas nuevas en el Sistema usando la aplicación de administración del sistema.		

ID: RF-06	TIPO: Funcional	Fuente: CU-11
Título: Modificar Áreas		
El Administrador tiene que poder Modificar los datos de las Áreas existentes en el Sistema usando la aplicación de administración del sistema.		

ID: RF-07	TIPO: Funcional	Fuente: CU-12
Título: Eliminar Áreas		
El Administrador tiene que poder Eliminar las Áreas existentes en el Sistema usando la aplicación de administración del sistema.		

ID: RF-08	TIPO: Funcional	Fuente: CU-13
Título: Inspeccionar Áreas		
El Administrador tiene que poder Acceder a la información de las Áreas existentes en el Sistema usando la aplicación de administración del sistema.		

ID: RF-09	TIPO: Funcional	Fuente: CU-05
Título: Crear Mensaje		
El Administrador tiene que poder Crear Mensajes Pertenecientes a Áreas previamente definidas en el Sistema usando la aplicación de administración del sistema.		

ID: RF-10	TIPO: Funcional	Fuente: CU-06
Título: Modificar Mensaje		
El Administrador tiene que poder Modificar los datos de los Mensajes existentes en el Sistema usando la aplicación de administración del sistema.		

ID: RF-11	TIPO: Funcional	Fuente: CU-07
Título: Eliminar Mensaje		
El Administrador tiene que poder Eliminar los Mensajes existentes en el Sistema usando la aplicación de administración del sistema.		

ID: RF-12	TIPO: Funcional	Fuente: CU-08
Título: Inspeccionar Mensaje		
El Administrador tiene que poder Acceder a la información de los Mensajes existentes en el Sistema usando la aplicación de administración del sistema.		

ID: RF-13	TIPO: Funcional	Fuente: CU-09
Título: Actuar sobre Mensajes		
El Administrador tiene que poder realizar acciones (definidas por los desarrolladores del Sistema), sobre los Mensajes que lo necesiten, usando la aplicación de administración del sistema.		

ID: RF-14	TIPO: Funcional	Fuente:
Título: Autenticación en el Sistema (Inicio de sesión Administrador)		
El Administrador debe disponer de medios para identificarse en el sistema para, posteriormente, realizar las acciones que requieran de identificación. Esta identificación debe realizarse se manera segura y persistente que permita mantener una sesión de trabajo.		

ID: RF-15	TIPO: Funcional	Fuente:
Título: Cierre de Sesión en el Sistema		
El Administrador debe poder terminar una sesión en el Sistema cuando lo desee utilizando la aplicación de administración.		

ID: RF-16	TIPO: Funcional	Fuente:
Título: Sesiones Múltiples		
Se debe permitir a los Administradores mantener activas varias Sesiones en distintos dispositivos a la vez.		

4. Esquema básico de la máquina.

En un proceso iterativo, con el fin de optimizar recursos y encapsular componentes, se opta por seguir un proceso inductivo para la gesta de la arquitectura final. Para ello se comienza desde la premisa de la arquitectura más básica; una sola máquina.

Se realiza un despliegue completo con el objetivo inicial de ver que todo funciona correctamente y sin problemas. Y después se procede a adaptar y optimizar componentes para poder en tomar a posteriori decisiones que permitan una mayor maniobrabilidad en la infraestructura.

4.1. Encapsulación del sistema.

Tras lo visto en el despliegue del esquema básico, se pueden observar tres zonas muy obvias que, tras un poco de ajuste en configuración y ficheros, permiten la segmentación en front-end de páginas puramente html5 y JavaScript, el back-end desarrollado en symfony y para terminar la base de datos MYSQL. Estas tres partes pueden separarse en varios contenedores que pueden ofrecer unas cualidades de escalabilidad muy altas.

Para poder hacer esto, solo se requieren dos tareas. La primera sería un cambio del *connection string* a una base de datos exterior a la propia máquina que tiene symfony. La segunda tarea sería un cambio de las URL en el front-end que llamen a la API.

Tras esto, se puede dejar el sistema de la siguiente forma:

- Servidores front-end con la web de administración.
- Servidores back-end con el núcleo de la red social en symfony.
- Servidores MYSQL con todos los datos que maneje el back-end.

4.2. Requisitos de cada parte.

Una vez delimitadas las necesidades de cada componente, se procede a especificar los requisitos de cada uno.

Servidor front-end:

Cualquier servidor básico (NGINX, APACHE, IIS, Tomcat,…) es suficiente para dar servicio a la web de administración, ya que solo usa HTML5 y JavaScript. No se usan tecnologías de lado del servidor como PHP, ASP o JSP.

- Servidor back-end:

  El requisito de software PHP y symfony es algo obligatorio en esta parte del servicio, debido a la imposibilidad de modificar el back-end, con lo que las opciones más comunes para utilizar serían NGINX o APACHE.

- Servidor de BBDD:

  El desarrollo del back-end no modificable no deja más opción que usar MYSQL como motor de base de datos.

4.3. Modificaciones necesarias.

Ya se han descrito brevemente anteriormente los cambios a realizar y en este punto se hará un análisis más exhaustivo de los cambios realizados.

- Cambios en la configuración:

  Gracias a que en el diseño inicial del back-end se contempló el despliegue en entornos de alta disponibilidad los cambios en configuración se limitan únicamente al *conection string*.

- Cambios en el front-end:

  Los cambios en el front-end han tenido varias fases. La primera de ellas fue el reemplazo de todo el código PHP por funciones JavaScript, AJAX o jQuery. Esta primera fase pretendía desligarse al completo de las necesidades de un servidor web concreto y abrazar las bondades de las herramientas ya citadas que trabajan en los dispositivos del cliente final.

La segunda fase ha consistido en modificar las URLs, convirtiéndolas en un elemento configurable desde un solo fichero, con el objetivo de que en caso de cambio del dominio donde se alberga el back-end, sea un cambio único y sencillo.

5. Alternativas de diseño.

En este punto se presentan a priori dos arquitecturas de red distintas para el proyecto.

La primera sería aquella que proteja el back-end tras el front-end y no permita acceso más que a través de este.

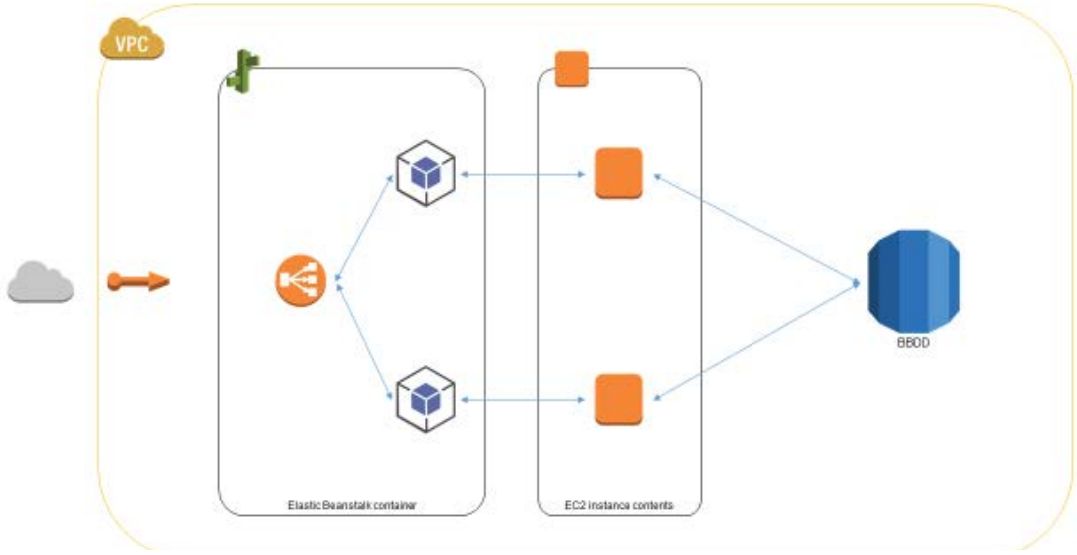


Ilustración 2 Esquema 2-2-1

Y la segunda sería una que exponga la API del back-end para que pueda usarse siguiendo los requisitos de autorización que esta misma aporta.

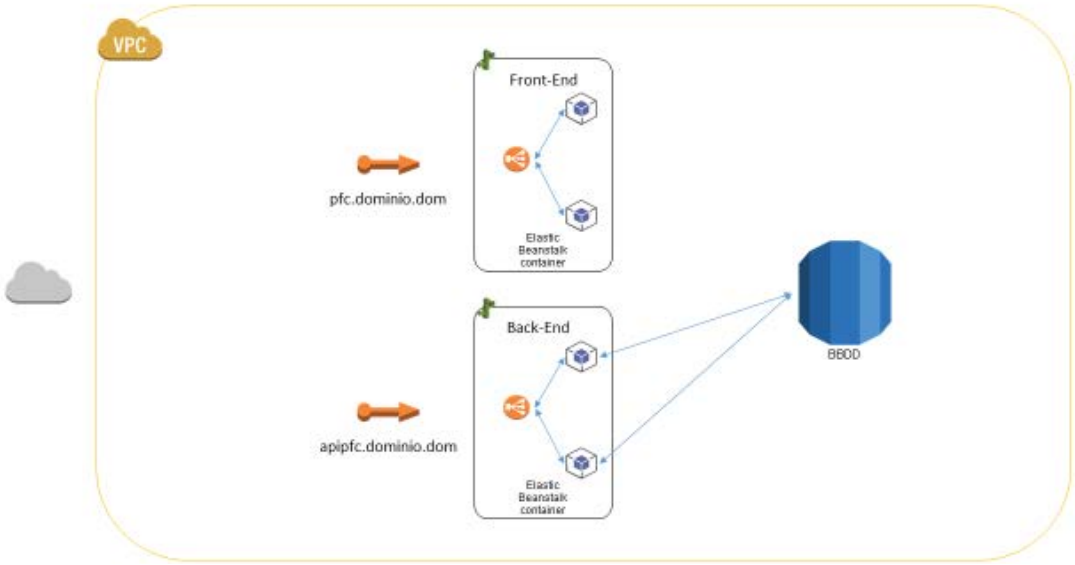


Ilustración 3Esquema 2; 2-1

Ventajas del primer caso.

- Protección de acceso a la API.
- Único balanceo de carga en el acceso (abarata costes).
- Solo requiere un registro DNS.

Ventajas del segundo caso.

- Autoescalado independiente entre front y back-end.
- Balanceo independiente a cada nivel.
- Permite dar acceso público a la API.

6. Elección de la infraestructura final.

Tras todos lo analizado previamente, la arquitectura escogida es el segundo modelo. Una arquitectura autoescalable con front-end y back-end en primera línea y BBDD detrás del back-end. La ausencia de una estructura avanzada para la base de datos es producida por falta de tiempo para estudiar cuál de las posibles opciones (espejo, arbitro,…) es la más adecuada para este proyecto.

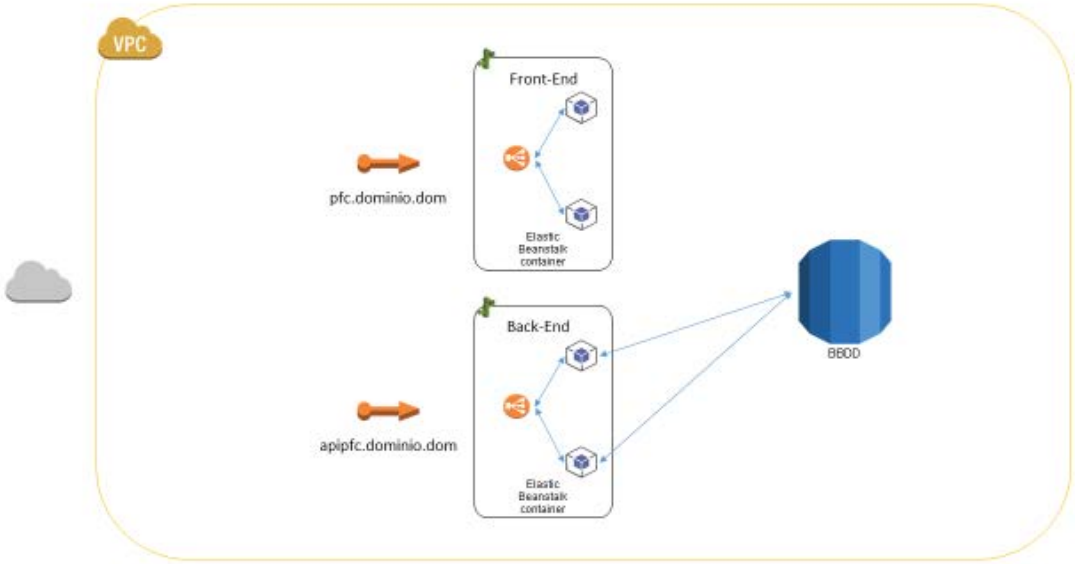


Ilustración 4Esquema escogido

IV. Planificación y Presupuesto.

1. Consideraciones previas.

Este presupuesto hace referencia únicamente a los costes humanos del proceso de adaptación a la nueva infraestructura y los costes de servicio generados por la plataforma de nube donde se aloje la red social.

De los siguientes datos podría hacerse una idea del coste que puede suponer a una empresa sin infraestructura la adopción de este proyecto excluyendo la licencia del servidor de la red social y los costes de desarrollo de los clientes para móviles.

2. Objetivos para la ejecución del proyecto.

En este apartado se establecen las necesidades a cumplir para dar por funcional un servidor de este proyecto.

El proyecto incluye, el despliegue del back-end en un entorno de alta disponibilidad y la adaptación del front-end de administración a las necesidades del cliente destinatario del servicio.

Análisis de las necesidades del cliente.

El equipo de desarrollo evaluará las necesidades del cliente con respecto al producto y tomará datos para definir los casos de uso que se acuerden.

Diseño de la solución a medida.

Tras la toma de requisitos y casos de uso se diseñará una solución a medida para el cliente.

Implementación del servidor.

El equipo implementará el front-end fruto de los requisitos del cliente y en el entorno de alta disponibilidad.

Creación del manual para la implementación concreta.

Como último objetivo, se establece la necesidad de tener una documentación que sirva para en el futuro poder hacer ajustes en la implementación. Tanto a modelos más exigentes, como a más modestos.

3. Planificación de tareas.

Para lleva a cabo las tareas del proyecto, se plantea la siguiente lista de tareas y restricciones junto a la asignación de las mismas.

3.1. Equipo de trabajo.

Para llevar a cabo este proyecto se formará un equipo de cinco personas con distintos tipos de experiencia.

Jefe de Proyecto.

Encargado por velar por el correcto progreso del proyecto en todas las áreas y máximo responsable del mismo.

Llevará todas las tareas de gestión del equipo compartiendo solo algunas responsabilidades con el Analista de Software y Sistemas.

Analista de Software y Sistemas.

Encargado de ofrecer un nivel intermedio de experiencia entre los programadores web y el jefe de proyecto. Su labor es dar cohesión a las ideas recibidas de la toma de requisitos y asesorar a los programadores en caso de algún tipo de duda con la tecnología o las expectativas del cliente.

El analista debería estar versado en cómo funciona el núcleo de la red social.

Programador web.

El perfil de programador que se necesita en este proyecto es aquel orientado a tecnologías front-end. Se pide que sepa de HTML5, CSS3 y JavaScript. Opcionalmente sería conveniente que supiera de jQuery y Bootstrap (u otros frameworks en JavaScript de *Responsive desing*).

De este perfil harán falta dos personas.

Maquetador.

En este proyecto este perfil está interpretado de una manera particular, ya que se tratará de un rol más vertical que todos los anteriores. Esta persona necesitará unos conocimientos como los de sus compañeros programadores en temas de HTML5, CSS3 y Bootstrap, aunque pueda pecar un poco más en JavaScript. Pero además se le requerirá unos conocimientos en tema de maquetación usabilidad y teoría de colores.

Este perfil como decía previamente, tiene un rol más vertical y es que dada su responsabilidad en el aspecto final del producto de cara al cliente, el maquetador deberá asistir a todas las reuniones con este.

3.2. Planificación.

Para llevar a cabo este proyecto, tomaremos el caso de un usuario que no solicite incorporar funcionalidades al núcleo de la aplicación.

El progreso de desarrollo seguirá un esquema básico de toma de requisitos, diseño en función de estos y revisión para continuar hacia adelante. Con la intención de no hacer este documento eterno, estos ciclos solo se representarán una vez, sobreentendiéndose que tras una revisión se retorna a una toma de requisitos o rediseño si en la revisión se encuentran discrepancias dentro del alcance del proyecto.

3.2.1. Toma de requisitos del cliente.

Esta tarea es muy importante. Una buena toma de requisitos en el inicio junto a un mutuo acuerdo de cuál es el alcance del proyecto son la mejor base para el posterior trabajo.

En esta tarea intervendrán: El jefe de proyecto, el analista de software y sistemas y el maquetador.

3.2.2. Despliegue en desarrollo de la API REST.

Tarea paralela e independiente de cualquier otra, siendo necesaria para poder empezar el paso cinco. El Analista que es el único versado en la plataforma realizará el despliegue del núcleo de la red social en un entorno de desarrollo.

3.2.3. Diseño del front-end.

Esta reunión entre el analista, el maquetador y los programadores dará a luz lo que será el diseño del front-end para el cliente.

3.2.4. Revisión del diseño con el cliente.

Tras la primera iteración del diseño se establece una reunión con la cual dar comienzo al proceso de gesta del producto. En esta reunión el analista y el maquetador intentarán ver si el cliente está satisfecho y el jefe de producto cerrar el alcance de todo lo que está por venir.

3.2.5. Desarrollo del front-end.

Esta tarea engloba a las cuatro próximas tareas y que conforman en conjunto el grueso del desarrollo web.

3.2.6. Creación de la pantalla de Login.

Creación de la pantalla de login que precede a todo el resto de contenido. Con el fin de aumentar la seguridad se intentará no usar cookies para esta tarea y usar LocalStorage o SessionStorage.

3.2.7. Creación de la web "main".

Pantalla que sucede al login y que sirve de marco para el resto de páginas que se cargaran con un load de jQuery.

3.2.8. Creación de subpáginas.

El resto de páginas que realicen acciones serán desarrolladas teniendo en cuenta que serán un div dentro del "*main*".

3.2.9. Maquetación de la web.

Al mismo tiempo que se van desarrollando las distintas páginas que se van a crear, el maquetador irá dando vistosidad y la usabilidad que el cliente desea.

3.2.10. Implementación y despliegue 'on promise'.

Una vez realizado el desarrollo se pondrá en una máquina de desarrollo local con el objetivo de hacer tareas de testeo y depuración con el núcleo de la red social. En los siguientes tres puntos se detalla un poco.

3.2.11. Integración del front-end de desarrollo.

Integración de todos los elementos desarrollados en el paso 3.2.3. además de con el back-end desplegado en 3.2.2.

3.2.12. Testeo interno.

Testeo de la solución completa.

3.2.13. Depuración y corrección de errores.

Arreglo de todos los fallos encontrados en la fase anterior.

3.2.14. Presentación con cliente.

Una vez completado el proceso y en un estado óptimo de funcionamiento, queda hacer una exposición del producto al cliente antes de entrar en la recta final.

3.2.15. Implementación y despliegue en la nube.

Con el visto bueno del cliente, se entra en la recta final que la conforman las siguientes tres tareas.

3.2.16. Despliegue de primeras versiones en la nube.

Subir a la arquitectura final en los formatos elegidos los elementos que se hayan aprobado con el cliente.

3.2.17. Testeo de infraestructura.

Pruebas de estrés y funcionalidades al completo, se ha de asegurar que con el cambio de infraestructura no se ha perdido ninguna.

3.2.18. Redimensionamiento de la infraestructura mínima.

Con las pruebas de estrés se puede ir viendo los límites de la infraestructura y como el escalado automático ayuda a no llegar a ellos. Este paso es crucial muy importante para el futuro del proyecto y aunque es algo constante a analizar una buena base ayudará a tener un mejor comienzo.

3.2.19. Cierre de documentación.

Con el producto completo cerrado, puede procederse a cerrar la documentación con todos los detalles necesarios. Incluidos los datos extraídos de las pruebas de estrés que podrán compararse en el futuro si la red social crece lo suficiente.

3.2.20. Entrega a Cliente.

Entrega de toda la documentación necesaria, y cierre del contrato de desarrollo, comenzando el periodo de soporte.

3.2.21. Periodo de soporte.

Durante el primer mes en un principio se dará soporte pudiendo modificarse la duración del mismo si el cliente lo solicita.

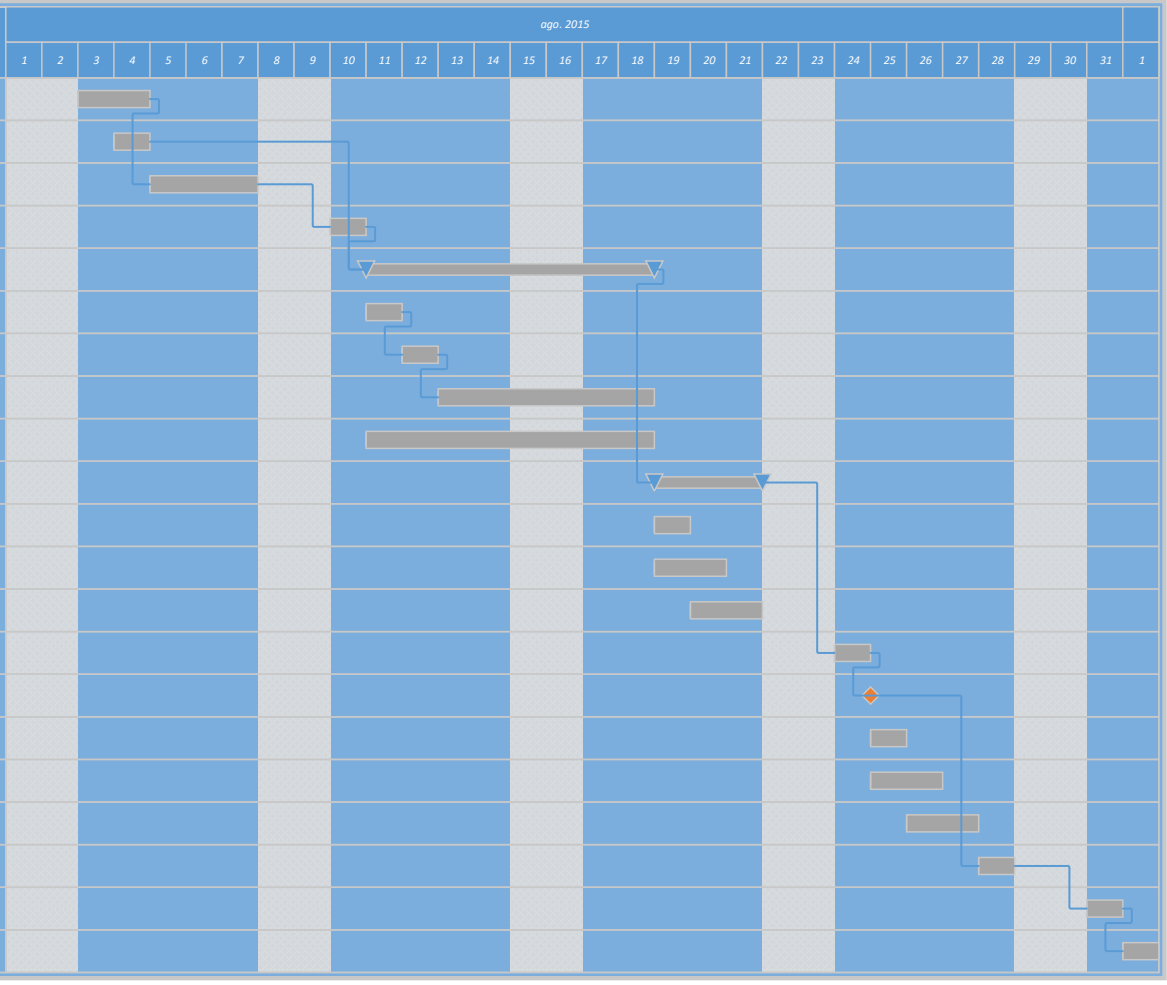
3.3. Aclaración de asignaciones.

A partir del punto 3.2.10. dejan de especificarse a quien se asigna la tarea, ya que en el fondo requieren del compromiso de todo el mundo en el caso de un proyecto tan pequeño como este.

3.4. Gantt.

Tabla 1 Diagrama Gantt del Proyecto

Id.	Nombre de tarea	Comienzo	Fin	Duración
1	Toma de requisitos del cliente	03/08/2015	04/08/2015	2d
2	Despliegue en desarrollo de la API REST	04/08/2015	04/08/2015	1d
3	Diseño del front-end	05/08/2015	07/08/2015	3d
4	Revisión del diseño con el cliente	10/08/2015	10/08/2015	1d
5	**Desarrollo del front-end**	**11/08/2015**	**18/08/2015**	**6d**
6	Creación de la pantalla de login	11/08/2015	11/08/2015	1d
7	Creación de la web "main"	12/08/2015	12/08/2015	1d
8	Creación de subpáginas	13/08/2015	18/08/2015	4d
9	Maquetación del la web	11/08/2015	18/08/2015	6d
10	**Implementación y despliegue on promise**	**19/08/2015**	**21/08/2015**	**3d**
11	Integración del front-end de desarrollo	19/08/2015	19/08/2015	1d
12	Testing interno	19/08/2015	20/08/2015	2d
13	Depuración y corrección de errores	20/08/2015	21/08/2015	2d
14	Presentación con cliente	24/08/2015	24/08/2015	1d
15	**Implementación y Despliegue en la nube**	**25/08/2015**	**27/08/2015**	**3d**
16	Despliegue de primeras versiones en la nube	25/08/2015	25/08/2015	1d
17	Testing de infraestructura	25/08/2015	26/08/2015	2d
18	Redimensionamiento de la infraestructura mínima	26/08/2015	27/08/2015	2d
19	Cierre de documentación	28/08/2015	28/08/2015	1d
20	Entrega a cliente	31/08/2015	31/08/2015	1d
21	Periodo de soporte	01/09/2015	01/10/2015	23d



4. Costes directos.

Los costes directos incluyen varios conceptos durante el periodo de desarrollo:

- Mano de obra del Jefe de Proyecto.
- Mano de obra del Analista de Software y Sistemas.
- Mano de obra de Programadores.
- Mano de obra del Maquetador.
- Coste de infraestructura de entornos de desarrollo.

4.1. Determinación de costes por hora.

En función de los precios encontrados en el mercado actual puede proponerse la siguiente lista.

- Jefe de Proyecto: 49 euros/hora.
- Analista de Software y Sistemas: 37 euros/hora.
- Programador: 32 euros/hora.
- Maquetador: 35 euros/hora.

4.2. Estimación de jornadas de trabajo.

Tomando la base de: 8 horas equivalen a una jornada laboral; se detallan a continuación el cómputo final de jornadas necesarias para cerrar este proyecto.

- Jornadas del Jefe de Proyecto:
  - 22
- Jornadas del Analista de Software y Sistemas:
  - 22
- Jornadas del Programador:
  - 44

Jornadas del Maquetador:

  - 22

4.3. Amortización del equipo empleado.

Como parte de los costes del proyecto, se incluyen los gastos de amortización del equipamiento.

Con el fin de calcular la depreciación del equipo empleado, se utiliza una función de estimación de la vida media de los equipos en un ambiente de desarrollo de software.

4.4. Vida media y coste de los equipos.

- Ordenador portátil:
  - Vida media: 36 meses.
  - Coste estimado: 600 €
  - Depreciación por jornada: 2,08 €

Con estos datos y habiéndose estimado la duración del proyecto en 22 jornadas, los costes de amortización del equipamiento (5 portátiles) de los empleados, se estima en 228,80 €.

5. Presupuesto final.

Se describe a continuación el presupuesto en conjunto.

5.1. Desglose de costes de mano de obra.

Tabla 2 Precios pormenorizados

PUESTO	JORNADAS	€/HORA	€/HORA (IVA)	COSTE	COSTE (IVA)
JEFE DE PROYECTO	22	49	59.29	1078	1304.38
ANALISTA DE SOFTWARE Y SISTEMAS	22	37	44.77	814	984.94
PROGRAMADORES	44	32	38.72	1408	1703.68
MAQUETADOR	22	35	42.35	770	931.7

5.2. Presupuesto global.

Tabla 3 Costes Totales

Concepto	**Coste**	**Coste con % IVA**
Costes por Trabajo		
Jefe de Proyecto	1078 €	1304.38 €
Analista	814 €	984.94 €
Programadores	1408 €	1703.68 €
Maquetador	770 €	931.7 €
Costes por Material		
	228.8 €	276.848 €
Costes totales		
	4298.8 €	5201.548 €

V. Conclusiones y trabajos futuros.

1. Conclusiones.

El trabajo realizado en este proyecto da los pasos suficientes para poder dar el soporte a una aplicación de gran capacidad, y las pistas necesarias para saber cómo continuar con ella sin verse muy perdido.

Tanto este proyecto como el que realizó Javier Iglesia Sabio han ido acompañadas de una demo funcional que mostrase lo real y asequible que puede ser este proyecto.

A nivel de implementaciones en la nube, por mi parte habría que recalcar que podrían existir implementaciones más eficientes en coste y operatividad que serían de gran interés para futuros desarrollos. Estas mejoras se expresan en el siguiente apartado.

Haciendo un análisis de los objetivos marcados en el principio del documento, llegamos a las siguientes conclusiones personales:

Estudio y aprendizaje de Arquitecturas de red:

El estudio llevado para comprender mejor, distintas formas de desplegar máquinas y que estas puedan mantener una conversación estable. Incluso en circunstancias de apagado y arranque de instancias de forma no controlada por el servicio web. Ha sido cumplido en buenas condiciones para un sistema joven y con una expectativa de uso y crecimiento moderado.

Estudio y aprendizaje de lenguajes de programación y diseño web:

En este campo hay que decir que aunque se han visto las bases y se ha hecho una aplicación de estas de una forma muy directa, queda mucho terreno que explorar si se quiere trabajar en esta materia. A nivel del proyecto, se

han conseguido encontrar las herramientas necesarias que dan soporte al front-end del servicio web.

Estudio y aprendizaje de comunicación con APIs REST y RESTFULL.

El estudio de cómo funciona la tecnología de comunicación y como aplicarla ha resultado muy fructífero, ya que gracias al conocimiento de la misma se ha podido llegar a un planteamiento en la arquitectura que reduce los costes del sistema de una forma óptima.

Estudio y comprensión de servicios de virtualización.

Junto al conocimiento de cómo funciona a nivel de red una API REST, el conocimiento de las distintas opciones de servicio en IaaS y PaaS, nos ha permitido cumplir este objetivo con creces. Tanto sabiendo que escoger y que no escoger y cuando hacerlo y cuando no. La elección entre instancias solo web o máquinas virtuales para el front-end es un ejemplo de ello.

2. Trabajos futuros.

Con el fin de establecer unos puntos de partida en caso de retomar del proyecto, a continuación se anotarán mejoras que podrían realizarse al sistema.

Certificado SSL para encriptación de las comunicaciones.

La instalación de un certificado firmado por una agencia de verificación de identidad en los servidores web otorgaría una vía de encriptación para los mensajes a través del navegador transparente para el usuario. Lo cual añadiría un factor de confianza para el usuario de la red social que acceda por navegador.

Ampliación de datos estadísticos.

Aunque en el panel de administración puedan observarse datos estadísticos de la red social, son muy superficiales y ampliamente mejorables. Gracias a la librería d3.js los datos serán fácilmente representables vía web en el panel de administración, o incluso si se estima oportuno, para el usuario.

Implantación de Clúster de Bases de Datos.

La estructuración de toda la capa relacional de los datos de la red social es algo muy importante para la escalabilidad del sistema y la alta disponibilidad de la misma. El objetivo sería crear un clúster con balanceo de carga que permita a múltiples fuentes preguntar a cualquier nodo, sin que se vea reducida la velocidad de respuesta del sistema. Dando también a opción a poder incorporar nuevos nodos que se ajusten al aumento de la demanda del sistema.

Balanceo de carga a nivel de red en base de datos.

El balanceo de carga es algo ya implementado en las capas del front-end y back-end para los entornos de servidor web. Por lo tanto tan solo sería necesario establecer un servicio de balanceo de carga entre los back-end y las bases de datos (a poder ser, ya implementadas en un clúster no centralizado).

Estudio y diseño de reglas de re-escalado automático de la arquitectura.

La infraestructura del proveedor de servicio en la nube permite establecer reglas bajo las cuales ampliar o reducir los recursos dedicados en varias áreas. En nuestro caso los recursos modificables serían réplicas de los tres tipos de máquinas (Front-end, back-end y BBDD). Siendo las máquinas de BBDD las más sensibles a la replicación, ya que no pueden ser accesibles hasta que todos los datos de las tablas se hayan actualizado.

Front-end estático servido desde almacenamiento estilo S3 de AWS.

El contenido estático de la web puede servirse directamente desde contenedores de ficheros, cosa muy interesante para el modelo de desarrollo del front-end de administración realizado en este proyecto.

Estudio de tecnologías anti denegación del servicio.

Los últimos años están llenos de noticias sobre casos de servicios tirados durante días por culpa de ataques de gran envergadura realizados desde ordenadores zombis desde cualquier punto del globo. El estudio de las opciones que el proveedor de la nube ofrece o la de cualquier tercero, sería algo imprescindible en pasos futuros.

Estudio y testeo de la red social bajo un entorno de redundancia geográfica.

Mantener servidores en todos los continentes y dar el mismo servicio de forma transparente para todos los usuarios, estén desde donde estén y sin sufrir la latencia de las comunicaciones transcontinentales es algo que quedaba fuera de los objetivos de este proyecto y sería muy útil para posibles clientes.

Búsqueda de un servicio de análisis de tráfico web “cookieless”

El proveedor casi sin rival de estadísticas de navegación web es *google analytics*, pero tiene la pega de usar cookies. Aunque es una pega nimia, sería considerable el eliminar cualquier tipo de

cookie del servicio web y evitar con ello muchos de los fallos de seguridad asociados a ellas.

Ampliar el log y los datos anónimos de uso de la red social para explotación con *Data Science*.

Ampliar los datos recogidos anónimos que puedan guardarse en cualquier tipo de contenedor de datos. Con el objetivo de un posterior procesado y análisis para obtener datos mercantiles que poder más tarde ofrecer o explotar.

VI. Marco Regulador.

Dentro del marco legal español, extraeremos las leyes susceptibles de entrar en conflicto con redes sociales. Estas son la Ley Orgánica de Protección de Datos (LOPD), la Ley de Servicios de la Sociedad de la Información y de Comercio Electrónico (LSSI) y por último la Ley de Propiedad Intelectual (LPI).

Este apartado sirve como introducción y se recomienda por defecto consultar siempre a un abogado especializado.

1. Ley Orgánica de Protección de Datos.

Dentro de la LOPD cabe destacar que el público objetivo de la red social será lo que condicione el nivel de seguridad. Citando el artículo 81 del Real Decreto 1720/2007, de 21 de diciembre:

APLICACIÓN DE LOS NIVELES DE SEGURIDAD.

1. TODOS LOS FICHEROS O TRATAMIENTOS DE DATOS DE CARÁCTER PERSONAL DEBERÁN ADOPTAR LAS MEDIDAS DE SEGURIDAD CALIFICADAS DE NIVEL BÁSICO.

2. DEBERÁN IMPLANTARSE, ADEMÁS DE LAS MEDIDAS DE SEGURIDAD DE NIVEL BÁSICO, LAS MEDIDAS DE NIVEL MEDIO, EN LOS SIGUIENTES FICHEROS O TRATAMIENTOS DE DATOS DE CARÁCTER PERSONAL:

a) LOS RELATIVOS A LA COMISIÓN DE INFRACCIONES ADMINISTRATIVAS O PENALES.
b) AQUELLOS CUYO FUNCIONAMIENTO SE RIJA POR EL ARTÍCULO 29 DE LA LEY ORGÁNICA 15/1999, DE 13 DE DICIEMBRE.
c) AQUELLOS DE LOS QUE SEAN RESPONSABLES ADMINISTRACIONES TRIBUTARIAS Y SE RELACIONEN CON EL EJERCICIO DE SUS POTESTADES TRIBUTARIAS.
d) AQUÉLLOS DE LOS QUE SEAN RESPONSABLES LAS ENTIDADES FINANCIERAS PARA FINALIDADES RELACIONADAS CON LA PRESTACIÓN DE SERVICIOS FINANCIEROS.
e) AQUÉLLOS DE LOS QUE SEAN RESPONSABLES LAS ENTIDADES GESTORAS Y SERVICIOS COMUNES DE LA SEGURIDAD SOCIAL Y SE RELACIONEN CON EL EJERCICIO DE SUS COMPETENCIAS. DE IGUAL MODO, AQUELLOS DE LOS QUE SEAN RESPONSABLES LAS MUTUAS DE ACCIDENTES DE TRABAJO Y ENFERMEDADES PROFESIONALES DE LA SEGURIDAD SOCIAL.
f) AQUÉLLOS QUE CONTENGAN UN CONJUNTO DE DATOS DE CARÁCTER PERSONAL QUE OFREZCAN UNA DEFINICIÓN DE LAS CARACTERÍSTICAS O DE LA PERSONALIDAD DE LOS CIUDADANOS Y QUE PERMITAN EVALUAR DETERMINADOS ASPECTOS DE LA PERSONALIDAD O DEL COMPORTAMIENTO DE LOS MISMOS.

3. ADEMÁS DE LAS MEDIDAS DE NIVEL BÁSICO Y MEDIO, LAS MEDIDAS DE NIVEL ALTO SE APLICARÁN EN LOS SIGUIENTES FICHEROS O TRATAMIENTOS DE DATOS DE CARÁCTER PERSONAL:

a) LOS QUE SE REFIERAN A DATOS DE IDEOLOGÍA, AFILIACIÓN SINDICAL, RELIGIÓN, CREENCIAS, ORIGEN RACIAL, SALUD O VIDA SEXUAL.
b) LOS QUE CONTENGAN O SE REFIERAN A DATOS RECABADOS PARA FINES POLICIALES SIN CONSENTIMIENTO DE LAS PERSONAS AFECTADAS.
c) AQUÉLLOS QUE CONTENGAN DATOS DERIVADOS DE ACTOS DE VIOLENCIA DE GÉNERO.

4. A LOS FICHEROS DE LOS QUE SEAN RESPONSABLES LOS OPERADORES QUE PRESTEN SERVICIOS DE COMUNICACIONES ELECTRÓNICAS DISPONIBLES AL PÚBLICO O EXPLOTEN REDES PÚBLICAS DE COMUNICACIONES ELECTRÓNICAS RESPECTO A LOS DATOS DE TRÁFICO Y A LOS DATOS DE LOCALIZACIÓN, SE APLICARÁN, ADEMÁS DE LAS MEDIDAS DE SEGURIDAD DE NIVEL BÁSICO Y MEDIO, LA MEDIDA DE SEGURIDAD DE NIVEL ALTO CONTENIDA EN EL ARTÍCULO 103 DE ESTE REGLAMENTO.

5. EN CASO DE FICHEROS O TRATAMIENTOS DE DATOS DE IDEOLOGÍA, AFILIACIÓN SINDICAL, RELIGIÓN, CREENCIAS, ORIGEN RACIAL, SALUD O VIDA SEXUAL BASTARÁ LA IMPLANTACIÓN DE LAS MEDIDAS DE SEGURIDAD DE NIVEL BÁSICO CUANDO:

a) LOS DATOS SE UTILICEN CON LA ÚNICA FINALIDAD DE REALIZAR UNA TRANSFERENCIA DINERARIA A LAS ENTIDADES DE LAS QUE LOS AFECTADOS SEAN ASOCIADOS O MIEMBROS.
b) SE TRATE DE FICHEROS O TRATAMIENTOS EN LOS QUE DE FORMA INCIDENTAL O ACCESORIA SE CONTENGAN AQUELLOS DATOS SIN GUARDAR RELACIÓN CON SU FINALIDAD.

6. TAMBIÉN PODRÁN IMPLANTARSE LAS MEDIDAS DE SEGURIDAD DE NIVEL BÁSICO EN LOS FICHEROS O TRATAMIENTOS QUE CONTENGAN DATOS RELATIVOS A LA SALUD, REFERENTES EXCLUSIVAMENTE AL GRADO DE DISCAPACIDAD O LA SIMPLE DECLARACIÓN DE LA CONDICIÓN DE DISCAPACIDAD O INVALIDEZ DEL AFECTADO, CON MOTIVO DEL CUMPLIMIENTO DE DEBERES PÚBLICOS.

7. LAS MEDIDAS INCLUIDAS EN CADA UNO DE LOS NIVELES DESCRITOS ANTERIORMENTE TIENEN LA CONDICIÓN DE MÍNIMOS EXIGIBLES, SIN PERJUICIO DE LAS DISPOSICIONES LEGALES O REGLAMENTARIAS ESPECÍFICAS VIGENTES QUE PUDIERAN RESULTAR DE APLICACIÓN EN CADA CASO O LAS QUE POR PROPIA INICIATIVA ADOPTASE EL RESPONSABLE DEL FICHERO.

8. A LOS EFECTOS DE FACILITAR EL CUMPLIMIENTO DE LO DISPUESTO EN ESTE TÍTULO, CUANDO EN UN SISTEMA DE INFORMACIÓN EXISTAN FICHEROS O TRATAMIENTOS QUE EN FUNCIÓN DE SU FINALIDAD O USO CONCRETO, O DE LA NATURALEZA DE LOS DATOS QUE CONTENGAN, REQUIERAN LA APLICACIÓN DE UN NIVEL DE MEDIDAS DE SEGURIDAD DIFERENTE AL DEL SISTEMA PRINCIPAL, PODRÁN SEGREGARSE DE ESTE ÚLTIMO, SIENDO DE APLICACIÓN EN CADA CASO EL NIVEL DE MEDIDAS DE SEGURIDAD CORRESPONDIENTE Y SIEMPRE QUE PUEDAN DELIMITARSE LOS DATOS AFECTADOS Y LOS USUARIOS CON ACCESO A LOS MISMOS, Y QUE ESTO SE HAGA CONSTAR EN EL DOCUMENTO DE SEGURIDAD.

Teniendo en cuenta el texto citado podemos ver que en un principio el nivel básico será aquel que se tenga que llevar a cabo. Pasándose a los siguientes niveles en el momento en que el cliente que ofrezca el servicio de la red social, solicite cualquiera de los datos que se estipulan en los puntos 2 y 3.

2. Ley de los Servicios de la Sociedad de la Información.

Sobre la aplicación de la LSSI, hay que partir de la base de que ha de haber una sección de la web dedicada en exclusiva a dar información tanto para el consumidor como a los organismos del estado de la actividad y fin de la web.

En caso de uso de cookies (de la web o de terceros), se tiene la obligación de avisar al usuario que se hace uso de ellas.

Además, en caso de incumplimiento de cualquiera de los motivos mostrados en el artículo 8, sección 1, el estado podría tomar control total de la web:

ARTÍCULO 8. RESTRICCIONES A LA PRESTACIÓN DE SERVICIOS Y PROCEDIMIENTO DE COOPERACIÓN INTRACOMUNITARIO.

1. EN CASO DE QUE UN DETERMINADO SERVICIO DE LA SOCIEDAD DE LA INFORMACIÓN ATENTE O PUEDA ATENTAR CONTRA LOS PRINCIPIOS QUE SE EXPRESAN A CONTINUACIÓN, LOS ÓRGANOS COMPETENTES PARA SU PROTECCIÓN, EN EJERCICIO DE LAS FUNCIONES QUE TENGAN LEGALMENTE ATRIBUIDAS, PODRÁN ADOPTAR LAS MEDIDAS NECESARIAS PARA QUE SE INTERRUMPA SU PRESTACIÓN O PARA RETIRAR LOS DATOS QUE LOS VULNERAN. LOS PRINCIPIOS A QUE ALUDE ESTE APARTADO SON LOS SIGUIENTES:

A) LA SALVAGUARDA DEL ORDEN PÚBLICO, LA INVESTIGACIÓN PENAL, LA SEGURIDAD PÚBLICA Y LA DEFENSA NACIONAL.
B) LA PROTECCIÓN DE LA SALUD PÚBLICA O DE LAS PERSONAS FÍSICAS O JURÍDICAS QUE TENGAN LA CONDICIÓN DE CONSUMIDORES O USUARIOS, INCLUSO CUANDO ACTÚEN COMO INVERSORES.
C) EL RESPETO A LA DIGNIDAD DE LA PERSONA Y AL PRINCIPIO DE NO DISCRIMINACIÓN POR MOTIVOS DE RAZA, SEXO, RELIGIÓN, OPINIÓN, NACIONALIDAD, DISCAPACIDAD O CUALQUIER OTRA CIRCUNSTANCIA PERSONAL O SOCIAL, Y
D) LA PROTECCIÓN DE LA JUVENTUD Y DE LA INFANCIA.

E) LA SALVAGUARDA DE LOS DERECHOS DE PROPIEDAD INTELECTUAL.

EN LA ADOPCIÓN Y CUMPLIMIENTO DE LAS MEDIDAS DE RESTRICCIÓN A QUE ALUDE ESTE APARTADO SE RESPETARÁN, EN TODO CASO, LAS GARANTÍAS, NORMAS Y PROCEDIMIENTOS PREVISTOS EN EL ORDENAMIENTO JURÍDICO PARA PROTEGER LOS DERECHOS A LA INTIMIDAD PERSONAL Y FAMILIAR, A LA PROTECCIÓN DE LOS DATOS PERSONALES, A LA LIBERTAD DE EXPRESIÓN O A LA LIBERTAD DE INFORMACIÓN, CUANDO ÉSTOS PUDIERAN RESULTAR AFECTADOS.

EN TODOS LOS CASOS EN LOS QUE LA CONSTITUCIÓN Y LAS LEYES REGULADORAS DE LOS RESPECTIVOS DERECHOS Y LIBERTADES ASÍ LO PREVEAN DE FORMA EXCLUYENTE, SÓLO LA AUTORIDAD JUDICIAL COMPETENTE PODRÁ ADOPTAR LAS MEDIDAS PREVISTAS EN ESTE ARTÍCULO, EN TANTO GARANTE DEL DERECHO A LA LIBERTAD DE EXPRESIÓN, DEL DERECHO DE PRODUCCIÓN Y CREACIÓN LITERARIA, ARTÍSTICA, CIENTÍFICA Y TÉCNICA, LA LIBERTAD DE CÁTEDRA Y EL DERECHO DE INFORMACIÓN.

3. Ley de la Propiedad Intelectual.

La aplicación de la LPI para este proyecto podría ser muy extensa, pero uno de los puntos que podría aplicarse por alusión casi directa es el Título VII Programas de Ordenador tras leer el artículo 96, apartado 1.

A LOS EFECTOS DE LA PRESENTE LEY SE ENTENDERÁ POR PROGRAMA DE ORDENADOR TODA SECUENCIA DE INSTRUCCIONES O INDICACIONES DESTINADAS A SER UTILIZADAS, DIRECTA O INDIRECTAMENTE, EN UN SISTEMA INFORMÁTICO PARA REALIZAR UNA FUNCIÓN O UNA TAREA O PARA OBTENER UN RESULTADO DETERMINADO, CUALQUIERA QUE FUERE SU FORMA DE EXPRESIÓN Y FIJACIÓN.

A LOS MISMOS EFECTOS, LA EXPRESIÓN PROGRAMAS DE ORDENADOR COMPRENDERÁ TAMBIÉN SU DOCUMENTACIÓN PREPARATORIA. LA DOCUMENTACIÓN TÉCNICA Y LOS MANUALES DE USO DE UN PROGRAMA GOZARÁN DE LA MISMA PROTECCIÓN QUE ESTE TÍTULO DISPENSA A LOS PROGRAMAS DE ORDENADOR.

Las Bases de Datos también tienen un apartado que hace referencia a ellas:

ARTÍCULO 12. COLECCIONES. BASES DE DATOS.

1. TAMBIÉN SON OBJETO DE PROPIEDAD INTELECTUAL, EN LOS TÉRMINOS DEL LIBRO I DE LA PRESENTE LEY, LAS COLECCIONES DE OBRAS AJENAS, DE DATOS O DE OTROS ELEMENTOS INDEPENDIENTES COMO LAS ANTOLOGÍAS Y LAS BASES DE DATOS QUE POR LA SELECCIÓN O DISPOSICIÓN DE SUS CONTENIDOS CONSTITUYAN CREACIONES INTELECTUALES, SIN PERJUICIO, EN SU CASO, DE LOS DERECHOS QUE PUDIERAN SUBSISTIR SOBRE DICHOS CONTENIDOS.

LA PROTECCIÓN RECONOCIDA EN EL PRESENTE ARTÍCULO A ESTAS COLECCIONES SE REFIERE ÚNICAMENTE A SU ESTRUCTURA EN CUANTO FORMA DE EXPRESIÓN DE LA SELECCIÓN O DISPOSICIÓN DE SUS CONTENIDOS, NO SIENDO EXTENSIVA A ÉSTOS.

2. A EFECTOS DE LA PRESENTE LEY, Y SIN PERJUICIO DE LO DISPUESTO EN EL APARTADO ANTERIOR, SE CONSIDERAN BASES DE DATOS LAS COLECCIONES DE OBRAS, DE DATOS, O DE OTROS

ELEMENTOS INDEPENDIENTES DISPUESTOS DE MANERA SISTEMÁTICA O METÓDICA Y ACCESIBLES INDIVIDUALMENTE POR MEDIOS ELECTRÓNICOS O DE OTRA FORMA.

3. LA PROTECCIÓN RECONOCIDA A LAS BASES DE DATOS EN VIRTUD DEL PRESENTE ARTÍCULO NO SE APLICARÁ A LOS PROGRAMAS DE ORDENADOR UTILIZADOS EN LA FABRICACIÓN O EN EL FUNCIONAMIENTO DE BASES DE DATOS ACCESIBLES POR MEDIOS ELECTRÓNICOS.

Como en los casos anteriores, se recomienda consultar con un abogado especializado. Sobre todo al tratarse de una red social, donde mucha parte del contenido procede de los usuarios y es necesario saber en qué medida es cada parte responsable del contenido publicado.

VII. Bibliografía.

[1] Iglesia Sabio, Javier "Diseño y desarrollo del software del servidor para un sistema para gestión de redes sociales geoposicionadas" UC3M 2013

[2] Kuppusamy, Prabhakaran; Vyas, Uchit "AWS development essentials: design and build flexible, highly scalable, and cost-effective applications using Amazon web services" Packt Publishing, 2014.

[3] Webber, Jim; Parastatidis, Savas; Robinson, Ian "REST in Practice: Hypermedia and SystemsArchitecture" O'Reilly Media, Incorporated, 09/2010.

[4] Freato, Roberto; Mackenzie, Neil "Microsoft Azure development cookbook: over 70 advanced recipes for developing scalable services with the Microsoft Azure platform" Packt Publishing, 2014.

[5] Monitorización de constantes vitales DIY http://www.nightscout.info/

[6] Cuadrante de competencia en Cloud IaaS del Q1 de 2015 https://www.gartner.com/doc/3056019/magic-quadrant-cloud-infrastructure-service

[7] Ghosh, Subhasish AWS, Microsoft Azure & Google in Gartner Magic Quadrant for IaaS 2015 "https://www.linkedin.com/pulse/aws-microsoft-azure-google-gartner-magic-quadrant-iaas-ghosh" 23 de Mayo de 2015.

[8] Artículo que compara AWS y Azure a nivel de servicios https://www.scriptrock.com/articles/aws-vs-azure

[9] Cambio del rol en toma de decisiones http://www.enter.co/chips-bits/enterprise/el-cambio-en-el-rol-del-cio-en-las-organizaciones/

[10] Definición servicios web W3C
http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb

[11] Amazon Web Services, Inc. AWS Simple Icons para diagramas de arquitectura
"https://aws.amazon.com/es/architecture/icons/" 2015

[12] Gregg, Brendan "Linux perf tools" O'Reilly, 2015

[13] Lindsey, Freddie; Publishing a static website with amazon web services. "http://news.mlh.io/publishing-a-static-website-with-amazon-web-services-09-15-2015?adbsc=social_blogs_20150918_52602136&adbid=644923905090568192&adbpl=tw&adbpr=66780587" 7 de Septiembre de 2015.

[14] LOPD "Real Decreto 1720/2007, de 21 de diciembre, por el que se aprueba el Reglamento de desarrollo de la Ley Orgánica 15/1999, de 13 de diciembre, de protección de datos de carácter personal. (Texto consolidado. Última modificación: 8 de marzo de 2012)." http://www.agpd.es/

[15] LSSI "http://www.boe.es/buscar/act.php?id=BOE-A-2002-13758" BOE, 10 de Mayo de 2014 (última modificación).

[16] LPI "http://www.boe.es/buscar/act.php?id=BOE-A-1996-8930" BOE, 5 de Noviembre de 2014 (última modificación).