Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Diseño e implementación de una red social para animales de compañía

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: David Mifsud Mengual

Tutor: Sergio Sáez Barona

2014-2015

Resumen

El siguiente Proyecto consiste en desarrollar una red social con la temática de los animales de compañía, centrándose en las funciones de comunicación entre usuarios mediante el uso de publicaciones y mensajes. Se ha usado como lenguaje de programación PHP, HTML, CSS y AJAX. Algunas de las funciones de las que está dotada la red social son: solicitud de amistad, notificaciones, mensajes y publicaciones.

Palabras clave: PHP, MySQL, HTML, CSS, AJAX, Red Social, CodeIgniter.

Tabla de contenidos

Lista de figuras

1. Introducción

La creación de la web 2.0 ha revolucionado las relaciones mediante el uso de internet, esto ha ocasionado la aparición de numerosas redes sociales en las cuales puedes encontrar personas con tus mismos intereses o aficiones.

La finalidad de este proyecto es la realización de un *software* usando las técnicas adecuadas de análisis y que cumpla con los requisitos obtenidos en la fase de análisis, para ello se aplicarán los conocimientos adquiridos en las asignaturas ISW y GPR correspondientes al Grado en Ingeniería Informática.

Mi motivación principal para escoger este proyecto es el potencial uso de las redes sociales, y el hecho que la creación de una red social cubre todos los aspectos necesarios para poder crear otras aplicaciones web, además de ser una oportunidad de poder ampliar mis conocimientos en la creación de software y más concretamente en elaborar páginas web dinámicas.

1.1. Objetivo

El objetivo es el diseño y la implementación de una red social con la temática de mascotas, ésta tendrá en una sola cuenta de usuario varias mascotas, ya que la tendencia de las personas que tienen animales es que suelen tener varios animales, de esta forma se cubrirán las necesidades de todos los usuarios.

Las mascotas en este proyecto constituirán los perfiles públicos ya que todas las interacciones entre usuarios se llevarán a cabo a través de éstas, es decir una red multiperfil en la que la finalidad de las cuentas de usuario es autenticarse en el sistema.

2. Estado del arte

Antes de empezar a realizar este proyecto, me documenté en algunas webs para observar cuáles eran las funcionalidades y cómo abordar la realización de una red social de este tipo.

A pesar de que existen bastantes redes sociales, no encontré ninguna en la que se pudiesen tener varios animales en una sola cuenta de usuario. También he de destacar, que la mayor parte de las redes sociales de esta temática suelen estar dedicadas específicamente a un determinado tipo de animal (por ejemplo perros, gatos, peces…).

Seguidamente expondré dos de las mejores redes con esta misma temática que he encontrado en la red.

2.1. Facepets

Facepets es una red social con la temática de animales de compañía, la cual posee como la inmensa mayoría de redes sociales la capacidad de publicar en un muro, subir fotos, enviar mensajes a otros usuarios y enviar peticiones de amistad. Esta red es casi una aproximación a lo que se quiere realizar en este TFG.

Por otra parte esta red social parece estar bastante encaminada a un uso comercial, ya que dispone de una tienda online integrada donde se venden productos para mascotas, lo cual tiene bastante sentido para monetizar esta red social. Además hay que destacar que esta red no es específicamente para un tipo de animal a diferencia de otras, puesto que se puede registrar todo tipo de mascotas.

Sin embargo veo que tiene algunas desventajas, sin entrar en temas de fiabilidad y seguridad de la red: no hay un control sobre los usuarios que se registran, ya que no se envía un mail de activación ni ningún tipo de comprobación. Además esta red es monoperfil y no contempla que un usuario pueda tener más de una mascota.

Figura 1 Captura de la portada de la web facepets.com

2.2. Dogster

Dogster es un foro inglés dedicado a los amantes de los perros, es uno de los claros ejemplos de páginas web que se dedican a un animal específico.

Entre sus características destacables está el hecho de poder crear más de una mascota con un solo usuario, además de otras funcionalidades como transferir un animal a otro perfil. Sin embargo esta web tiene sus desventajas en la interfaz de usuario, la cual es muy difícil de manejar, siendo éste su talón de Aquiles.

Figura 2 Captura de la portada de la web dogster.com

3. Especificación de requisitos software

En esta sección se realizará una descripción sobre cómo debe comportarse el sistema que se va a desarrollar. Esta especificación está dirigida tanto al cliente como al desarrollador ya que establece un contrato acerca del proyecto.

3.1. Propósito

Dedicaremos esta sección a la especificación de requisitos software según el estándar IEEE 830/1998 que usaré en este TFG.

3.2. Descripción general

En las siguientes secciones se describirá el producto definiendo hacia que usuarios va orientado, así como explicar cuál es la idea de lo que se pretende realizar y cuáles son las funciones que deberán ser cubiertas en este proyecto.

3.2.1. Perspectiva del producto

Lo que se va a desarrollar en este proyecto es una red social, en la cual, cada usuario podrá dar de alta a todas sus mascotas, hacer amigos con ellas e interactuar usando los perfiles de las mascotas. En este producto, el usuario será un mero trámite, ya que todas las acciones en las que interactuará con otros usuarios se llevarán a cabo a través de los perfiles de las mascotas, es decir, cada mascota tendrá sus amigos, sus fotos, sus publicaciones... Cabe destacar que este tipo de red social se debería clasificar como una red multiperfil ya que los perfiles son los de las mascotas.

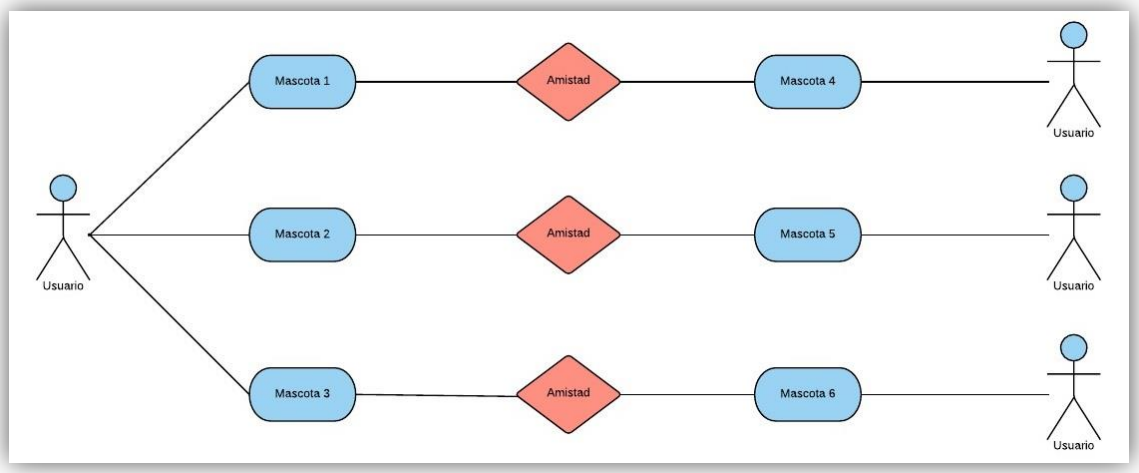

Figura 3 *Ilustración de las relaciones entre los perfiles de las mascotas*

3.2.2. Funcionalidad del producto

Las funcionalidades que cubrirá esta red social, son principalmente la de comunicación con otros usuarios. En este caso, otras mascotas mediante el uso de publicaciones y mensajes. También se proporcionara la funcionalidad de buscar nuevos amigos, cambiar las imágenes del perfil de cada usuario y mascota y la gestión de la activación de nuevos usuarios mediante confirmación de un correo electrónico. De esta forma tendremos usuarios reales, no creados mediante bots, evitando que nuestra base de datos se sature a causa de un exceso de nuevos usuarios inactivos.

3.2.3. Características de los usuarios

Las características del usuario de este tipo de red deben ser las de una persona que participe activamente en redes sociales convencionales, foros o incluso blogs. Aunque no se requiera de una experiencia muy extensa en el uso de las redes sociales, cualquier persona con pocos conocimientos en el uso de redes sociales podría perfectamente adaptarse a su uso.

Adjunto una tabla a modo de resumen del tipo de usuario que encajaría en el perfil.

Usuario	Usuario red social
Nivel educativo	No se exige formación específica
Experiencia	Aconsejable previa experiencia en el uso de redes sociales convencionales (Facebook, Twitter, Linkedin)
Experiencia técnica	Ninguna

3.3. Requisitos específicos

A continuación, se explicarán los requisitos que deberá cumplir la aplicación. Todos ellos deberán ser satisfechos una vez finalizada la aplicación ya que los mismos son un acuerdo entre el cliente y el desarrollador.

En las siguientes secciones, se explicarán los requisitos funcionales, que son las funciones que tiene que llevar a cabo la aplicación. Por otra parte los requisitos de rendimiento, abarcarán las acciones que deberán ser tomadas una vez desarrollado el proyecto y éste esté implementado en un servidor web totalmente funcional. Para ello se definirán algunas políticas o reglas que deberán ser llevadas a cabo y por último los requisitos de base de datos especificarán las políticas que se deberán llevar a término para preservar la información.

3.3.1. Requisitos funcionales

Como resultado del caso de estudio, se implementarán las siguientes funcionalidades:

Nombre	Envío de solicitudes de amistad a las mascotas y gestión de las mismas.
Tipo	Requisito TFG.
Descripción General	Las mascotas podrán enviar solicitudes de amistad a otras mascotas.
Entradas	Enviar solicitud de amistad.
Salidas	El sistema nos indicará que la petición se ha enviado y que será evaluada próximamente.
Descripción del proceso	El usuario busca las mascotas desde su panel de control, envía una petición de amistad a la mascota y esta deberá evaluar si acepta la petición, en caso afirmativo los solicitantes podrán comentar los estados del muro de la mascota al igual que crear nuevos estados en su muro.

Nombre	Búsqueda de otras mascotas registradas.
Tipo	Requisito TFG.
Descripción General	Los usuarios una vez se hayan autenticado como mascota podrán realizar búsquedas para poder encontrar amigos.
Entradas	Raza y región o país.
Salidas	Se mostrarán las mascotas que cumplan los requisitos de búsqueda.
Descripción del proceso	El usuario busca las mascotas según la raza y/o la región que desee, una vez realizada esta búsqueda el sistema mostrará las que cumplan los requisitos de búsqueda, y el usuario podrá acceder al perfil de la mascota como invitado.

Nombre	Publicar en el muro de una mascota amiga.
Tipo	Restricción.
Descripción General	Las publicaciones son la base de esta red social, ya que es la forma principal de entablar una conversación entre las mascotas. Esta acción está limitada a las mascotas amigas.
Entradas	Comentario o publicación realizada.
Salidas	El texto quedará publicado en el muro de la mascota amiga.
Descripción del proceso	Un usuario podrá crear un nuevo estado en el muro de una mascota amiga si existe una relación de amistad entre ambos. Para ello, el usuario realizará un nuevo estado mediante el uso de una “textarea” que estará disponible en el muro de la mascota amiga. El usuario también podrá comentar los estados existentes.

Nombre	Publicar en el muro propio.
Tipo	Requisito TFG.
Descripción General	Todos los usuarios podrán publicar en los muros de las mascotas propias, esta publicación quedará a la vista tanto de las mascotas amigas como de los desconocidos, aunque solo podrá ser comentado por mascotas amigas.
Entradas	Comentario o publicación realizada.
Salidas	La publicación o el comentario quedará plasmado en el muro de nuestra mascota.
Descripción del proceso	Accederemos a nuestro perfil y dispondremos de un área en la que poder crear un nuevo estado, también puede darse el caso de que queramos comentar un estado existente, en ese caso procederemos realizando el comentario en el área de texto del propio estado.

Nombre	Gestión de las publicaciones realizadas en el muro de las mascotas.
Tipo	Restricción.
Descripción General	Los usuarios autenticados como una determinada mascota, podrán administrar sus publicaciones y comentarios que hayan realizado, así como las publicaciones y comentarios realizados por otras mascotas en su perfil de mascota, pudiendo este eliminar los comentarios por cualquier razón. Esta funcionalidad estará limitada al propietario del perfil o a la mascota que ha realizado el comentario o publicación.
Entradas	Eliminar comentario mediante un botón o hipertexto.
Salidas	Se eliminará el comentario o publicación del muro.
Descripción del proceso	El usuario de la aplicación autenticado como mascota eliminará las publicaciones o comentarios que crea que no son apropiados en su muro, para ello deberá localizarlo y eliminarlo mediante un enlace que deberá estar dentro del encabezado del comentario o publicación.

Nombre	Envíos de mensajes a otras mascotas.
Tipo	Requisito TFG.
Descripción General	Los usuarios podrán enviar y recibir mensajes de otras mascotas sin restricción alguna. Esta forma de comunicación estará destinada a la comunicación privada entre usuarios.
Entradas	Texto del mensaje.
Salidas	El mensaje aparecerá en la cuenta de la mascota a la que se enviará.
Descripción del proceso	Para enviar un mensaje a otra mascota el usuario deberá acceder al perfil de la mascota a la que quiere enviar el mensaje y pulsar sobre el apartado dedicado a este menester, se redirigirá a una sección en la que el usuario podrá enviar el mensaje.

Nombre	Registros de mascotas.
Tipo	Requisito TFG.
Descripción General	Los usuarios podrán registrar tantas mascotas como deseen, no existirá límite. Las mascotas constituirán el perfil que usarán los usuarios.
Entradas	Nombre de la mascota, sexo, raza …
Salidas	Se creará una mascota que estará disponible en el menú del usuario.
Descripción del proceso	Un usuario accederá al apartado de registro de mascota, rellenará los datos y si estos son correctos, la mascota quedará registrada en el sistema. En cambio si hubiese alguna incongruencia durante el registro, el sistema alertaría al usuario de que la información proporcionada no puede ser usada para crear la mascota y esta información deberá ser cambiada para finalizar el registro.

Nombre	Modificación de los datos de la mascota.
Tipo	Requisito TFG.
Descripción General	Los usuarios podrán modificar la información de la mascota y actualizarla según crean conveniente.
Entradas	Raza, foto perfil ...
Salidas	La información modificada quedará expuesta en el perfil de la mascota.
Descripción del proceso	Cada apartado que pueda ser modificado dispondrá de una sección por la cual la información podrá ser modificada por el usuario propietario de la mascota.

Nombre	Registro de usuario.
Tipo	Requisito TFG.
Descripción General	Cualquier persona podrá formar parte de esta red social. Para realizar este proceso de registro solo deberá seguir los pasos que se le indicarán en esta sección.
Entradas	Información del usuario: email, nombre, apellidos, contraseña, sexo, edad ...
Salidas	Envío de email de confirmación.
Descripción del proceso	Un usuario deberá cumplimentar el formulario de registro con información válida. Una vez haya sido comprobada si esta es correcta, se enviará un enlace al email del usuario que deberá pulsar para finalizar el proceso de registro.

Nombre	Modificación de datos de usuario.
Tipo	Opcional.
Descripción General	Los datos del usuario podrán ser modificados mediante el envío de la nueva información, a través de los formularios disponibles en los apartados modificables.
Entradas	Información personal a modificar: edad, imagen de perfil ...
Salidas	La información quedará actualizada en el perfil del usuario.
Descripción del proceso	Se accederá a cada sección que se quiera modificar, para realizar este proceso, se podría habilitar un enlace en cada apartado donde queramos modificar los datos o podrá existir una sección en la que se pueda modificar toda la información.

Nombre	Autenticación de usuario en el sistema.
Tipo	Requisito TFG.
Descripción General	Los usuarios podrán acceder a su sección mediante las credenciales que proporcionaron al registrarse.
Entradas	Email y contraseña.
Salidas	Acceso a la sección propia del usuario.
Descripción del proceso	Un usuario que quiera acceder a esta red social deberá ingresar en el formulario de entrada su email con el que se registro y la contraseña, los cuales serán comprobados por el sistema y en caso de que hayan sido activados mediante el enlace de confirmación enviado a su email podrán acceder.

3.3.2. Requisitos de rendimiento

Los requisitos de rendimiento de esta aplicación no serán relativamente importantes, ya que la misma se ha desarrollado con fines didácticos y sin la finalidad de explotarla comercialmente, por lo tanto estos requisitos no constituyen una parte tan importante del proyecto, aún así se tomarán medidas para que funcione de forma óptima y el proyecto incluso pueda ser usable con fines comerciales.

Seguridad

La aplicación deberá cumplir unos mínimos de seguridad, ya que se almacenaran datos de carácter personal como pueden ser nombre, apellidos y el email, el cual está protegido por la LOPD y no debe caer en manos de otras personas, por lo tanto el acceso a la BD estará limitado a los usuarios que se conecten desde "localhost", evitando las conexiones remotas, de esta forma, estos datos quedarían más protegidos.

Por otra parte las contraseñas se almacenaran de forma encriptada mediante encriptación md5 en la base de datos.

Se empleara la confirmación de creación de un nuevo usuario mediante un email de activación que se envía al usuario, haciendo que la información del sistema sea un poco más verídica.

Fiabilidad

Al ser una versión beta se toleraran los fallos, ya que hacer un test completo será imposible de realizar, ya que se necesitaran usuarios reales y un periodo de uso extenso. No obstante, las operaciones deberían llevarse a cabo sin ningún problema.

Disponibilidad

La disponibilidad del sistema deberá ser como mínimo del 90% del tiempo, y en caso de que hubiese que ampliar el proyecto o realizar tareas de mantenimiento, éstas deberían llevarse a cabo en las horas de menor uso del sistema, para evitar cualquier molestia a los usuarios.

Mantenimiento

Las tareas de mantenimiento se deberán realizar por los desarrolladores y administradores del sitio web. Habrá varios tipos de tareas:

- Tareas programadas o *cronjobs*, éstas serán las tareas programadas dentro de un archivo PHP las cuales se realizan automáticamente en el periodo de tiempo que estén programadas, un claro ejemplo es la eliminación de los usuarios no activados o la eliminación de usuarios que no han usado el sistema en un periodo de tiempo muy prolongado en el tiempo.

- Tareas de remodelación o reparación de errores, al ser una versión beta, pueden surgir cambios y reparación de algunas partes o módulos, estas tareas serán llevadas a cabo por el programador del proyecto.

- Tareas de administración del sitio web, estas tareas serán llevadas por los administradores, al ser una versión beta el administrador no cuenta con un panel de control y por lo tanto las tareas sobre la base de datos se realizarán manualmente. Entre las tareas del administrador del sitio web estarán las de generar estadísticas del número de usuarios que han usado la aplicación, para ello se deberá crear un campo donde se almacene la fecha del último login del usuario.

Portabilidad

La portabilidad de una aplicación comprende todos los atributos necesarios para facilitar su traslado a otro servidor, en este caso sera necesario:

- La exportación de los archivos PHP generados a la carpeta htdocs al nuevo servidor.

- La configuración de un usuario en la base de datos, así como en caso que hubiese cambios en el nombre de usuario, el nombre de la base de datos o la contraseña de la misma, se deberá modificar el archivo donde se almacena la configuración de la conexión de la base de datos.

- La creación de las tablas mediante la ejecución del script que contiene las tablas de la base de datos.

- Configurar el servidor de correos para el envío de los emails de confirmación.

Otros requisitos

En el caso de que esta red social se quisiese implementar para uso comercial, se deberá asignar un responsable de los archivos de carácter personal como son nombres, apellidos, o incluso el email, ya que estos están amparados por la LOPD y podríamos tener problemas legales en caso de que alguien decidiese usar la lista de correos para enviar SPAM.

A modo de ampliación del proyecto, una vez esté implementado en el servidor definitivo, se deberán realizar las pruebas apropiadas para medir que cantidad de recursos hardware son necesarios para que el sistema funcione de forma óptima. Para ello, se hará una previsión del número de usuarios al igual que el número de usuarios que se espera que usen el sistema de forma simultánea. De esta forma, podremos sacar las medidas de la cantidad de recursos que necesitaremos, no conviene tener mucho margen de recursos sin usar, ya que éstos cuestan de mantener y por lo tanto la previsión debe dejar un margen estrecho de recursos que estén sin uso, puesto que podrían producirse picos de uso momentáneos que necesitaran de estos recursos.

3.3.3. Requisitos base de datos

Para la realizar este proyecto usaré el sistema de gestión de bases de datos MySQL, el cual es *Open Source* y por lo tanto no es necesaria la adquisición de una licencia.

La elección de este tipo de BD es debido a su extendido uso y la facilidad de creación de las tablas, ya que pueden ser creadas fácilmente en el gestor phpmyadmin o manualmente.

Los requisitos de la base de datos de este proyecto serán que la información quede almacenada de forma íntegra y no se produzcan borrados accidentales. Entre algunas de las acciones que se deberán llevar a cabo es la creación de contraseñas seguras para la base de datos, se deberán definir contraseñas que cumplan un mínimo de seguridad, por ejemplo usar números, signos y letras. Otro de los requisitos será que el servidor sea capaz de almacenar la base de datos durante el crecimiento de la red social, para ello el servidor deberá disponer de suficiente espacio en el disco duro. El servidor de BD definitivo deberá soportar el número de conexiones que se realicen al mismo, para ello se deberán hacer pruebas para comprobar qué tamaño de tráfico es capaz de soportar.

4. Análisis

En la siguiente sección, se definirá una estructura válida que deberá seguirse para alcanzar los objetivos en la creación de este producto, para realizar esta tarea se usarán diagramas que explicarán cómo están estructuradas las entidades de este proyecto. En otra sección se mostrarán los distintos casos de uso que se deberán realizar para cumplir los objetivos.

4.1. Diagrama Entidad-Relación

En este apartado se definirán las relaciones entre las distintas entidades, así como su multiplicidad. Los atributos de cada tabla se definirán posteriormente en la fase de diseño.

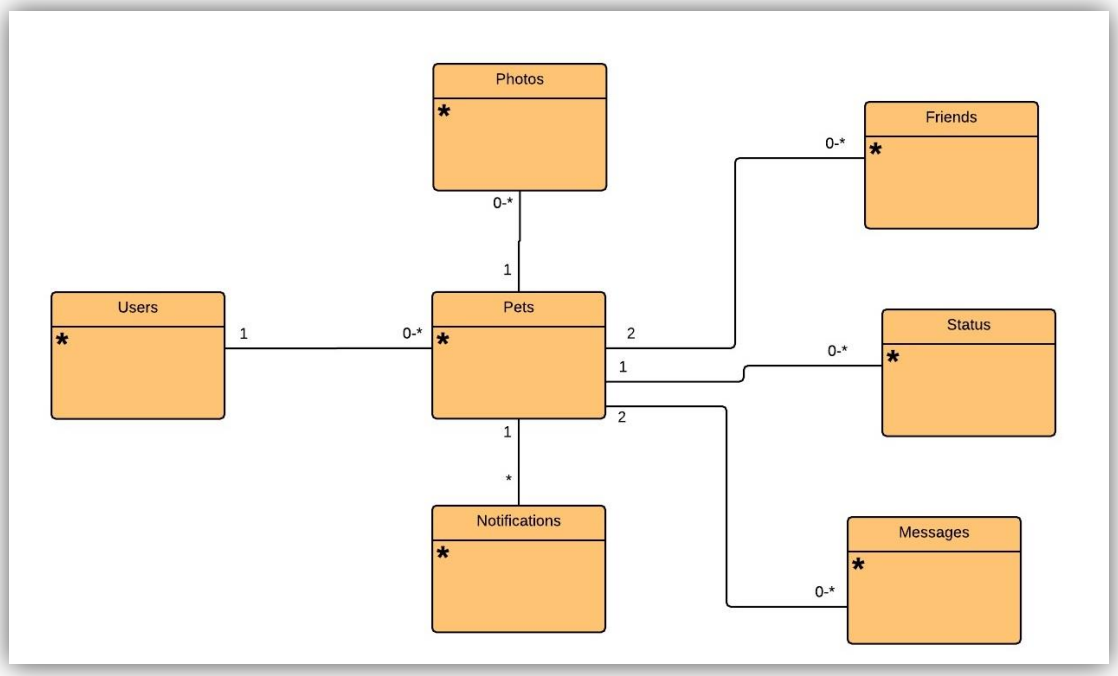

Figura 4 *Diagrama entidad relación*

Tal y como se muestra en el diagrama anterior, un usuario puede tener ninguna o varias mascotas, y una mascota pertenece siempre a un usuario. Una mascota puede tener ninguna, una o varias fotos, y una foto pertenece únicamente a una mascota. Un animal puede no tener ninguna relación de amistad o tener varias, por otra parte una relación de amistad pertenece obligatoriamente a dos mascotas, una de ellas es el solicitante y otro el solicitado. Un estado pertenece obligatoriamente a una mascota concreta, y una mascota puede tener ninguno, uno o varios estados. Un mensaje pertenece a una mascota o a dos, depende de si se quiere mantener un registro de quién

realiza los envíos de mensajes, en cuyo caso quedará como multiplicidad 2, ya que existirá tanto el emisor del mensaje como el receptor, y una mascota puede tener ninguno, uno o varios mensajes. Una mascota puede tener ninguna, una o varias notificaciones, y una notificación pertenece únicamente a una mascota.

4.2. Casos de uso

Al proceder con los diagramas de casos de uso se ha definido un solo rol, ya que la parte del administrador quedará pendiente como ampliación del proyecto, puesto que el mismo puede ser ampliado de forma que la administración de las incidencias pueda quedar en manos de personal sin conocimientos de administrar sitios web, pudiendo ser administrada desde un panel de control. La siguiente imagen representa los casos de uso, los cuales serán explicados con más detalle en sus correspondientes tablas.

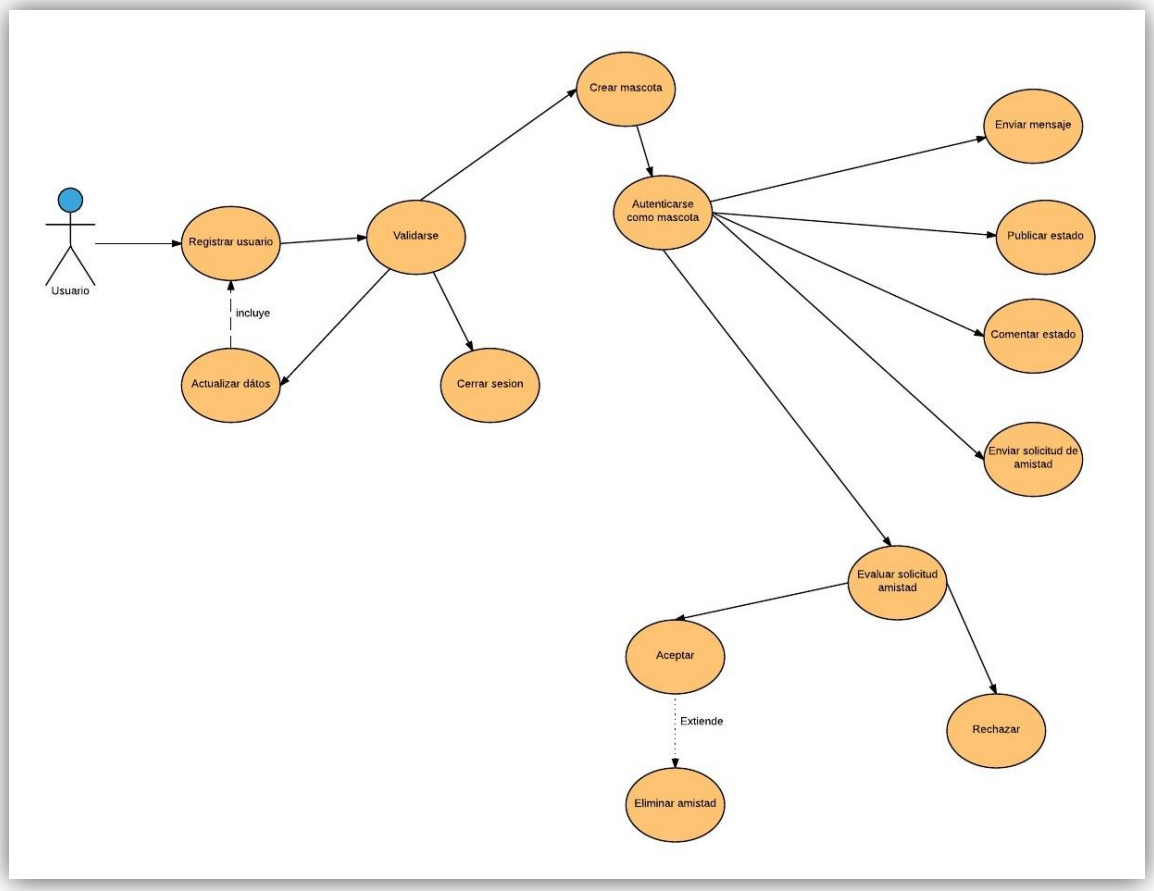

Figura 5 *Esquema casos de uso aplicación anisocial*

A continuación se exponen con más detalle los casos de uso mencionados en el diagrama anterior.

Referencia	001
Caso de uso	Registrar usuario
Actores	Usuario web
Resumen	En este caso de uso los usuarios que visiten nuestra web podrán registrar una cuenta de usuario con sus datos personales
Precondiciones	Ninguna
Pos condiciones	El usuario deberá confirmar la creación mediante algún método de verificación, demostrando que no ha sido un usuario creado mediante un programa malicioso.
Flujo de eventos	
Usuario	Usuario no registrado
Comentarios	A este registro solo puede acceder los usuarios que no están validados en el sistema

Referencia	002
Caso de uso	Actualizar datos
Actores	Usuario web
Resumen	Los usuarios registrados podrán modificar la información almacenada referente a sus datos personales.
Precondiciones	Estar registrado en el sistema y haber activado la cuenta con éxito. Estar autenticado en el sistema
Pos condiciones	Los datos son actualizados
Flujo de eventos	1.Validarse en el sistema
Usuario	Usuario registrado
Comentarios	Los datos deberán ser comprobados con las mismas reglas a las que se somete el usuario durante su registro para evitar inconsistencias en la información proporcionada

Referencia	003
Caso de uso	Validarse
Actores	Usuario
Resumen	En caso de que el usuario este dado de alta y activado, podrá acceder a su cuenta
Precondiciones	Estar registrado en el sistema
Pos condiciones	Una vez validado dentro del sistema se creará una sesión de usuario
Flujo de eventos	1.El usuario se registra y confirma el registro
Usuario	Usuario registrado
Comentarios	Se comprobará que el usuario y la contraseña son correctos

Referencia	004
Caso de uso	Cerrar sesión
Actores	Usuario
Resumen	El usuario cierra sesión saliendo del sistema.
Precondiciones	Tener iniciada una sesión en el sistema
Pos condiciones	La sesión se cierra
Flujo de eventos	1. El usuario tiene una sesión activa 2. El usuario pulsará en el botón salir de la barra superior
Usuario	Usuario registrado
Comentarios	Al cerrar sesión se cierra también la autenticación de la mascota en la que estemos identificados en ese momento.

Referencia	005
Caso de uso	Crear Mascota
Actores	Usuario
Resumen	Un usuario podrá crear tantas mascotas como desee, que quedarán almacenadas en su cuenta de usuario
Precondiciones	Tener una cuenta de usuario que esté activa, además el usuario deberá estar validado en el sistema
Postcondiciones	La mascota creada aparecerá en el menú de las mascotas del usuario y a partir de ese momento el usuario podrá estar autenticado en el sistema como la mascota que ha creado.
Flujo de eventos	1. El usuario tiene una sesión activa 2. Accede al apartado crear mascota 3. Completa el formulario con datos correctos y pulsa crear
Usuario	Usuario registrado
Comentarios	El usuario deberá crear por lo menos una mascota, ya que sin ésta no puede realizar ninguno de los siguientes casos de uso que se explicarán a continuación.

Referencia	006
Caso de uso	Autenticarse como Mascota
Actores	Usuario
Resumen	El usuario podrá estar autenticado como una de las mascotas que ha registrado previamente.
Precondiciones	Tener una cuenta de usuario activada y tener la mascota registrada previamente.
Postcondiciones	El usuario tendrá una sesión con la mascota seleccionada.
Flujo de eventos	1. El usuario se valida en el sistema con una cuenta de usuario activa 2. Pulsa sobre una de las mascotas registradas en la cuenta del usuario
Usuario	Usuario registrado
Comentarios	La autentificación de la mascota se utilizará para poder usar las funciones de red social en nombre de la mascota.

Referencia	007
Caso de uso	Enviar Mensaje
Actores	Usuario
Resumen	Los usuarios podrán comunicarse mediante los perfiles de las mascotas ya que éstas poseen todas las funciones de comunicación.
Precondiciones	Tener una cuenta de usuario activa, tener una mascota registrada y estar autenticado como mascota
Postcondiciones	El usuario destinatario recibirá el mensaje
Flujo de eventos	1. El usuario inicia sesión 2. Selecciona una de las mascotas que tiene registradas y se autentica 3. Accede al perfil de la mascota a la que quiere enviar el mensaje 4. Pulsa enviar mensaje 5. Redacta el mensaje y lo envía
Usuario	Usuario registrado
Comentarios	Todas las comunicaciones se deben realizar mediante las cuentas de usuario, ya que es uno de los requisitos del TFG

Referencia	008
Caso de uso	Publicar estado
Actores	Usuario
Resumen	Las mascotas pueden publicar estados en los muros propios y en los de sus amigos.
Precondiciones	Poseer una cuenta de usuario activa y estar autenticado en el sistema, tener una mascota registrada y estar autenticado como mascota
Postcondiciones	El estado quedará disponible a otros usuarios que podrán visualizarlo y comentarlo
Flujo de eventos	1. El usuario inicia sesión 2. Selecciona una de las mascotas que tiene registradas y se autentica 3. Accede al muro de la mascota donde quiere publicar el estado, en caso de que sea en el propio muro no será necesario este paso 4. Redacta el nuevo estado y lo publica
Usuario	Usuario registrado
Comentarios	Los estados solo pueden ser borrados por el propietario del muro en el que se escriba y por el creador del estado (que puede ser un amigo o el propietario)

Referencia	009
Caso de uso	Comentar estado
Actores	Usuario
Resumen	Una mascota que tenga una relación de amistad con otra, puede comentar un estado en el muro de ésta, sin embargo si no existe la relación de amistad no podrá crearse ningún comentario en el muro de esa mascota.
Precondiciones	Tener una sesión de usuario activa en el sistema, estar autenticado como una de las mascotas registradas, y que la publicación esté en el perfil de una mascota amiga o en el muro de la mascota desde el que comentamos el estado.
Postcondiciones	La respuesta podrá visualizarse en el estado, esta respuesta solo podrá ser eliminada por el usuario o el propietario del perfil.
Flujo de eventos	1. El usuario inicia sesión 2. Selecciona una de las mascotas que tiene registradas y se autentica 3. Accede al muro de la mascota 4. Redacta un comentario y lo publica
Usuario	Usuario registrado
Comentarios	Al comentar un estado se deberá enviar una notificación a todas las mascotas amigas para puedan ver que ha habido cambios en la publicación.

Referencia	010
Caso de uso	Enviar solicitud de amistad
Actores	Usuario
Resumen	Una mascota puede enviar solicitudes de amistad a otras mascotas, las cuales podrán decidir si aceptarla o rechazarla
Precondiciones	Tener una sesión activa y estar autenticado como una mascota
Postcondiciones	Establecer una relación de amistad, estableciendo nuevos privilegios.
Flujo de eventos	1. El usuario inicia sesión 2. Selecciona una de las mascotas que tiene registradas y se autentica 3. Hay dos alternativas, la primera es acceder al muro de la mascota y enviar la solicitud de amistad, la otra opción es acceder desde el menú de nuestra mascota a la pestaña nuevas mascotas 4. Pulsar enviar solicitud
Usuario	Usuario registrado
Comentarios	Las solicitudes de amistad determinarán si ambos usuarios podrán comunicarse, este sistema de amistad es bidireccional, si no es aceptado por ambos no podrá haber comunicación

Referencia	011
Caso de uso	Evaluar solicitud de amistad
Actores	Usuario
Resumen	Una mascota puede recibir peticiones de amistad de otras mascotas, éstas deberán ser evaluadas por la mascota y de ser aceptada, dará acceso a la otra mascota a que pueda publicar estados en el perfil de esta y responder estados
Precondiciones	Tener una sesión de usuario abierta, estar autenticado como una de las mascotas registradas por el usuario y haber recibido una petición de amistad.
Postcondiciones	Según el resultado de la evaluación se asignarán unos privilegios al usuario o se le negarán, ver "Aceptar/Cancelar (Evaluar solicitud de amistad)"
Flujo de eventos	1. Iniciar un sesión 2. Autenticarse como una de las mascotas registradas 3. Acceder al apartado de solicitudes de amistad pendientes 4. Evaluar las posibles peticiones de amistad mediante los botones aceptar o cancelar
Usuario	Usuario registrado
Comentarios	Hay tres posibles estados: Aceptar, rechazar y no responder, éste último no queda representado dentro del diagrama ya que no responder significa que la acción se queda en pendientes.

Referencia	012
Caso de uso	Aceptar (Evaluar solicitud de amistad)
Actores	Usuario
Resumen	La acción de aceptar una solicitud de amistad implica el hecho de dejar acceso a este usuario a comentar en tu muro publico
Precondiciones	Tener una sesión de usuario abierta, estar autenticado como una de las mascotas registradas por el usuario y haber recibido una petición de amistad.
Postcondiciones	Una petición aceptada asignará privilegios de comunicación con la mascota.
Flujo de eventos	1. Iniciar un sesión 2. Autenticarse como una de las mascotas registradas 3. Acceder al apartado de solicitudes de amistad pendientes 4. Aceptar la petición
Usuario	Usuario registrado
Comentarios	En caso de que nos arrepintiésemos de aceptar al usuario siempre podremos eliminarlo de nuestros amigos con el caso de uso con referencia 014

Referencia	013
Caso de uso	Rechazar (Evaluar solicitud de amistad)
Actores	Usuario
Resumen	En el caso que decidiésemos rechazar una petición de amistad, implicaría que esta mascota sigue sin poder comentar ninguna de nuestras publicaciones del muro y tampoco crear nuevos estados, pero no implica que el usuario esté bloqueado, ya que nuevamente podrá volver a enviar otra petición de amistad
Precondiciones	Tener una sesión de usuario abierta, estar autenticado como una de las mascotas registradas por el usuario y haber recibido una petición de amistad.
Postcondiciones	El usuario no podrá comunicarse con nosotros mediante el perfil de la mascota creando nuevos estados o respondiendo a nuestras publicaciones
Flujo de eventos	1. Iniciar un sesión 2. Autenticarse como una de las mascotas registradas 3. Acceder al apartado de solicitudes de amistad pendientes 4. Cancelar la petición
Usuario	Usuario registrado
Comentarios	Una vez eliminada la petición, el usuario podrá volver a enviarla si así lo desea.

Referencia	014
Caso de uso	Eliminar amistad
Actores	Usuario
Resumen	Una vez aceptada una petición podemos eliminar cualquiera de nuestros amigos
Precondiciones	Tener una cuenta de usuario activa, tener una mascota registrada, estar autenticado como mascota, haber recibido una petición de amistad y ésta ser evaluada a aceptada.
Postcondiciones	No podremos crear nuevas publicaciones ni comentar las publicaciones creadas en el perfil de la mascota que hemos eliminado como amiga.
Flujo de eventos	1. Iniciar un sesión 2. Autenticarse como una de las mascotas registradas 3. Acceder al perfil de la mascota amiga 4. Pulsar sobre el botón eliminar amistad y confirmar
Usuario	Usuario registrado
Comentarios	Al eliminar una relación de amistad, se podrá generar de nuevo procediendo mediante el caso de uso con referencia 010 y siendo aceptada por el otro usuario

5. Detalles de implementación

En este apartado, veremos cómo se ha desarrollado la aplicación. Por una parte, se explicará porqué se usan ciertas tecnologías para la realización de este proyecto, además de las herramientas que serán esenciales para la realización del mismo, también se hará hincapié en el entorno de trabajo que se utilizará y se hará una descripción detallada de la configuración en caso de que sea necesaria.

5.1. Tecnología utilizada

Las tecnologías usadas en este proyecto, son PHP, HTML, CSS, MySQL y Bootstrap. PHP es un lenguaje de programación orientado al desarrollo web de contenidos dinámicos, el cual junto a HTML nos proporciona webs dinámicas. Por otra parte se incluyeron plantillas bootstrap con sus correspondientes hojas de estilo, las cuales proporcionan un aspecto homogéneo al sitio web, además de tener un diseño más limpio del que podría obtenerse, sin necesidad de tener experiencia en el uso de aplicaciones de diseño web, y por último la información queda registrada en la base de datos.

5.1.1. Entorno de Trabajo

En este apartado nombraré las aplicaciones que he usado para realizar este proyecto, las más importantes, haré una breve descripción justificando el motivo de haberlas usado, así como sus principales características que las han hecho optimas para su uso en este proyecto.

- Editor de textos **Notepad++**, esta aplicación es bastante útil dado que el sistema operativo Windows no cuenta con un buen editor de textos.

- Para realizar los diagramas necesarios he usado la aplicación online **Lucidchart**.

A continuación en los siguientes apartados se describirán las aplicaciones más importantes para la realización de este proyecto y se explicarán algunos de los pasos de configuración que fueron necesarios para tener el entorno local en condiciones.

5.1.1.1. Justinmind Prototyper

Para la realización de los bocetos se ha usado Justinmind Prototyper en su versión 6.3.1, el motivo de crear un prototipo de la interfaz es que el usuario final o cliente valide si la aplicación responde a sus necesidades, hacerse una idea de cómo quedará el producto final y buscar inconsistencias, antes de empezar con el desarrollo de la interfaz.

Algunas de las características por las cuales lo he elegido como programa de prototipo es su facilidad de uso, ya que permite crear un prototipo con muy poco tiempo, algo importante a la hora de trabajar en un proyecto. Otra de las características es el hecho de que se puedan crear interfaces para distintos dispositivos, como pueden ser tablets, smartphones, incluso interfaces para dispositivos como las google glass, con lo cual demuestra que es una aplicación muy actualizada al uso de las tecnologías actuales. También he de destacar el hecho de que ofrece una versión gratuita, la cual cubre las necesidades de este proyecto.

5.1.1.2. Xampp

Xammp es un entorno de desarrollo, el cual incluye Apache, Mysql, PHP y PerL. Esta herramienta es muy popular entre los programadores PHP debido a su fácil instalación, además en una sola aplicación dispones de todo lo necesario para poder poner en marcha de forma local tu servidor y hacer pruebas.

Para la realización de este desarrollo, se ha usado la versión PHP 5.6.3 y MySQL 5.6.21 incluida en la aplicación xampp.

Configuración envío correos con xampp

Para poder enviar correos desde nuestro entorno de trabajo local y poder realizar las pruebas deberemos configurar xampp para que envíe correos a los *emails* que proporcionan los usuarios durante la fase de registro.

Esta configuración no será necesaria en caso de que lo implementemos en un servidor de *hosting,* ya que ellos se encargan de proporcionarnos el servidor smtp ya instalado. A continuación se explicarán los pasos que se deben realizar.

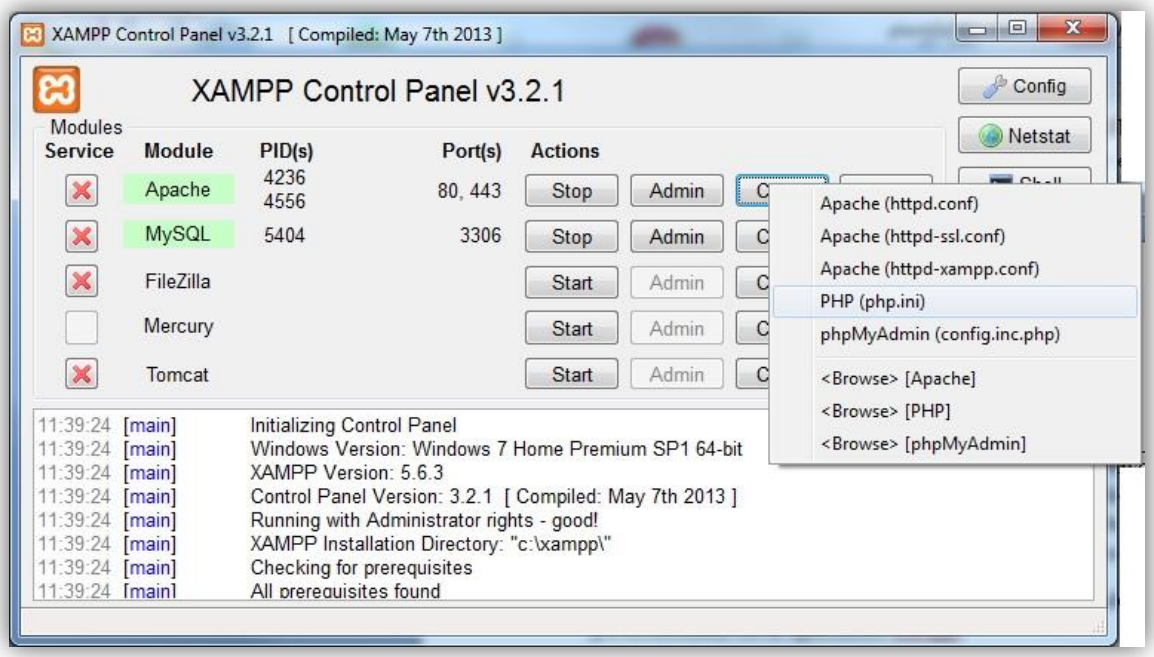

Figura 6 *XAMPP configuración*

El primer paso será abrir con un editor de textos el archivo php.ini, éste lo podremos encontrar fácilmente pulsando en el botón de configuración en el apartado apache de xampp, donde se desplegará un menú en el cual encontraremos los archivos de configuración de apache, tal y como se puede observar en la imagen anterior.

Una vez abierto el archivo php.ini buscaremos las líneas sendmail_path y descomentaremos la siguiente línea, para descomentar la línea tendremos que eliminar el símbolo ";" que precede a la línea.

sendmail_path = "\"C:\xampp\sendmail\sendmail.exe\" -t"

A continuación, deberemos comentar el otro sendmail_path que por defecto está descomentado en el fichero, la función de este "path" es la de enviar los emails al disco, algo que no necesitamos ya que queremos ver si el envío se realiza correctamente y cómo se visualiza, para ello comentaremos la línea dejándola de la siguiente manera.

;sendmail_path="C:\xampp\mailtodisk\mailtodisk.exe"

También podremos editar la dirección del email que realiza el envío, modificando el parámetro sendmail_from, en el cual pondremos la dirección que queramos. Finalmente el archivo debe quedar de la siguiente manera tal y como se muestra en la siguiente imagen.

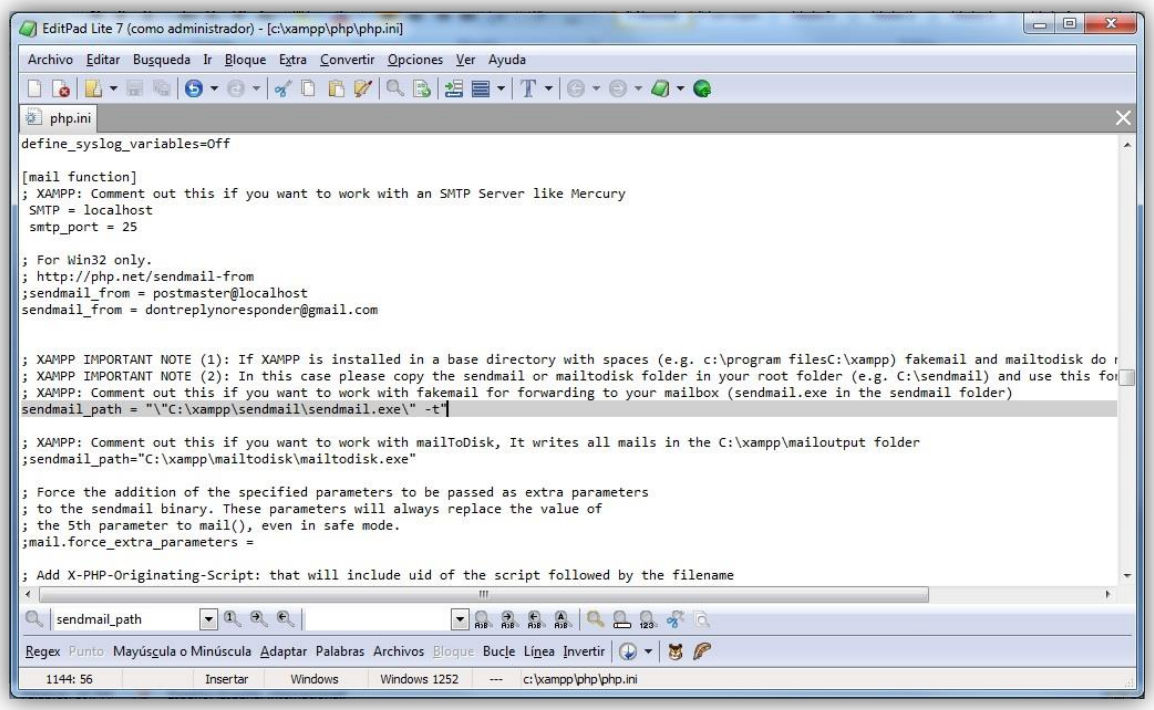

Figura 7 *Archivo de configuración php.ini*

Una vez realizado todos los pasos, guardamos el archivo, ya que no lo necesitaremos modificar más, y ahora procederemos a modificar el archivo sendmail.ini, para ello tendremos que ir al directorio "xampp/sendmail" donde se encuentra el archivo sendmail.ini, lo abriremos con un editor de texto y modificaremos las siguientes líneas.

La línea smtp_server la modificaremos estableciendo la url del servidor smtp de gmail, en mi caso he decidido usar gmail, pero podría usarse otro servidor.

smtp_server=smtp.gmail.com

En la línea smtp_port pondremos el puerto que usa gmail, en caso de que se quisiera usar otro servidor de correo este número de puerto podría ser diferente, en este caso la línea quedaría de la siguiente forma.

smtp_port=587

En auth_username pondremos nuestra cuenta de correo, la cual he creado previamente para este proyecto, en la línea auth_password escribiremos la contraseña de nuestra cuenta de gmail, en mi caso para que la contraseña no se viera he puesto asteriscos pero en realidad la contraseña no queda oculta.

auth_username=dontreplynoresponder@gmail.com

auth_password=****

Finalmente, guardaremos el archivo y reiniciaremos el servidor apache, pero esta vez lo haremos como administrador del sistema ya que de no hacerlo como administrador se bloqueará la aplicación y no se realizarán los envíos. Otra cosa a tener

en cuenta es si tenemos el firewall activo, ya que éste puede causarnos problemas al bloquear los puertos evitando el envío de emails, para ello estableceremos una regla en nuestro firewall para que no bloquee el puerto 587 ó desactivaremos nuestro firewall.

5.1.1.3. Netbeans

Después de haber buscado entre todos los *IDE's* disponibles, escogí Netbeans en su versión 8.0.2 ya que éste está especialmente orientado a proyectos web, también usé el *plugin* PHP el cual ayuda a escribir el código aportando sugerencias, lo cual agiliza la programación si no estás acostumbrado a programar en ese lenguaje.

Otro de los motivos de usar Netbeans es que es *open source* y puede descargarse en el siguiente link: https://netbeans.org/downloads/ . La instalación es muy sencilla, sólo hay que descargar el instalador correspondiente a nuestro sistema operativo y ejecutarlo.

Una vez instalado el *IDE* podremos instalar el *plugin* de PHP que nos asistirá en nuestra programación, para ello iremos al menú superior, pulsamos sobre la pestaña Tools, tal y como se muestra en la siguiente imagen, pulsamos sobre *plugins*.

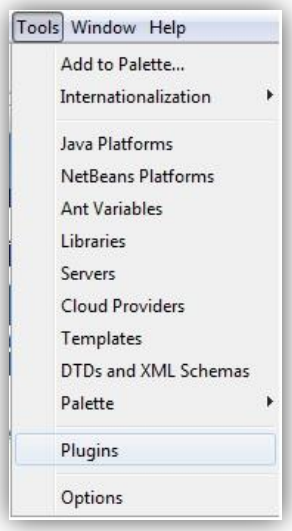

Figura 8 Captura de pantalla del menú *tools* de la aplicación netbeans

Una vez dentro de este apartado, podremos instalar los *plugins* que necesitemos para realizar nuestras tareas, en mi caso lo tuve que instalar usando la pestaña *AvailablePlugins*.

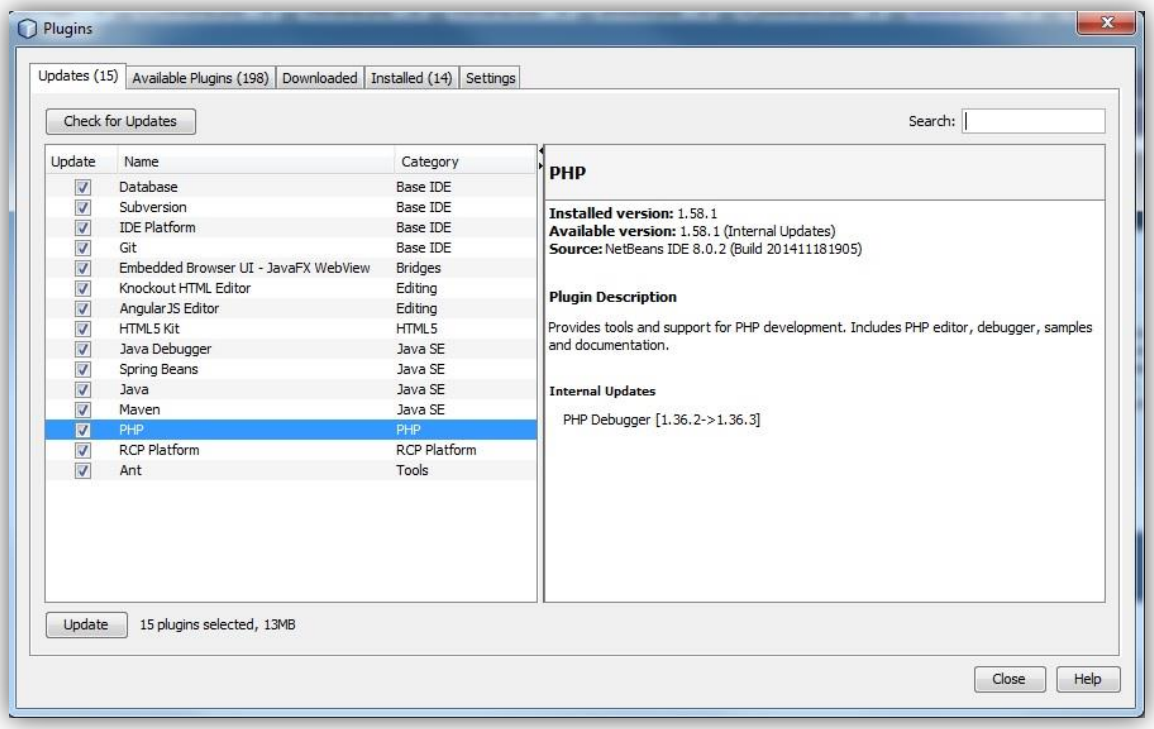

Figura 9 Captura netbeans apartado *plugins*

5.2. Diseño

La fase de diseño se compone de cuatro secciones, MVC en la cual explicaré en qué consiste esta arquitectura de software. La sección de persistencia mostrará las entidades con sus correspondientes campos. Lógica mostrará cómo se programará la aplicación y dónde se establecerán las reglas de la aplicación.

Y finalmente, la sección presentación, la cual, contendrá los diseños o *mockups* dentro de su apartado esbozos.

5.2.1. MVC

El Modelo Vista Controlador es un patrón de arquitectura de software que separa los datos, la lógica de negocio y la interfaz de usuario. La finalidad de este patrón es favorecer la reutilización del código, así como, facilitar su mantenimiento y la migración a otros modos de persistencia, ya que en el caso que se quisiera migrar a otro sistema de base de datos, sólo tendríamos que modificar el modelo.

Para utilizar esta arquitectura se propone la creación de tres componentes:

- El modelo: es la representación de la información, el cual gestiona los accesos a la base de datos, tanto si es para una consulta, actualización o eliminación de datos. Las peticiones de manipulación de información, se llevan a cabo a través del controlador, por otra parte, las peticiones de visualización, se envían a la vista.

- El controlador: es el encargado de la lógica de negocio, es decir, se encarga de gestionar los eventos que se producen en el sistema. Su función es la de un *middleware,* ya que este hace de intermediario entre la interfaz de usuario y el modelo. Esta separación, aporta un extra de seguridad, al no estar todo el código en un mismo archivo.

- La vista: es la forma mediante la cual la información es mostrada, es decir, es lo que el usuario final podrá ver. La vista se compone por la interfaz de usuario, la cual, recibe información desde el modelo para ser mostrada en un formato adecuado.

5.2.2. Persistencia

Tras el análisis que dio lugar al diagrama entidad relación, ahora vamos a definir los campos que compondrán nuestra BD, y explicaré por qué se escogieron ciertos campos así como su uso en esta red social.

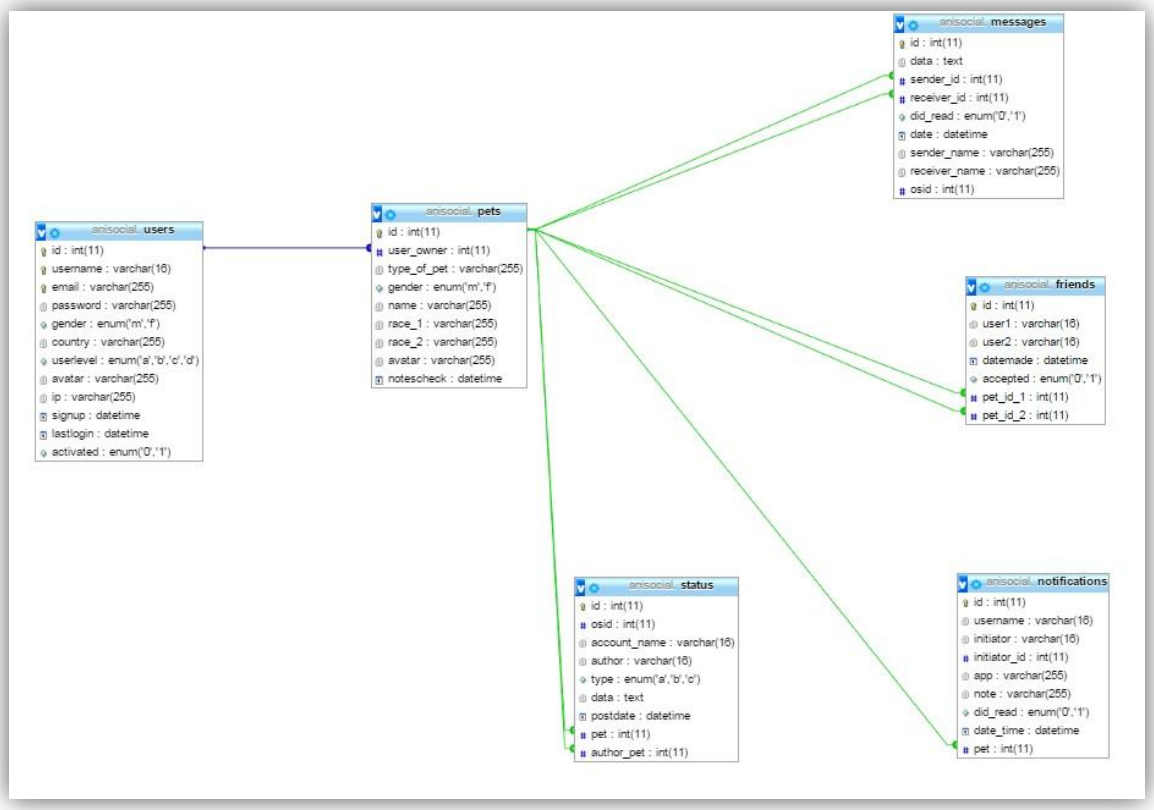

Figura 10 *Captura de phpmyadmin, tablas definitivas de la BD anisocial*

A continuación, se expondrán las funciones de cada campo y qué valores se almacenarán en el mismo.

Tabla: users		
Nombre	Tipo	Explicación
Id	Int	Dentro de este campo se almacena el identificador del usuario, el cual será único y se autoincrementará según se registren nuevos usuarios.
Username	Varchar	Este campo será elegido por el usuario al registrarse y será único. Durante la fase de registro, el usuario colocará un nombre de usuario, el cual deberá ser comprobado por el sistema antes de ser insertado, si éste ya existe en la BD se avisará al usuario al instante para que elija un nombre de usuario distinto.
Email	Varchar	El campo email se registrará en la base de datos durante la fase de registro. Este campo es necesario tanto para la autenticación del usuario en el sistema como para recibir el email de activación, el cual activará el usuario.
Password	Varchar	Este campo se almacenará codificado en la base de datos mediante la encriptación md5 y será usado para autenticarse en el sistema.
Gender	Enum	Para almacenar la información referente al sexo del usuario usaremos las opciones "m" para hombre y "f" para mujer, el motivo de usar solo una letra se debe a motivos de ahorro de espacio en el servidor.
Country	Varchar	Se almacena la información referente al país mediante los códigos ISO (ejemplo España = ES)
Userlevel	Enum	Este campo finalmente no lo usaremos, pero lo he dejado para que quede constancia de cómo podría ser usado este apartado para clasificar los usuarios según el rango que tuviesen, es decir podríamos clasificarlos como usuarios *premium* que se dedican a vender productos en la web o que tienen alguna funcionalidad extra como puede ser una protectora de animales. La forma de clasificarlos seria: 1. "a" usuario normal 2. "b" usuario tienda online 3. "c" protectora animales 4. "d" cuentas administrador
Avatar	Varchar	En este campo se almacenará el nombre de la imagen que se usará como imagen de perfil.
Ip	Varchar	La IP es quizás uno de los métodos más útiles para saber cosas sobre nuestros usuarios. Este campo es un *log* de el último sitio desde el cual se ha conectado un usuario, el cual puede ser de utilidad para *bannear* ip's en caso de ataque de denegación de servicio a nuestra web.
Signup	Datetime	Dentro de este campo se almacenará la fecha de registro del usuario, este apartado lo usaremos en las tareas de mantenimiento, ya que si comparamos la fecha de registro y la del apartado *Lastlogin* y ésta es muy prolongada en el tiempo, podremos optar por desactivar la cuenta o enviar un email de recordatorio al usuario para que vuelva a acceder
Lastlogin	Datetime	Aquí se almacenará la última fecha de acceso del usuario, la cual se actualiza cada vez que se accede.
Activated	Enum	Este campo sirve para activar el usuario, solo se permiten dos opciones 0 ó 1, 0 significa que el usuario

		no ha sido activado, 1 el usuario ha sido activado después de confirmar mediante el acceso del enlace que se ha enviado al email.

Tabla: pets		
Nombre	Tipo	Explicación
Id	Int	Este campo contiene el identificador de la mascota, el cual se pondrá de forma automática. En cada inserción de un nuevo animal dentro de esta carpeta el id se autoincrementa, además este campo será único y servirá para relacionar el animal con casi todas las restantes tablas.
User_owner	Int	En este campo se almacenará el identificador del propietario del animal, este es el id de la tabla users, el cual puede repetirse ya que esta tabla puede contener más de una mascota del usuario.
Type_of_pet	Varchar	Durante la fase de creación de la nueva mascota el usuario deberá elegir el tipo de animal, éste está limitado a los siguientes valores: mamífero, pájaro, pez, reptil, anfibio y artrópodo.
Gender	Enum	El género del animal sigue la mismas reglas que con el campo *gender* de la tabla *users*, se almacenará dentro de la BD de la siguiente forma, f significara sexo femenino y m sexo masculino.
Name	Varchar	Almacena el nombre de la mascota.
Race_1	Varchar	Este campo sirve para determinar la raza principal de la mascota.
Race_2	Varchar	Se almacena la segunda raza del animal.
Avatar	Varchar	En este campo se encontrará el nombre de la imagen de perfil de la mascota, la cual se localizará dentro de la carpeta users y en una subcarpeta del usuario.
Notescheck	Datetime	Almacena la fecha del último chequeo que se ha hecho a las notificaciones de la mascota.

Tabla: status		
Nombre	Tipo	Explicación
Id	Int	Este campo es único, se autoincrementa de forma automática según se vayan publicando estados en los muros de los perfiles de las mascotas.
Osid	Int	La finalidad de este campo es la de agrupar las respuestas de un estado, ya que las respuestas a un determinado estado se almacenarán con el mismo identificador, haciendo que de esta forma podamos agruparlos según el Osid.
Account_name	Varchar	La finalidad de este campo es almacenar el nombre del animal en cuyo muro se ha publicado, el motivo de crear este campo a pesar de tener su id en el campo Pet de esta misma tabla es por el hecho de no tener que hacer otra query y desperdiciar potencia de cálculo del servidor.
Author	Varchar	En el campo *author* se almacena quien es el nombre del responsable del estado o respuesta, la decisión de incluir el nombre del autor dentro de la BD es la misma que en el campo *Account_name*, ya que de esta forma simplificamos el acceso y ahorramos en potencia de cálculo.
Type	Enum	La función de este campo es la de clasificar las publicaciones, a continuación se mostrará el significado de cada opción. a: Si el creador del estado es el propietario del muro sobre el que lo publica. b: Lo usaremos para las respuestas a un estado c: Si el creador del estado no es el propietario del muro, es decir, es un estado creado por una mascota amiga.
Data	Text	Dentro de este campo se almacenará el texto del estado o de la respuesta que desde el perfil de la mascota se ha enviado.
Postdate	Datetime	Este campo es de vital importancia para los estados, ya que no solo sirve para indicar en qué fecha fue enviado cada uno de los estados y sus respuestas, sin este campo sería imposible establecer el orden de las respuestas de cada estado. Para ordenar las publicaciones, se ejecutará una consulta en la cual se ordene ascendentemente, quedando ordenadas las respuestas.
Pet	Int	Cuando se crea un nuevo estado, dentro de este campo se almacena de forma automática el identificador de la mascota, en cuyo muro se mostrará la publicación. Este campo se repetirá según el número de publicaciones que existan en el muro de la mascota.
Author_pet	Int	Dentro de este campo se almacenará identificador de la mascota que ha creado el estado o la respuesta.

Tabla: notifications		
Nombre	Tipo	Explicación
Id	Int	El identificador de la notificación se crea de forma automática, y se autoincrementa conforme se van creando las notificaciones.
Username	Varchar	Este campo lo creé con la finalidad de que el usuario al iniciar la aplicación, tuviese algún indicador de todas las notificaciones que han tenido sus mascotas, pero finalmente lo dejé sin usar. Creo que es conveniente dejarlo para que quede constancia y en caso de querer implementarlo, sería sencillo.
Initiator	Varchar	La finalidad de este campo es almacenar el nombre del iniciador de la notificación. Este campo podría perfectamente ser obtenido mediante el id que se almacena en el siguiente campo, pero para simplificar la lectura de los datos y evitar hacer uso de dos queries se almacena en esta tabla.
Initiator_id	Int	Guarda el identificador único de la mascota generadora de la notificación. El motivo de guardarla es por el hecho de poder generar un enlace en la notificación con el nombre de la mascota y que de esta forma, quien reciba la notificación, pueda acceder al muro de la misma.
App	Varchar	Este campo indicará el tipo de notificación si es una nueva publicación o si se ha comentado un post. La única finalidad es que al imprimir las notificaciones se identifique el tipo.
Note	Varchar	Dentro de este campo, se almacenará la notificación creada, la cual contendrá un enlace en el cual se podrá acceder a la notificación generada.
Did_read	Enum	La finalidad de este campo es la de indicar si la notificación ha sido leída, para ello se almacenará un 0 en caso de que sea una nueva notificación, en cambio una vez leída, ésta pasará a ser un 1.
Date_time	Datetime	En este campo se almacena la fecha de creación de la notificación. El motivo de almacenar este campo es la finalidad de poder ordenar las notificaciones según el orden descendente mediante la ejecución de una query.
Pet	int	Este campo guarda el identificador de la mascota sobre la cual va destinada esta notificación, cuando se implemente la parte lógica, la recuperación de los datos se realizará usando este campo.

Tabla: Friends		
Nombre	Tipo	Explicación
Id	Int	El identificador se crea de forma automática, y se autoincrementa conforme se van creando las relaciones de amistad.
User1	Varchar	Este campo contiene el nombre de usuario del propietario de la mascota, en este caso se puso por si en un futuro se quisiera remodelar la red social y hacer que los usuarios tuviesen amistades basándose en las amistades de las mascotas.
User2	Varchar	Contiene el nombre de usuario del propietario de la mascota a la que se le ha enviado la solicitud de amistad.
Datemade	Datetime	Registra la fecha en la que se realizó la petición de amistad.
Accepted	Enum	La aceptación de una petición de amistad es dual, es decir no solo se envía y ya son amigos, se tendrá que aceptar por ambas partes, y por ello este campo debe cumplir ciertas condiciones. Tras el envío de una petición de amistad el campo Accepted estará a 0, ya que aún no se habrá aceptado la petición de amistad. En este caso la petición de amistad está pendiente de evaluación, sin embargo si la petición es aceptada pasará a ser 1. En cambio si la petición es rechazada, se realizará la eliminación de la fila que contiene la petición.
Pet_id_1	Int	Contiene el identificador de la mascota que inició el proceso de envío de la petición de amistad. En la tabla se podrá encontrar tantas veces como peticiones haya enviado y no hayan sido canceladas.
Pet_id_2	Int	Dentro de este campo se almacena el identificador de la mascota a la cual se le ha enviado la petición de amistad, ésta puede aparecer tantas veces como el número de peticiones de amistad que haya recibido y no haya rechazado.

Tabla: messages		
Nombre	Tipo	Explicación
Id	Int	Contiene el identificador del mensaje, el cual será único y se autoincrementará automáticamente.
Data	Text	Aquí se guardará el texto del mensaje enviado.
Sender_id	Int	Contiene el identificador de la mascota que ha enviado el mensaje, este identificador puede repetirse dentro de la tabla mensajes, ya que un usuario puede enviar varios mensajes.
Receiver_id	Int	Este campo contendrá el identificador de la mascota a la que se le envía el mensaje.
Did_read	Enum	Según el valor que tenga este campo, indicará que el mensaje ha sido leído o no, en caso de no haber sido leído dentro de la base de datos, quedará registrado como 0. En cambio una vez haya sido visitado por el usuario, este apartado quedará registrado como 1. Este campo servirá para múltiples operaciones en la tabla de mensajes, ya que gracias a él podremos determinar el número de mensajes no leídos aún y mostrarlo en el menú, además de usarlo para resaltar los mensajes nuevos en el apartado de mensajes.
Date	Datetime	Registrará la fecha de envío del mensaje
Sender_name	Varchar	En este campo quedará registrado el nombre de la mascota que envió el mensaje, este campo podría eliminarse y hacer una query para recuperarlo, pero sería complicar el acceso a la información, con lo cual queda justificado el hecho de tenerlo dentro de esta tabla.
Receiver_name	Varchar	El campo receiver_name almacenará el nombre de la mascota que recibe el mensaje. Esta parte se almacena para una futura ampliación del proyecto por si se quiere mejorar el apartado de mensajes creando una especie de directorio de mensajes enviados.
Osid	Int	Este campo al final no se ha usado, pero si al ampliar el proyecto alguien quiere mejorar la gestión de los mensajes, podrá hacerlo gracias a este campo. La finalidad de este campo es agrupar los mensajes de una misma conversación.

En la versión final del proyecto se optó por prescindir de la tabla photos, ya que las fotos de perfil de usuario y las de las mascotas, finalmente se guardarán en el campo avatar. De todas formas, he creado la tabla para ver como quedaría en caso de que se quisiera hacer una galería de fotos, la cual se puede observar en la siguiente figura.

Figura 11 *Ejemplo tabla photos para galería*

5.2.3. Lógica

En este proyecto, la capa lógica queda definida tanto en archivos javascript como en archivos php, ya que las comprobaciones se llevan a cabo tanto en el cliente como en el servidor.

Esta capa de la aplicación es la encargada de validar si las acciones pueden ser llevadas a cabo, en esta aplicación los archivos de la capa de lógica se usará el sufijo _system o _data, en estos archivos se realizará la comprobación en el servidor.

5.2.4. Presentación

La presentación determinará el aspecto visual que tendrá la red social. Previamente a la realización del mismo, se realizará un esbozo o mockup en el que se plasmarán las ideas principales a realizar en este proyecto. Una vez realizado este esbozo, se presentará para su aprobación por el usuario final.

A continuación, expondré los resultados de la realización del esbozo así como expondré el uso de Bootstrap para la realización de este proyecto, y las ventajas de su uso.

5.2.4.1. Esbozo

Antes de la realización de cualquier interfaz, debe hacerse un boceto con las características que tendrá nuestra aplicación, la cual debe ser validada por el cliente. En este caso he decidido hacer un esbozo con la aplicación *Justinmind*, la cual ofrece fácilmente crear un boceto en el cual plasmar las ideas sobre cómo debería ser esta red social.

Previo a exponer la versión final de los bocetos, hubo algunos cambios, ya que el cliente, en este caso el profesor, me evaluó la interfaz encontrando algunas posibles modificaciones que resultaron ser interesantes y que mejoraban notablemente la usabilidad de la interfaz. A continuación, mostraré y compararé las interfaces modificadas que se obtuvieron después de la evaluación del tutor de este proyecto.

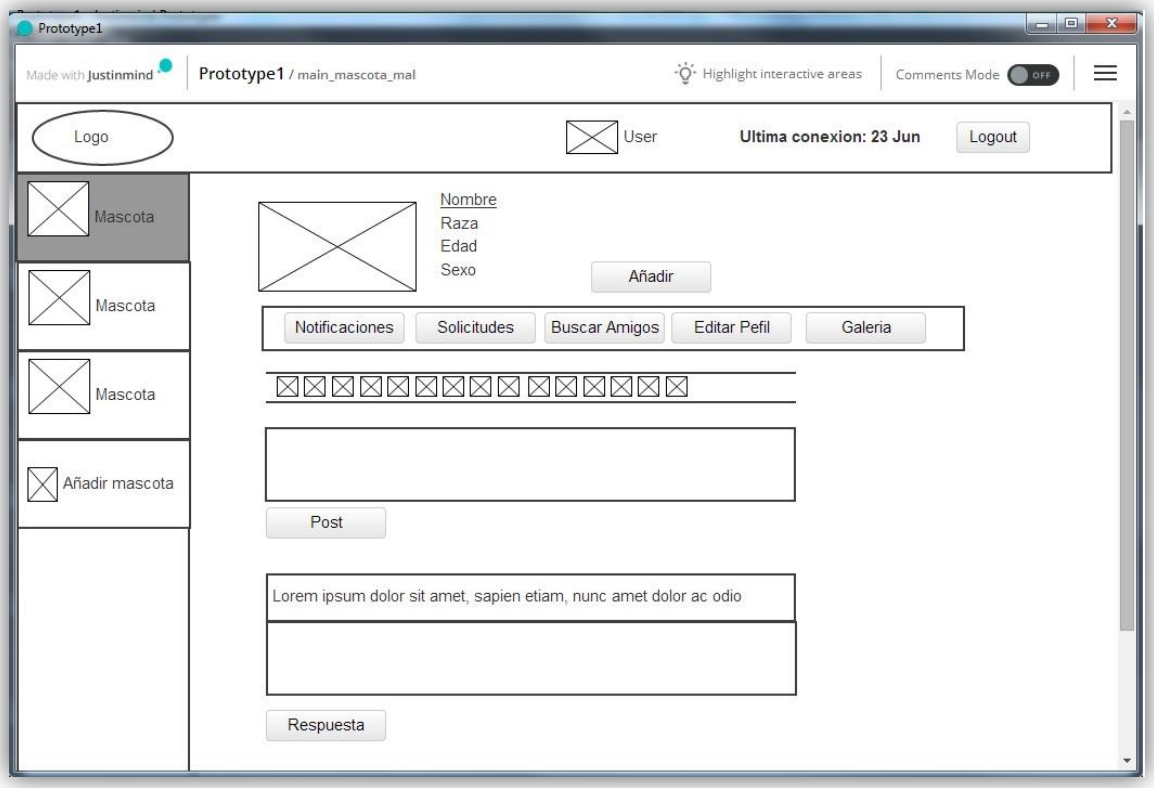

Figura 12 ***Mockup*** **del prototipo del muro de la mascota finalmente descartado**

Tal y como se muestra en la imagen anterior, la interfaz de usuario cuenta con una barra en horizontal, la cual contiene las funcionalidades del menú. Esta distribución no es la mejor, ya que la tendencia de las pantallas actuales es de crecer hacia los laterales pero no en altura, y por lo tanto la interfaz desperdicia la altura ocasionando que el usuario lo único que vea es la parte superior, ocultando las publicaciones y los envíos de las mismas.

Finalmente se optó por la siguiente distribución, ya que aprovechaba el ancho de la pantalla mucho mejor, además de poder ser trasladada a todas las partes de la red social, haciendo que el usuario tenga a mano las funcionalidades de la mascota y haciéndola visualmente más cómoda de usar.

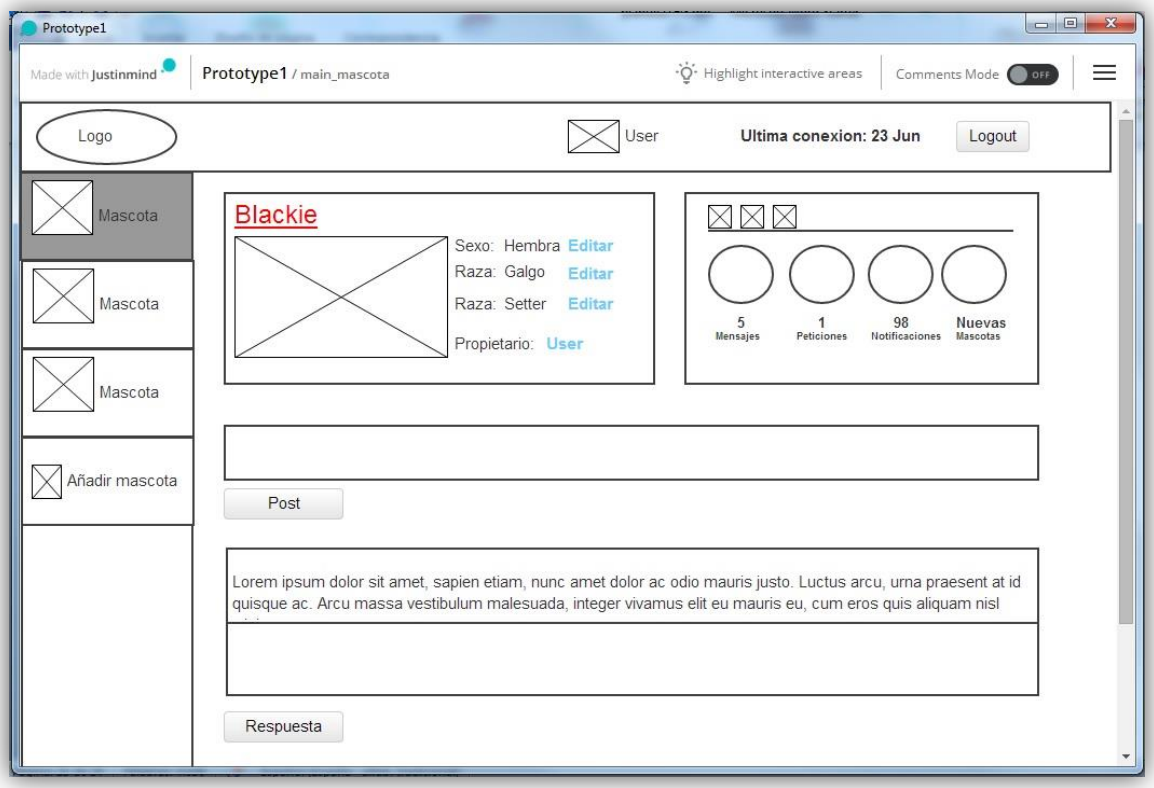

Figura 13 *Mockup* **del muro de la mascota definitivo**

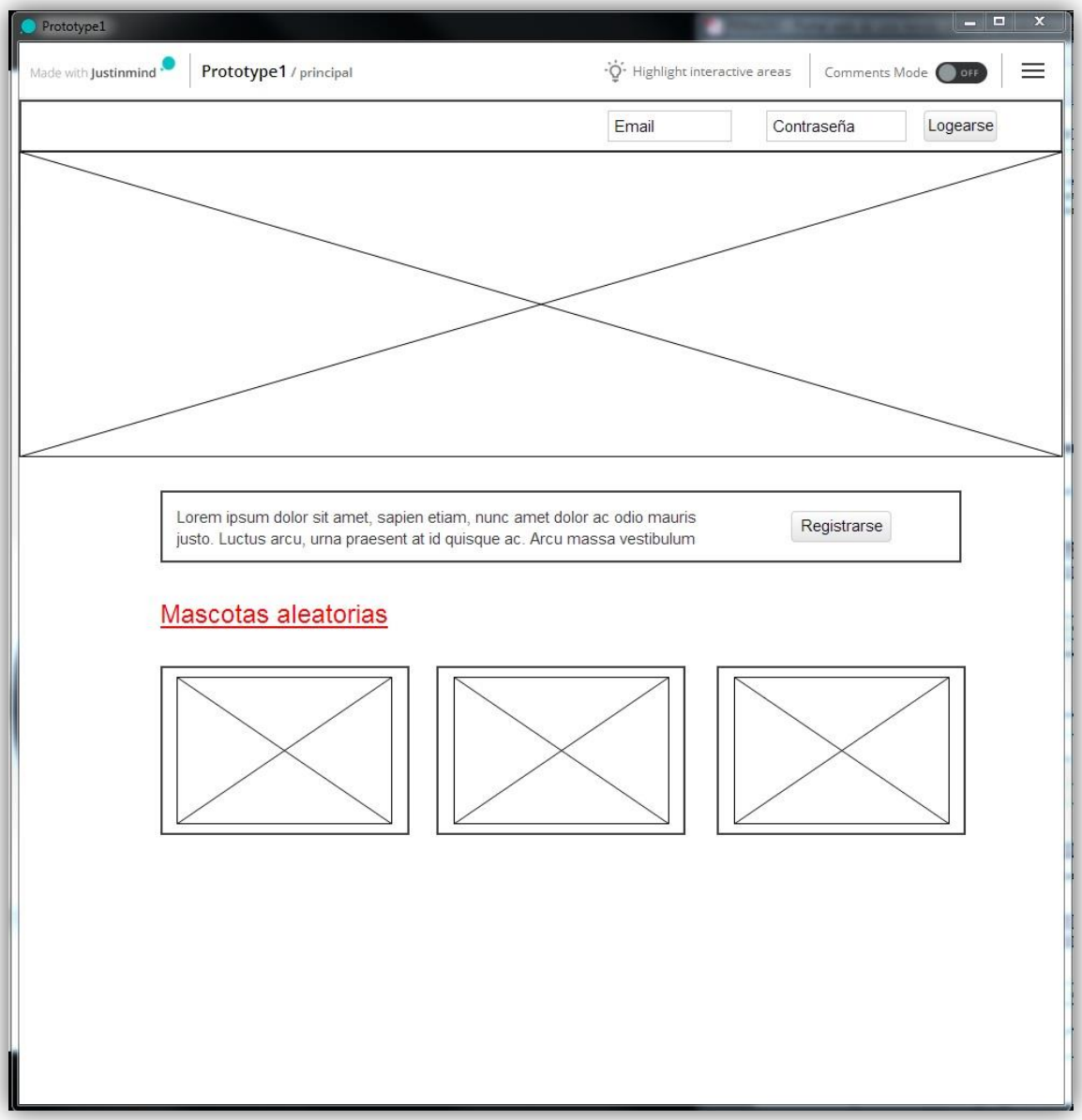

Figura 14 *Mockup* **de la portada**

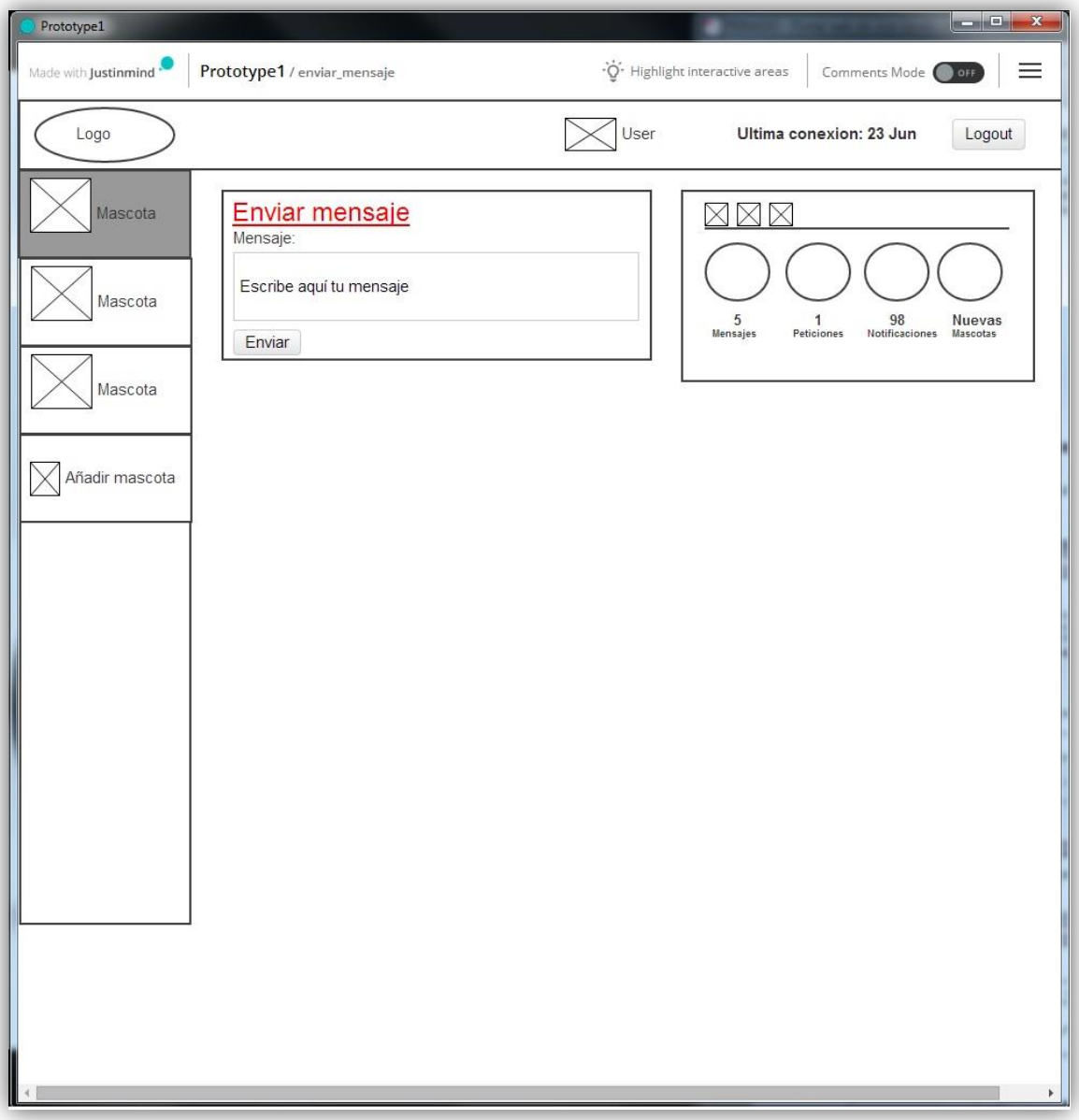

Figura 15 *Mockup* apartado de mensajes de la mascota

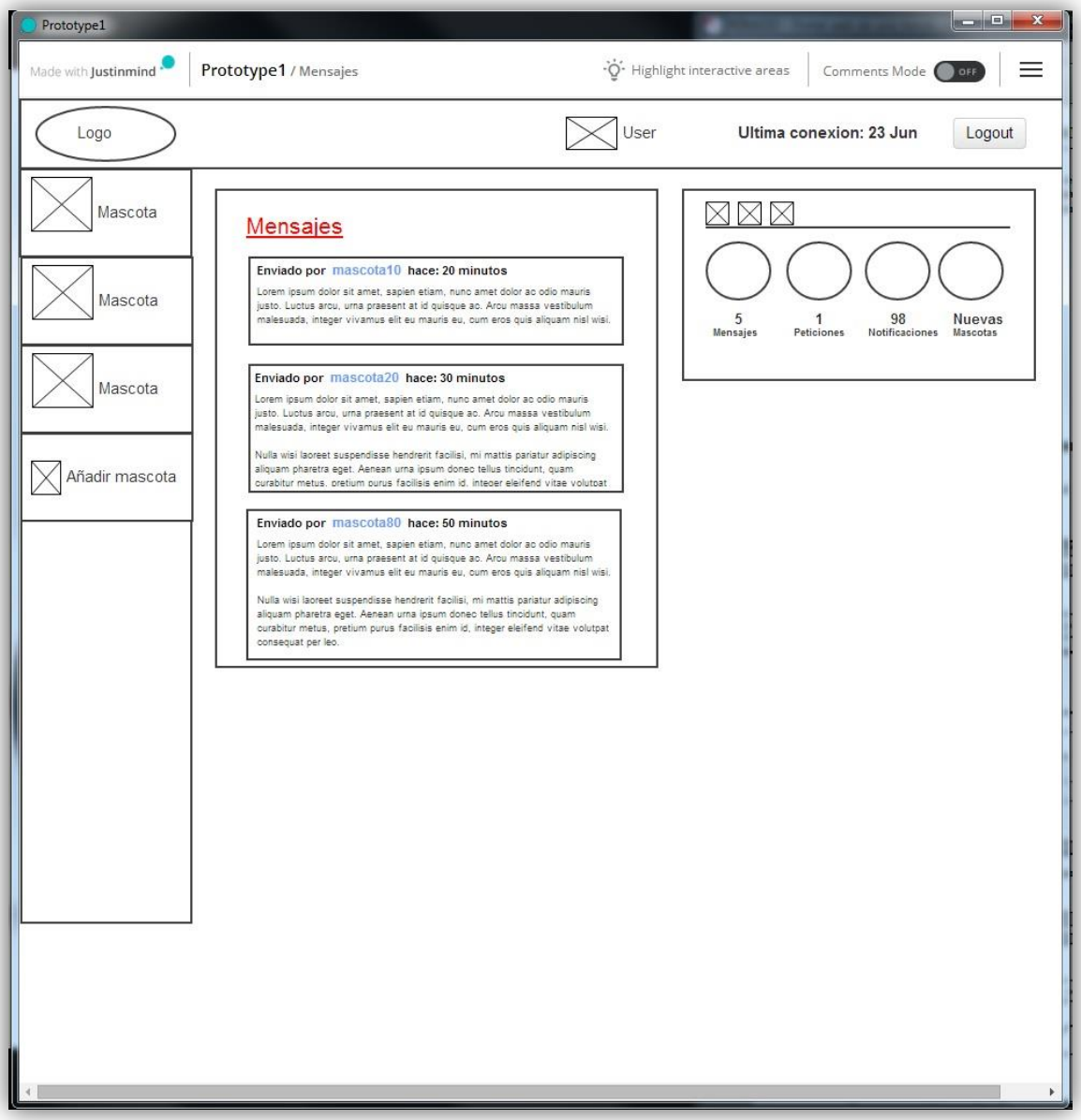

Figura 16 *Mockup* listado mensajes de la mascota

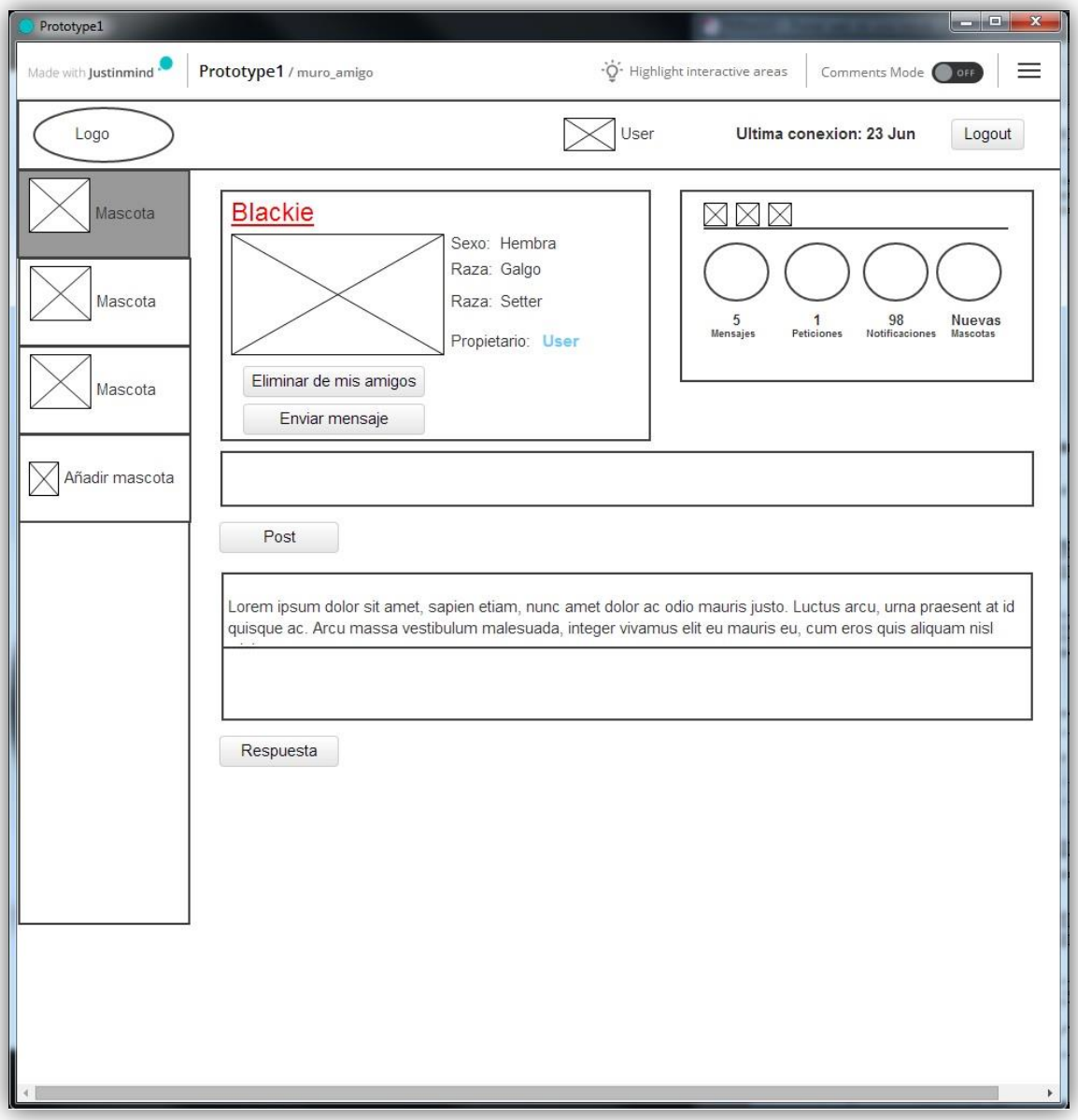

Figura 17 *Mockup* **del muro de la mascota visualizado desde un perfil de un amigo**

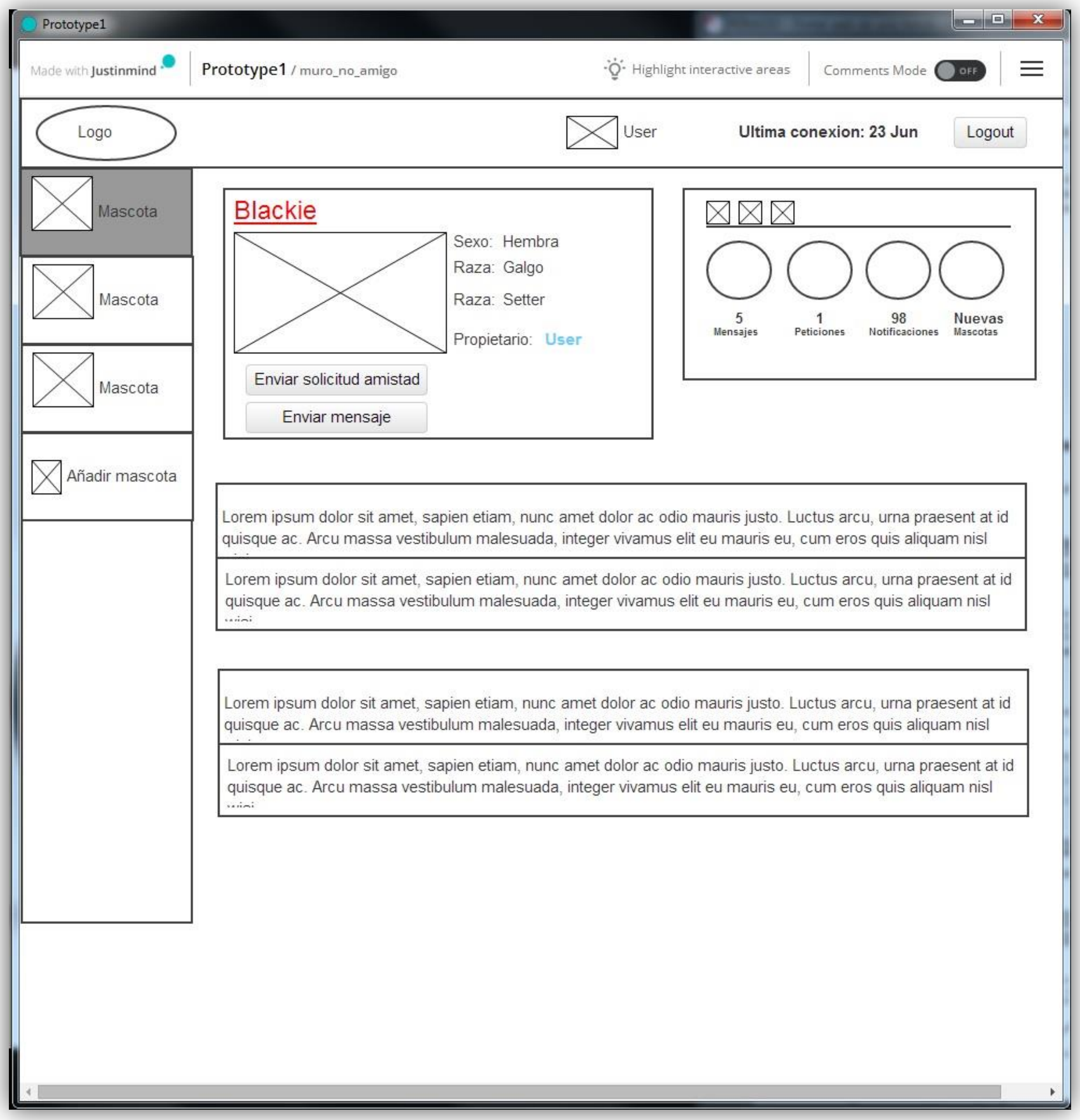

Figura 18 ***Mockup*** **del muro de la mascota visualizado desde el perfil de un desconocido**

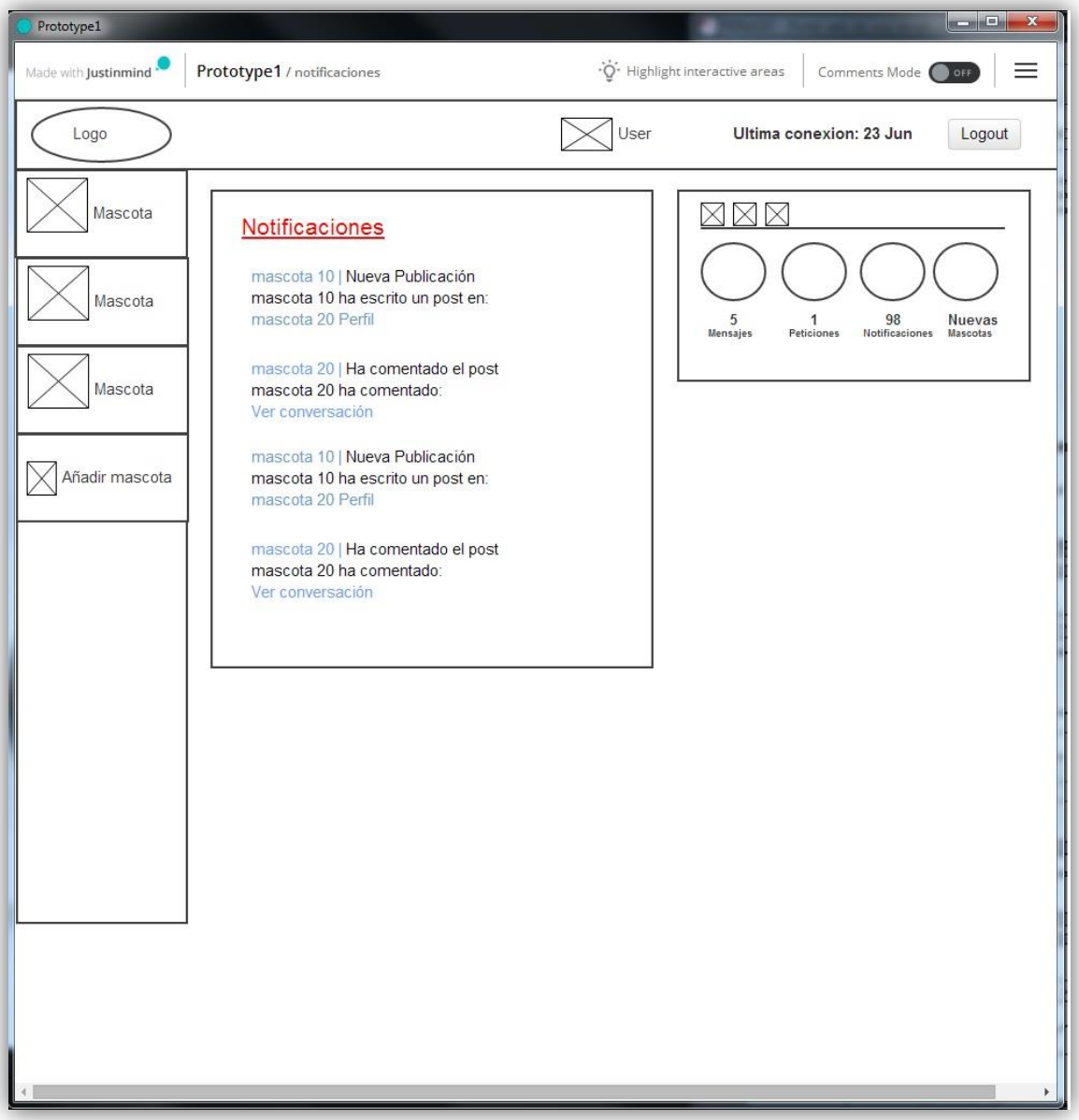

Figura 19 *Mockup* del apartado de notificaciones

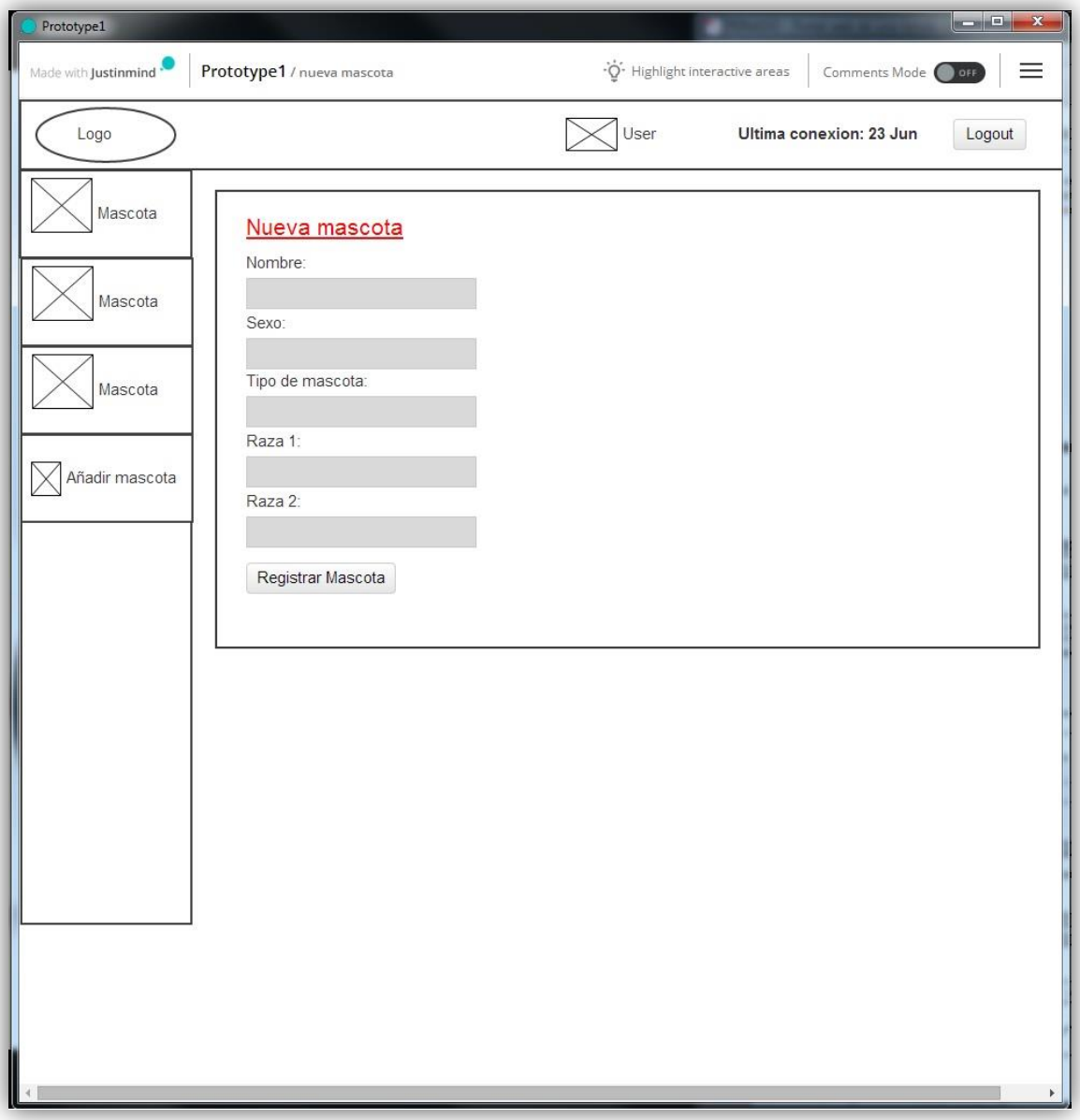

Figura 20 *Mockup* del apartado de registro de una nueva mascota

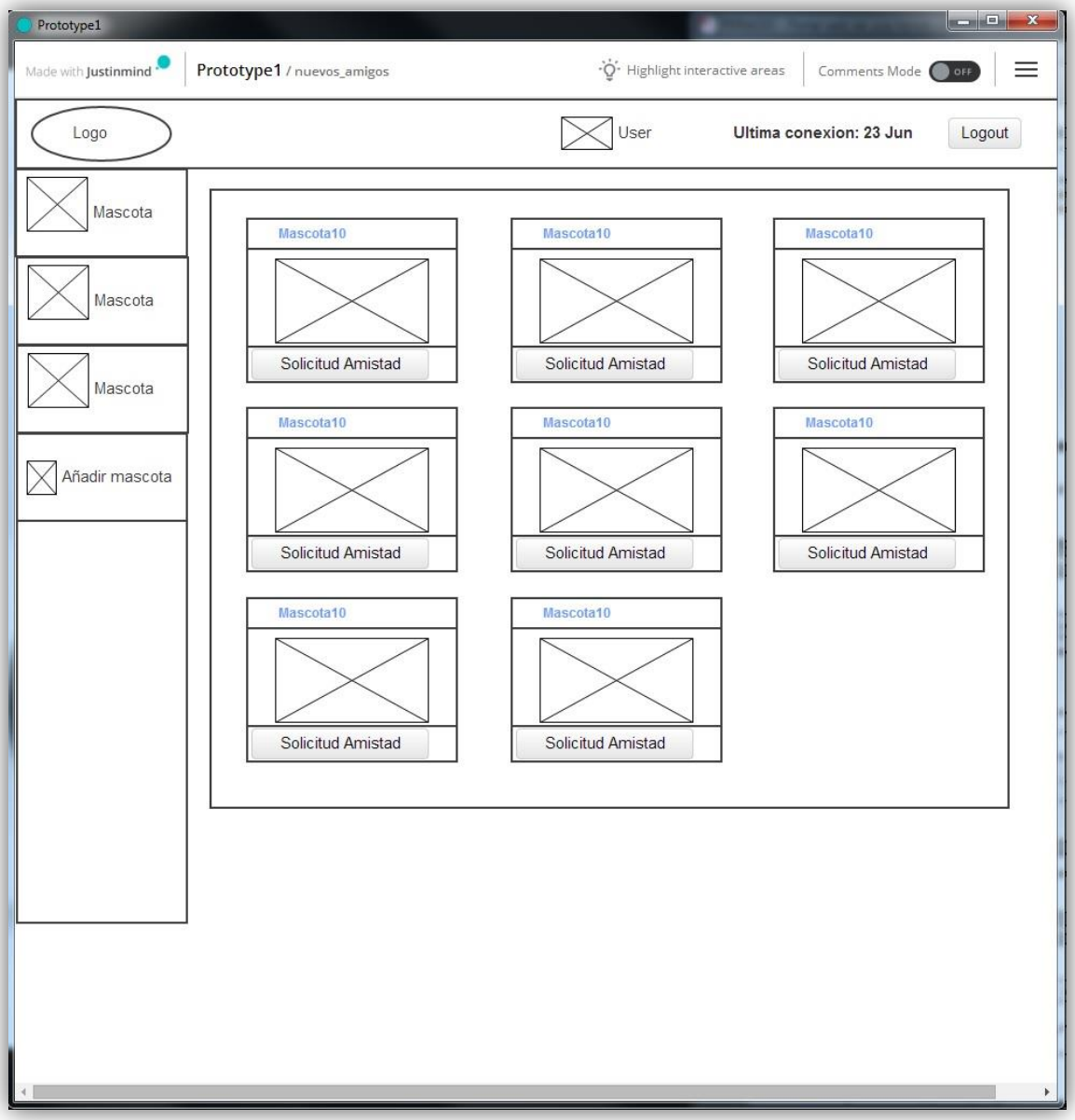

Figura 21 *Mockup* del apartado de búsqueda de nuevas amistades

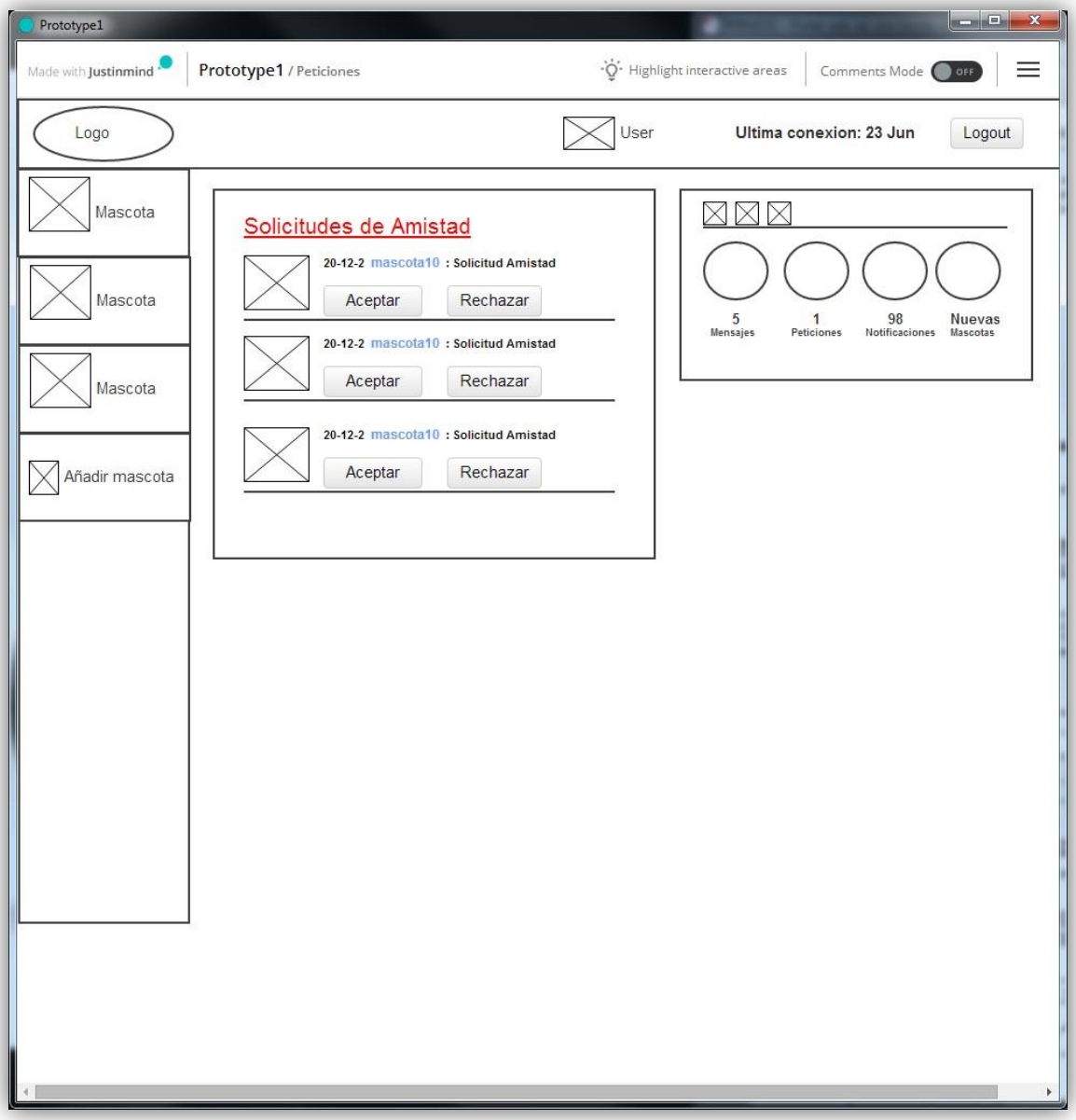

Figura 22 *Mockup* del apartado de solicitudes de amistad recibidas

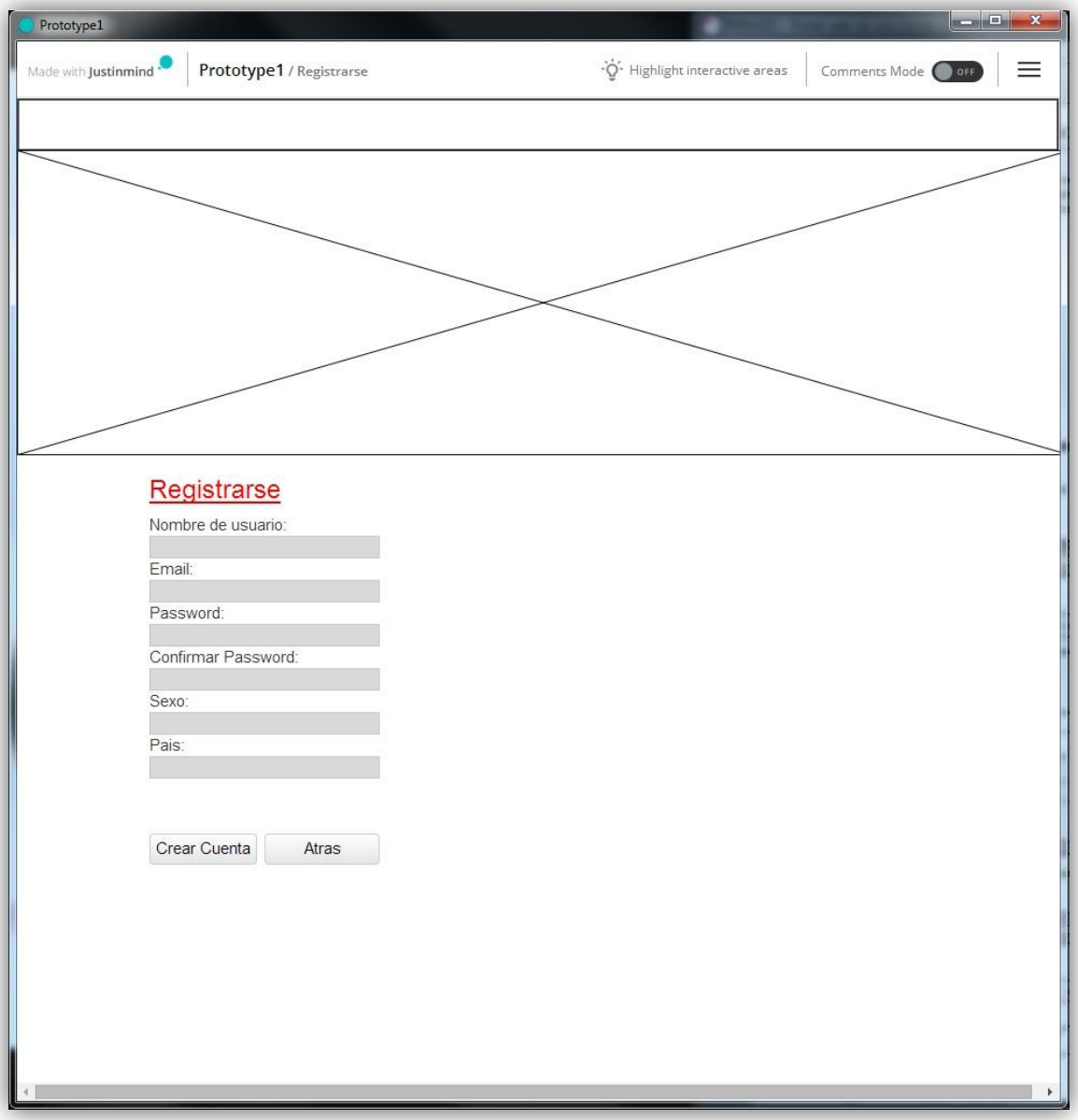

Figura 23 *Mockup* **del registro de nuevos usuarios**

5.2.4.2. Bootstrap

La decisión de usar bootstrap en este proyecto fue tomada por el hecho de querer emplear un diseño *responsive* y más apurado, ya que generalmente los *templates* de Bootstrap están diseñados por diseñadores gráficos y suelen ser bastante mejor de lo que puede obtenerse con los pocos conocimientos de diseño que puedan ser adquiridos en el espacio de tiempo de desarrollo de este proyecto. Aún así, se han incluido otras hojas de estilo propias dentro del proyecto para complementar algunas de las funciones del proyecto, especialmente en los estilos de post y mensajes.

Por una parte la portada principal se ha implementado mediante el uso de un Bootstrap de tipo carrusel, éste puede ser obtenido gratuitamente en la siguiente Url:http://ironsummitmedia.github.io/startbootstrap-modern-business/. La elección de este tipo de portada es para que sirva como base para futuras ampliaciones, como puede ser integrar un blog, ya que fácilmente se pueden añadir secciones sin tener que modificar el diseño.

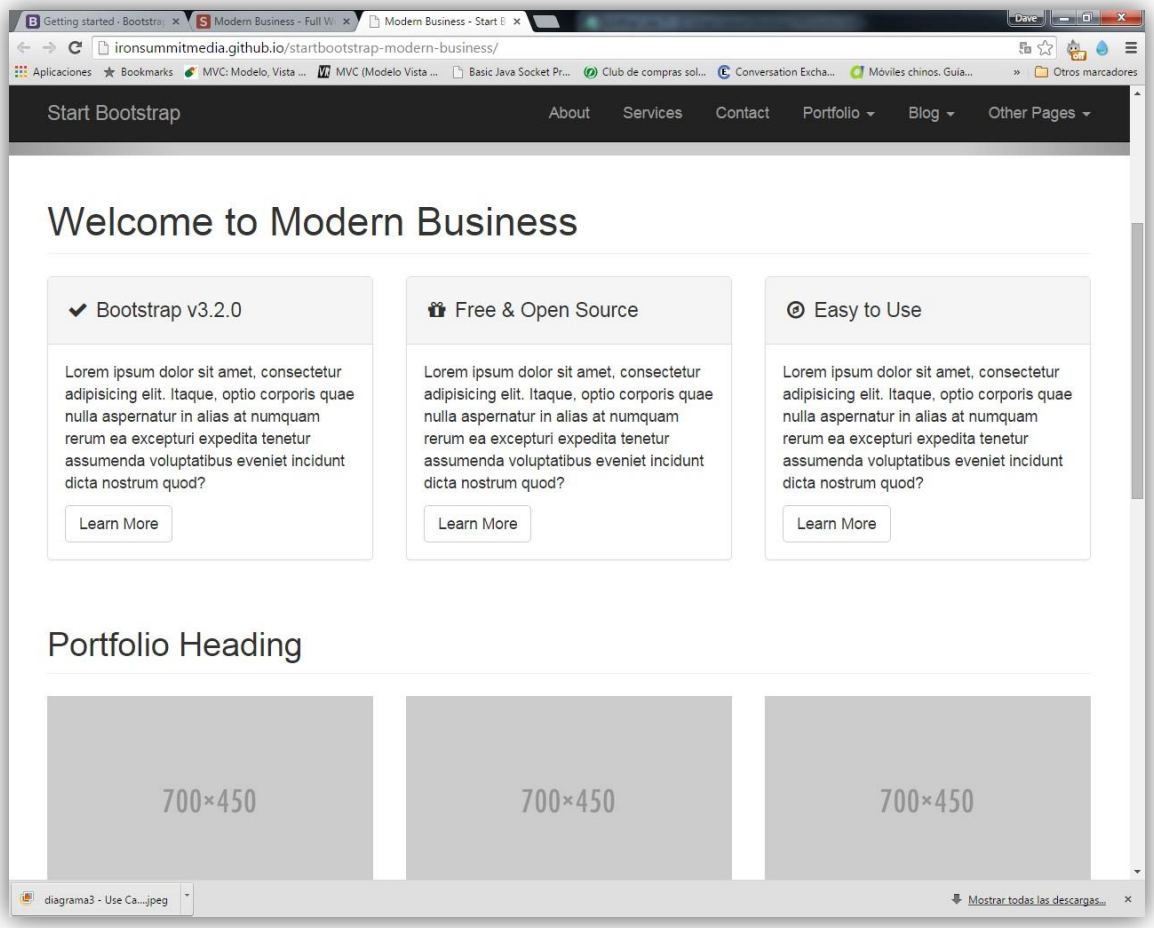

Figura 24 Captura de pantalla del Bootstrap de tipo carrusel "Modern Business"

Por otra parte se escogió el "BootstrapAdminTemplate - BinaryAdmin" como *template* para la interficie de usuario, ya que era una buena base para adaptar a este

proyecto, dada su estructura lateral que permitía alojar de forma cómoda a todas las mascotas del usuario sin que se tuviese que reestructurar la interfaz. Sin embargo, se tuvieron que adaptar los menús interiores de la interficie para que se adaptase a los requisitos del esbozo realizado. Este *template* puede obtenerse gratuitamente en la siguiente Url:http://binarycart.com/.

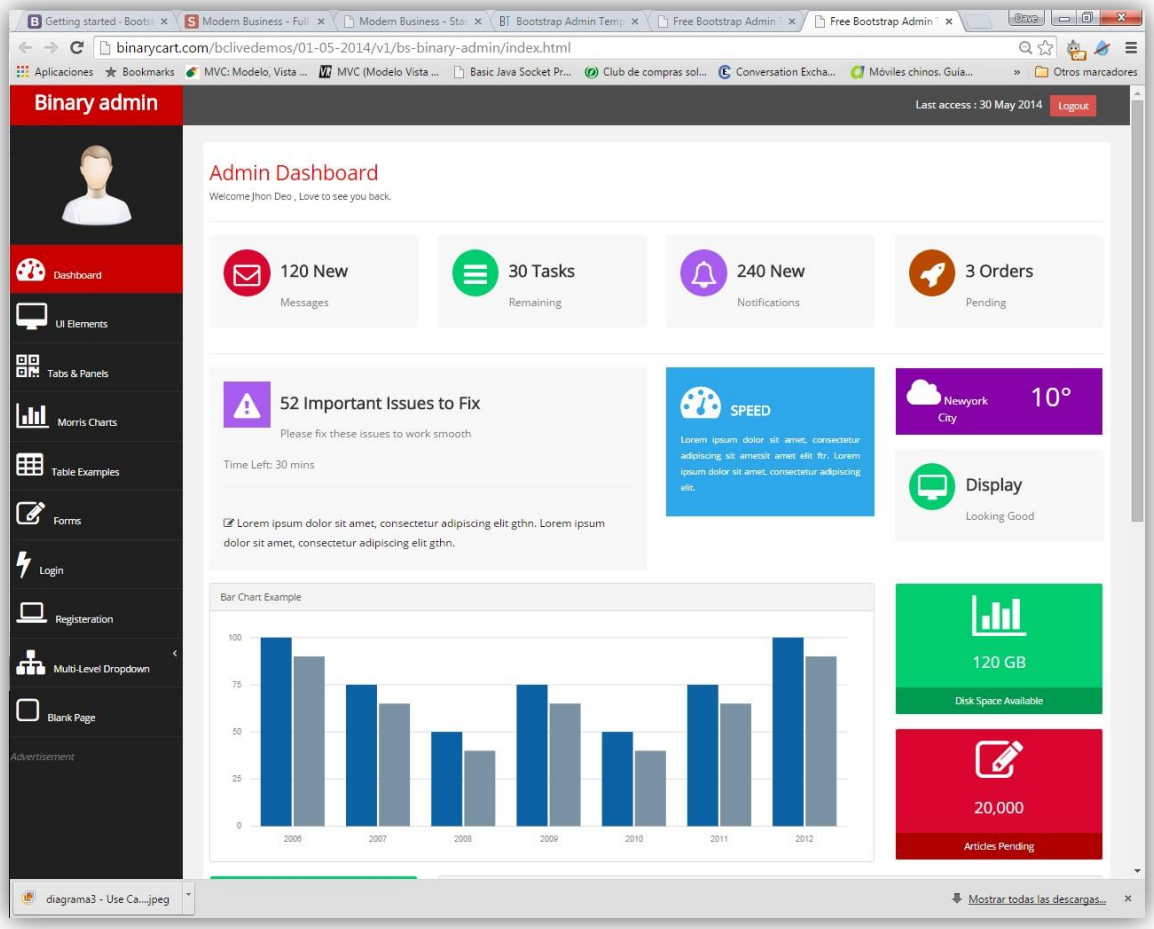

Figura 25 Captura plantilla Bootstrap Binary Admin

5.3. Estructura de los ficheros y directorios

En este apartado, se describirá cómo está estructurado el proyecto, con la finalidad de servir de guía para alguna posible ampliación.

Assets

La carpeta Assets contiene todo lo relacionado con el comportamiento visual de la web, es decir los estilos de la web.

Dentro de esta carpeta existen los siguientes directorios, los cuales son los estilos de Bootstrap usados para la realización de este proyecto:

- Carousel: En este directorio se almacenan las hojas de estilo, las fuentes, las imágenes y los archivos Javascript, encargados de la visualización de la portada de este proyecto. Dentro de este directorio se encuentran otros 4 directorios.
  - La carpeta Css se almacenan las hojas de estilo de la portada
  - Font-awesome, las fuentes usadas en la portada
  - Img, las imágenes usadas en la portada
  - Js, los archivos Javascript usados en la portada
- Css: Este directorio contiene todos los estilos Bootstrap usados en toda la interfaz de usuario.
- Font-awesome: Dentro de este directorio se encuentran las fuentes usadas en la interfaz de usuario.
- Fonts: Contiene las fuentes de la interfaz de usuario.
- Img: Este espacio está dedicado a almacenar las posibles imágenes prediseñadas de la interfaz.
- Js: Contiene los archivos Javascript de la interfaz

Dentro de la carpeta se encuentran varias carpetas que describiré a continuación, la primera carpeta que describiré será Carrousel, esta carpeta contiene los estilos propios de la portada de la web, dentro de esta carpeta hay 4 directorios más.

Cronjobs

Los *Cronjobs* son tareas de mantenimiento automatizadas, las cuales se realizan durante ciertos periodos de tiempo, debiendo ser programadas para ser ejecutadas al implementar el servidor. Para ello contaremos con archivos php en los cuales se realizarán sobretodo tareas de eliminación o actualización de la BD. Estas tareas están destinadas principalmente para evitar que la BD se sobresature y liberar espacio, pero ésta no es su única funcionalidad. También puede ser usado como timer para ciertas acciones, es decir si una acción queda pendiente de realizarse al pasar cierto tiempo ésta podría ser ejecutada.

En este proyecto tenemos un directorio donde se encuentran los *cronjobs* programados para esta red social. En el interior de nuestro cronjob se encuentra el archivo once_daily.php programado para la eliminación de los usuarios inactivos, la cual al implementarla en el servidor de explotación haré que se ejecute cada cierto tiempo, y de forma totalmente automatizada. Esto nos liberará de esta tarea de mantenimiento.

Images

Dentro de esta carpeta se encuentran las imágenes prediseñadas de la aplicación, por ejemplo las imágenes de perfil de la mascota por defecto.

Js

Esta carpeta contiene los archivos Javascript de la aplicación.

Php_includes

El objetivo de esta carpeta es almacenar los archivos que se incluyen numerosas veces en el proyecto. Entre los archivos que se encuentran dentro de esta carpeta, está el archivo db_conx.php, en el cual se encuentra la configuración necesaria para la conexión. En caso de querer cambiar el nombre, la dirección IP del servidor, o el usuario y password, deberemos editar este archivo. En la fase de implementación sobre el servidor definitivo, se editará este archivo para adaptarlo al nuevo servidor.

Otros archivos no menos importantes son los encargados de comprobar que el usuario y la mascota están autenticados correctamente.

Php_parsers

En este directorio se encuentran los sistemas encargados de la parte lógica de la aplicación, entre los cuales están los sistemas encargados de las relaciones de amistad, publicaciones, actualización de datos y la administración de las imágenes de perfil.

Style

Dentro de esta carpeta encontraremos los estilos propios creados por mí, el motivo de ponerlos separados de los propios del Bootstrap es para poder editarlos sin tener que modificar los propios del diseño.

User

Este directorio es usado específicamente para almacenar los archivos de cada usuario. En este caso dentro de cada carpeta de usuario se encontrarán las imágenes de perfil del usuario, así como las imágenes de perfil de cada una de las mascotas del mismo.

Source_files

Las vistas se encontrarán en el directorio principal de la aplicación. Este directorio contendrá los archivos encargados de realizar la correcta visualización de la interfaz.

6. Pruebas

Ahora realizaremos las correspondientes pruebas de la aplicación para comprobar si su comportamiento es el adecuado. Para ello se usarán distintos exploradores para comprobar si hubiese algún comportamiento extraño. También se usarán distintos dispositivos como son un *smartphone* o una *tablet*. Gracias a Bootstrap, la versión móvil se verá correctamente ya que el *template* usado tiene hojas de estilo *responsive*.

6.1. Validación

Para realizar la validación se harán las pruebas del W3C (World Wide Consortium), W3C es una organización que rige los estándares y recomendaciones para el desarrollo de webs. Si pasamos la prueba podremos mostrar el icono del w3c en nuestra web, con lo cual los usuarios verán que el webmaster se ha preocupado por validar su web. Esto no quiere decir que la red social se visualice correctamente, en ocasiones incluso habiendo pasado las pruebas, algunos exploradores pueden interpretar el código de forma distinta.

La validación que realiza el W3C es revisar el código HTML o XHTML en busca de errores de sintaxis, para posteriormente devolver el resultado de la validación. Algunos de estos errores no son apreciables en algunos exploradores, ya que estos interpretan las líneas de código incluso en ocasiones cuando algunas de ellas son erróneas.

A continuación se muestran los resultados de evaluar nuestro proyecto. En primer lugar procedimos a evaluar la portada, la cual se encuentra dentro del archivo index.php.

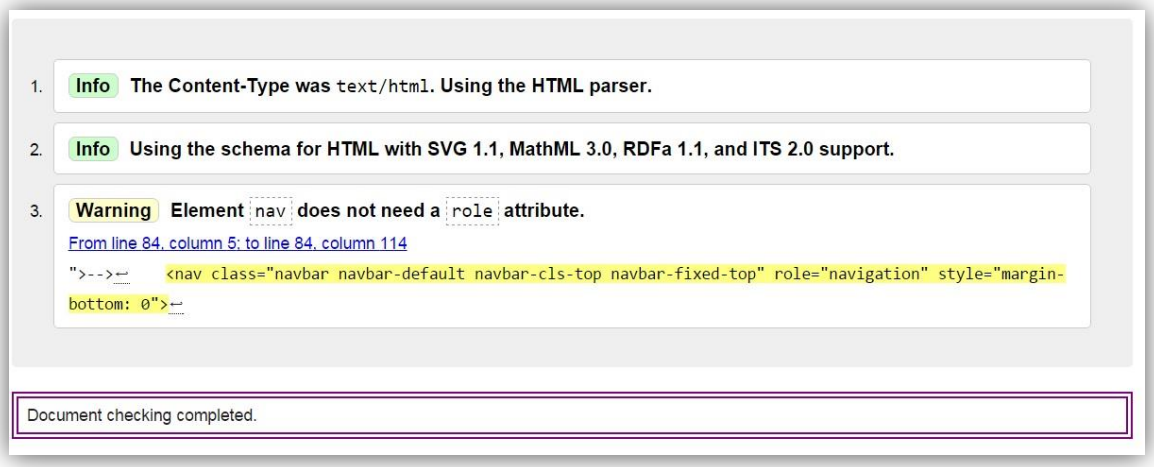

Figura 26 Validación de la portada mediante W3C

El resultado de evaluar la portada nos dio sólo un warning, el cual nos indica que el marcador “nav” no necesita el atributo role, lo eliminamos y volvemos a comprobar la web, esta vez ya no existen warnings.

Comprobar la portada fue bastante fácil, ya que al ser una página bastante estática existen pocos errores, en cambio la parte de la interfaz de la aplicación presentaba numerosos errores, por ejemplo en la parte de la interfaz del usuario se llegaron a encontrar cerca de 13 fallos. Esto nos da una clara imagen del por qué se debe evaluar un pagina dinámica, además de quedar claro que una página web dinámica puede esconder muchos errores que solo serán perceptibles por este tipo de herramienta, ya que el explorador interpreta la mayor parte de las veces qué es lo que se quiso escribir en el código por parte del programador.

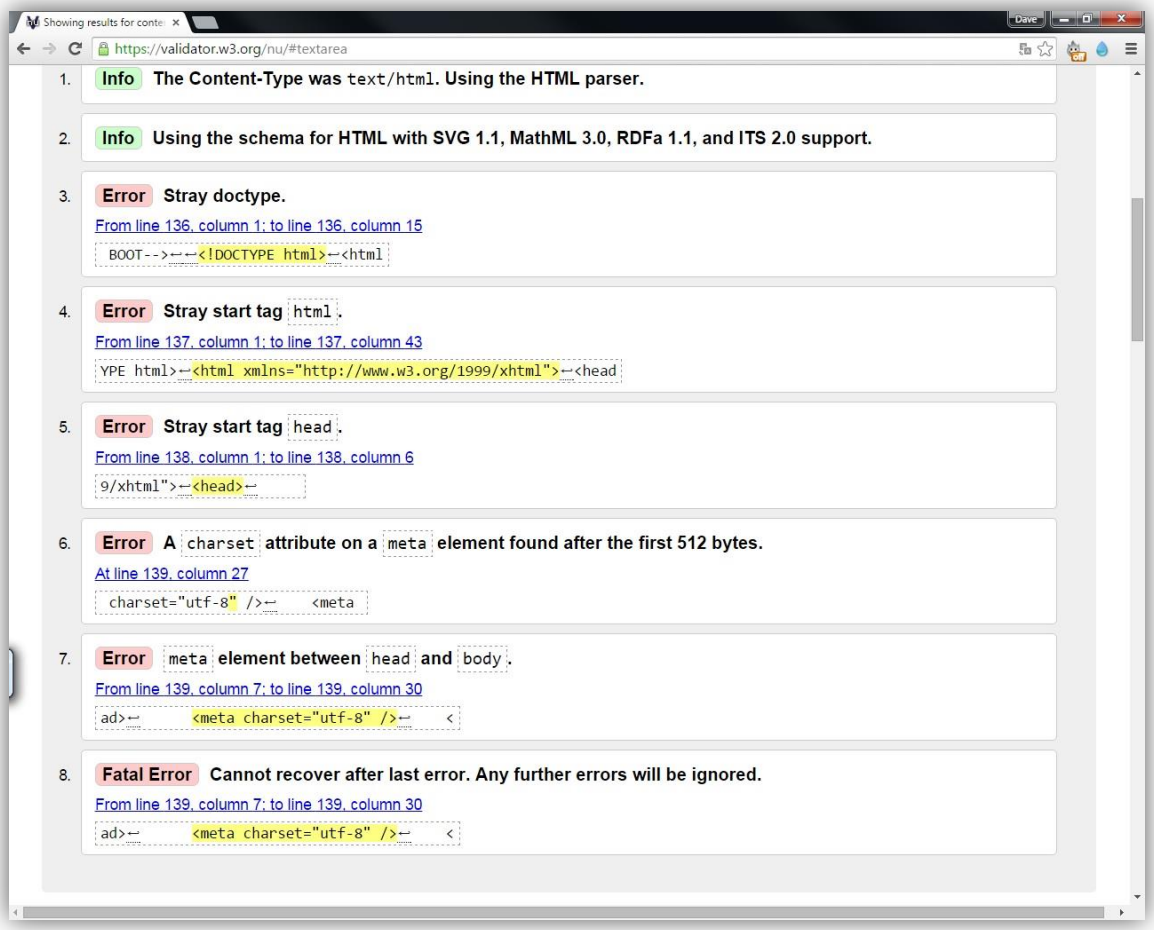

Figura 27 *Validación de la interfaz usuario W3C*

El resto de interfaces siguieron la misma dinámica y se corrigieron una a una usando el validador, una tarea bastante ardua debido a la cantidad de páginas que componen la red social.

He de destacar que existen otros validadores a parte del W3C como son Valet y WDG, pero el motivo de escoger W3C es por su popularidad y su renombre en lo que concierne a desarrollo de webs.

Otros validadores disponibles son:

VELET: http://valet.webthing.com/page/

WDG: http://www.htmlhelp.com/tools/validator/

Previamente a realizar las pruebas de resolución y navegadores, haremos un pequeño estudio para determinar en qué resoluciones y navegadores debemos hacer las pruebas para que sean lo más efectivas posibles.

El primer análisis será sobre cuáles son los navegadores web más usados durante el mes de Julio del 2015. Esta estadística ha sido realizada por W3Counter, usando la información proporcionada por las 33.916 visitas registradas. Esta estadística puede ser visitada en el siguiente enlace: http://www.w3counter.com/globalstats.php?year=2015&month=7 .

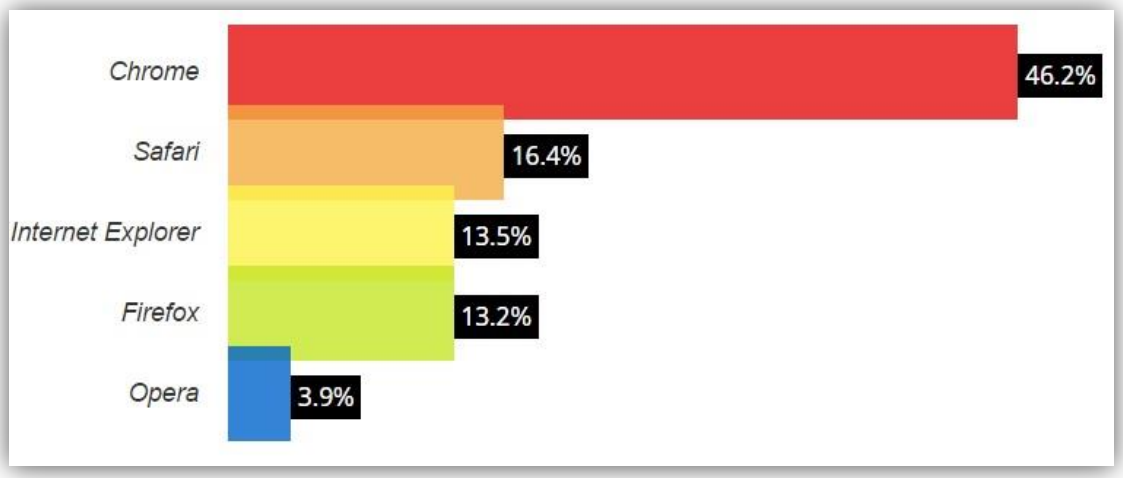

Figura 28 *Estadísticas de uso de exploradores a nivel mundial, proporcionada por W3Counter en el mes de julio 2015*

Otra cosa y no menos importante a tener en cuenta, es la resolución de pantalla usada por los usuarios y de esta forma poder hacer las pruebas adaptadas a este tamaño, así como durante la programación de la web hacerla especialmente para las resoluciones más usadas. En mi caso durante la programación usé una pantalla con resolución 1280x1024.

A continuación, mostramos una tabla con las estadísticas de las resoluciones más usadas durante el mes de Julio de 2015, esta estadística se ha obtenida del W3Counter.

Resoluciónes de pantalla (Julio 2015)		
1	1366x768	18.89%
2	1920x1080	7.69%
3	360x640	7.07%
4	1024x768	5.91%
5	768x1024	5.25%
6	1280x800	4.86%
7	1600x900	4.28%
8	1280x1024	4.27%
9	320x568	4.25%
10	1440x900	4.03%
11	Otras	33.5%

Figura 29 Resoluciones de pantalla más usadas en Julio 2015

Chrome resolución1366x768

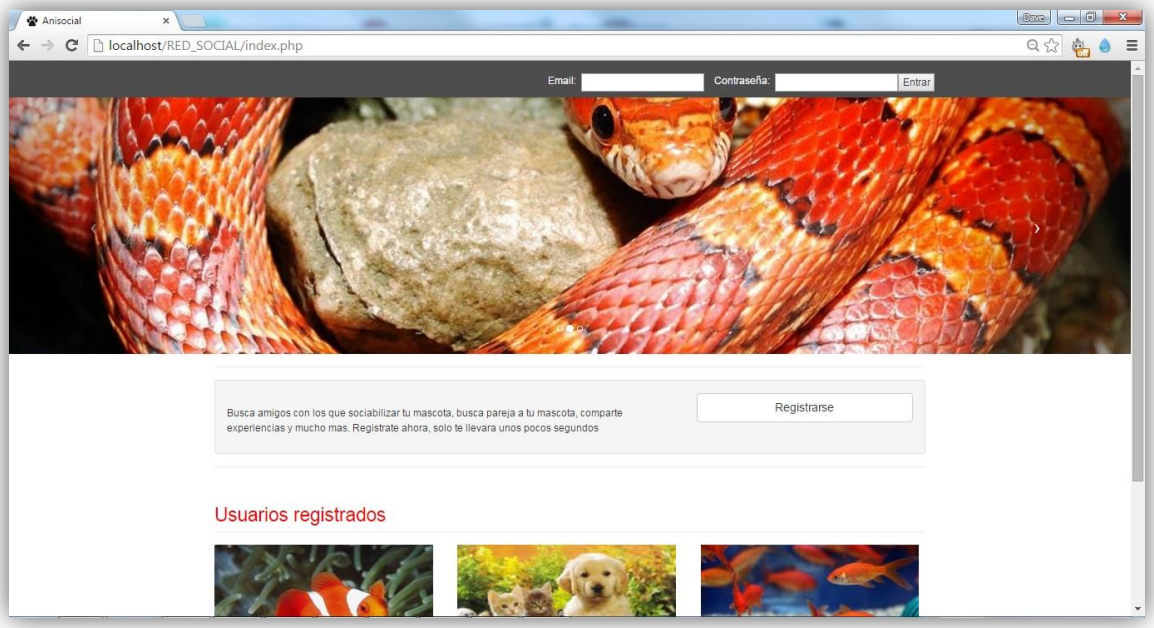

Figura 30 Captura de pantalla de la portada en Chrome, resolución 1366x768

Empezaremos con la resolución 1366x768, debido a que es una de las más comunes según la estadística de W3Counter. Tal y como se puede observar, usando esta resolución se tiene que hacer un pequeño scroll vertical para que el usuario pueda ver las imágenes del apartado de usuarios registrados, algo bastante común en cualquier aplicación web y que no supone ningún problema.

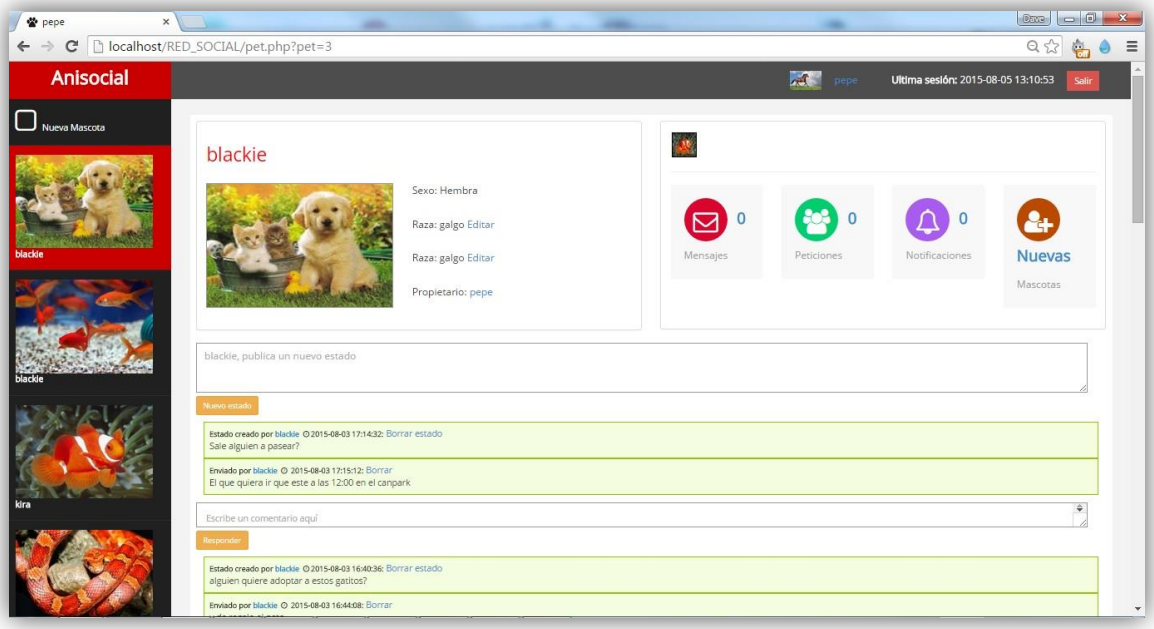

Figura 31 Captura de pantalla del muro en Chrome, resolución 1366x768

Chrome resolución1920x1080

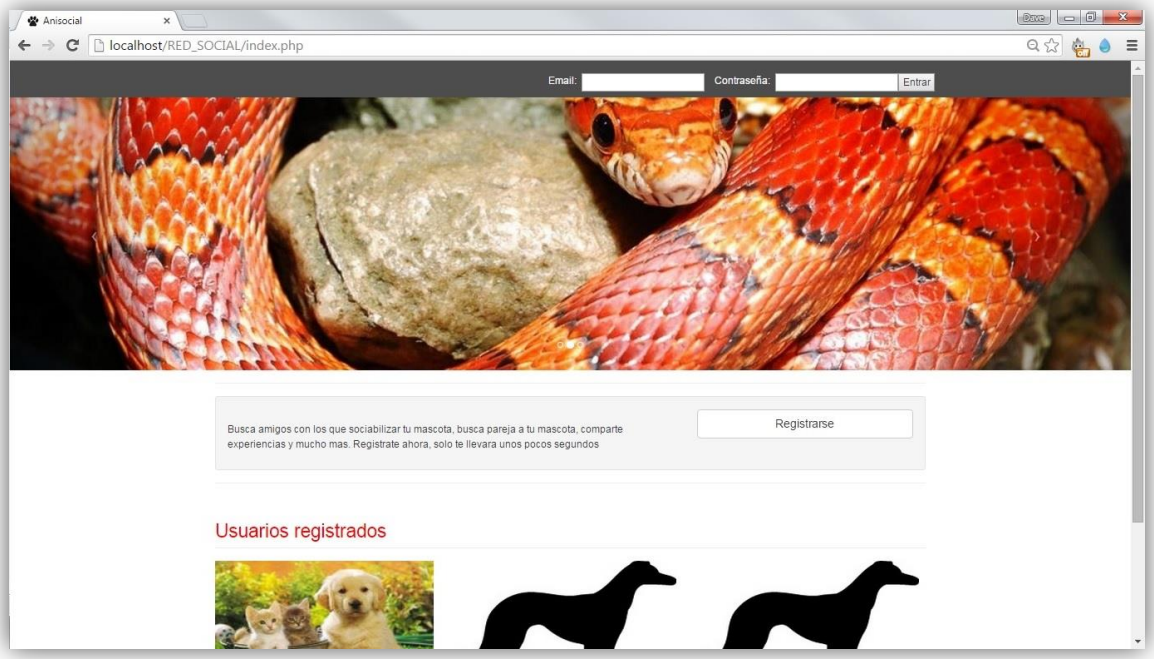

Figura 32 Captura de pantalla de la portada en Chrome, resolución 1920x1080

He de destacar que no hay ninguna diferencia significativa entre la resolución 1920x1080 y la 1366x768 anteriormente citada.

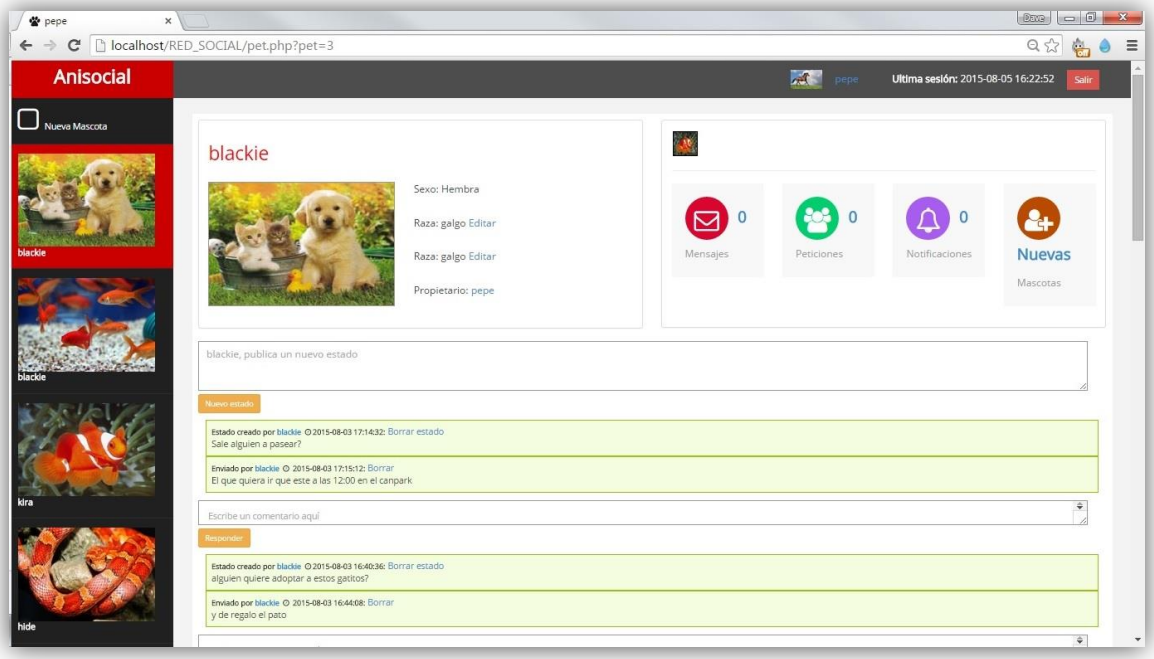

Figura 33 Captura de pantalla del muro en Chrome, resolución 1920x1080

Firefox resolución1366x768

Ahora procedemos a realizar las pruebas con otro de los exploradores más populares, para ello usaremos la resolución 1366x768, una resolución bastante común en equipos modernos.

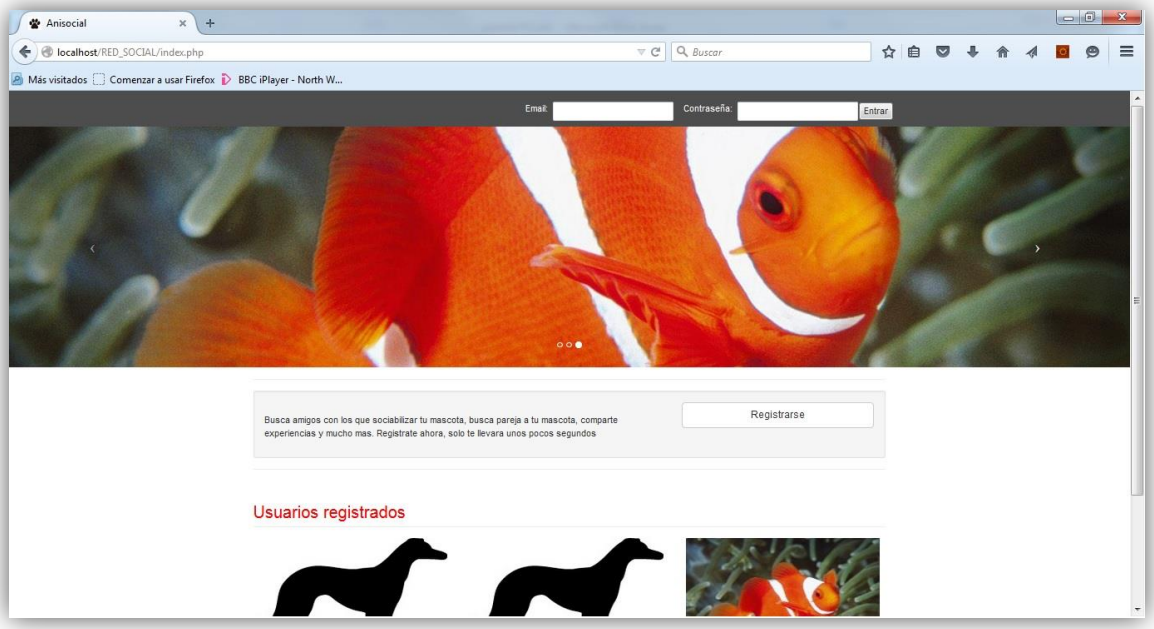

Figura 34 Captura de pantalla de la portada en Firefox, resolución 1366x768

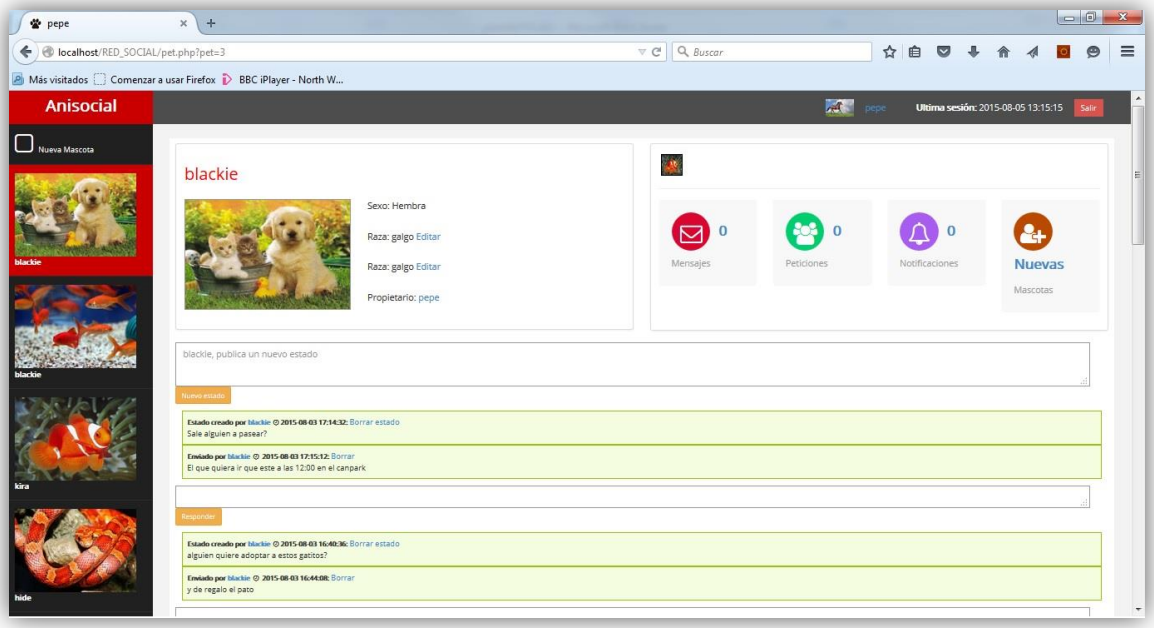

Figura 35 Captura de pantalla del muro en Firefox, resolución 1366x768

Firefox resolución1920x1080

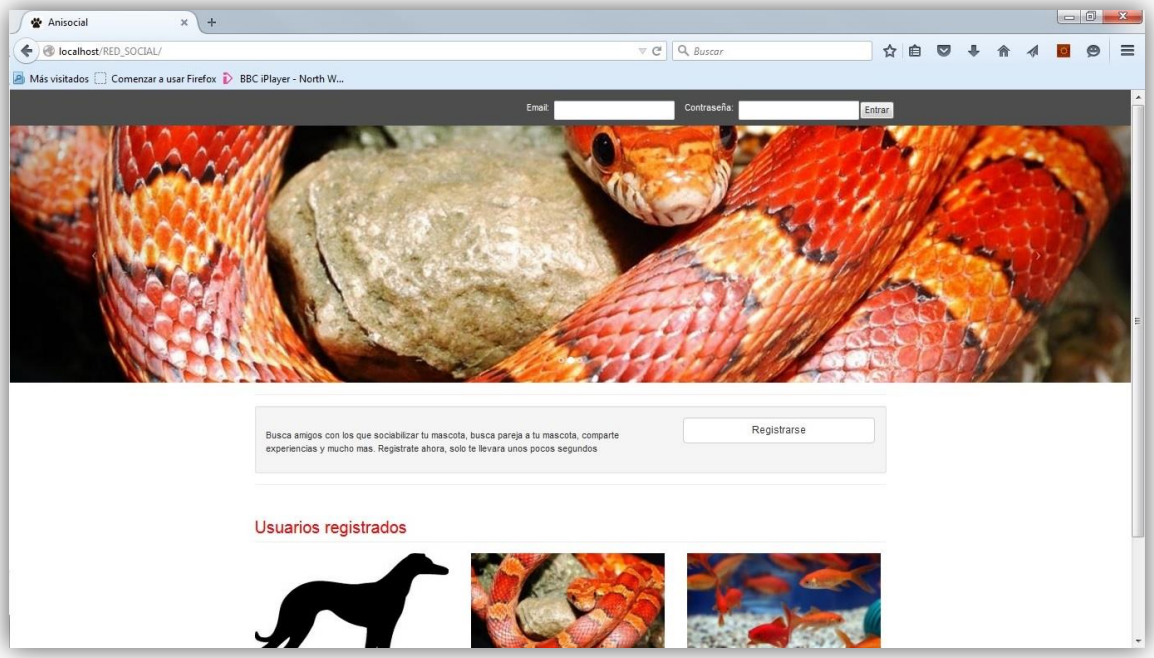

Figura 36 Captura de pantalla de la portada en Firefox, resolución 1920x1080

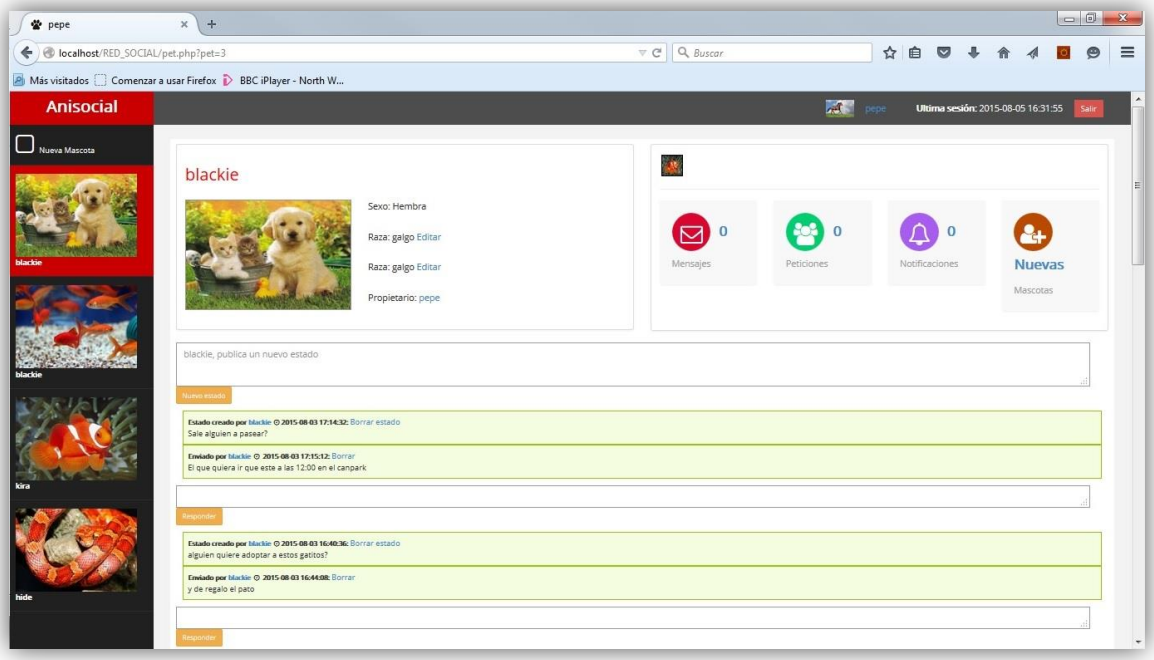

Figura 37 Captura de pantalla del muro en Firefox, resolución 1920x1080

Internet Explorer resolución1366x768

Internet Explorer es quizás uno de los más usados a causa de que viene instalado por defecto en los sistemas operativos de Windows, por lo tanto en todo proyecto web deberemos tenerlo en cuenta.

Figura 38 Captura de pantalla de la portada en Explorer, resolución 1366x768

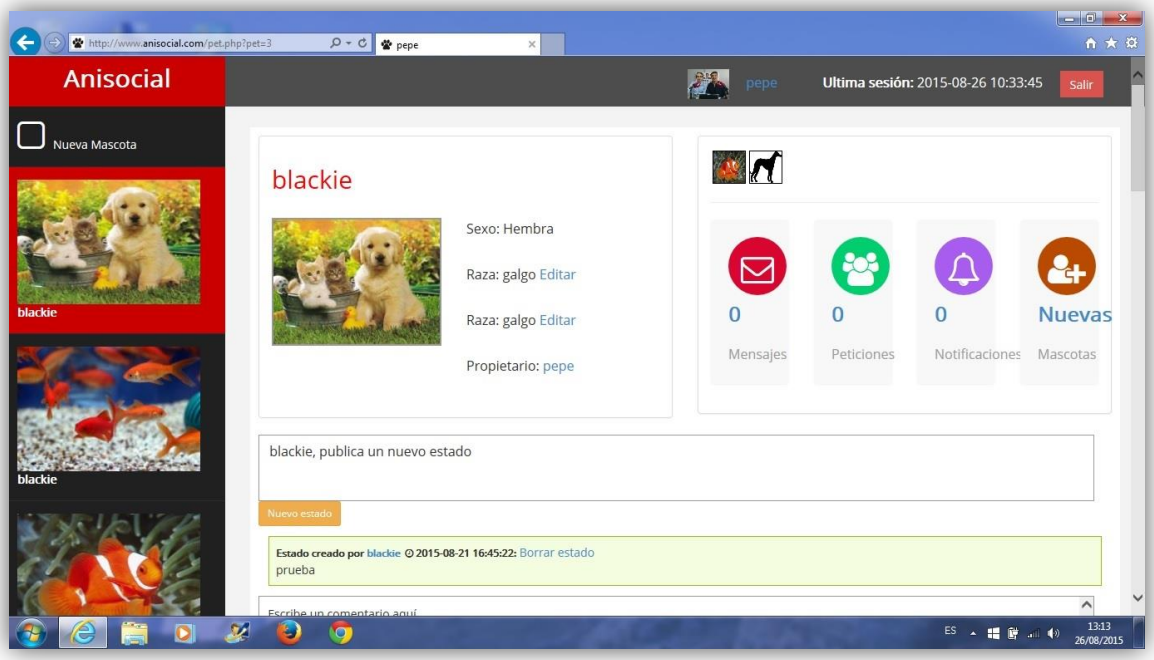

Figura 39 Captura de pantalla del muro en Explorer, resolución 1366x768

Internet Explorer resolución1920x1080

Figura 40 Captura de pantalla de la portada en Explorer, resolución 1920x1080

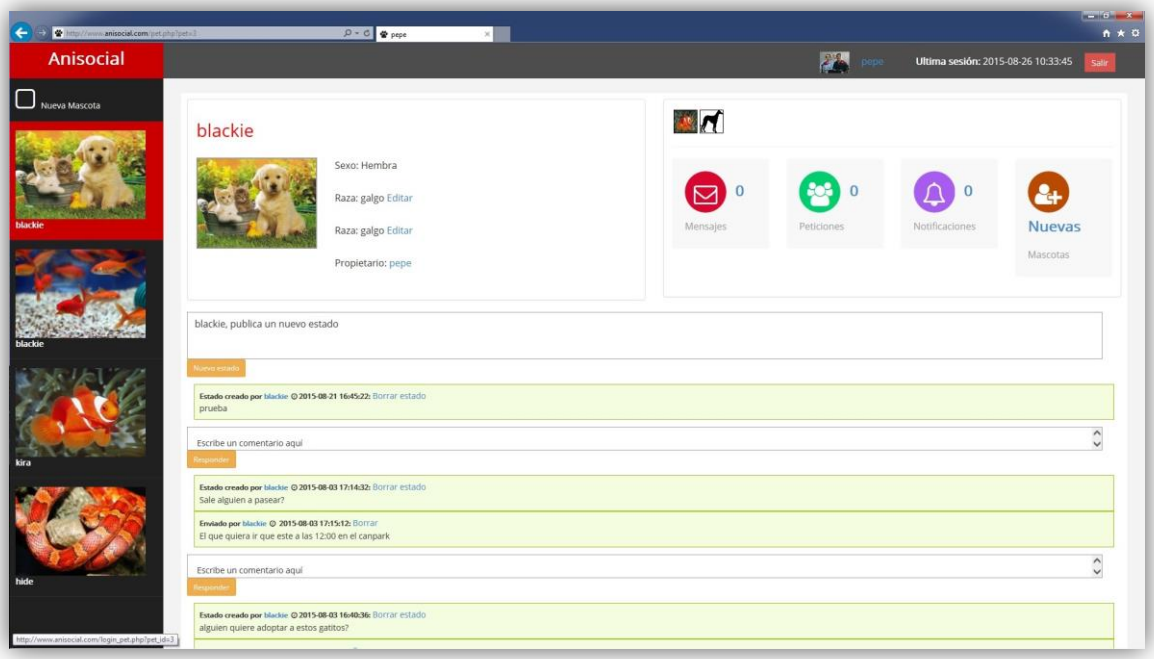

Figura 41 Captura de pantalla del muro en Explorer, resolución 1920x1080

Explorador Safari en un ipad 2

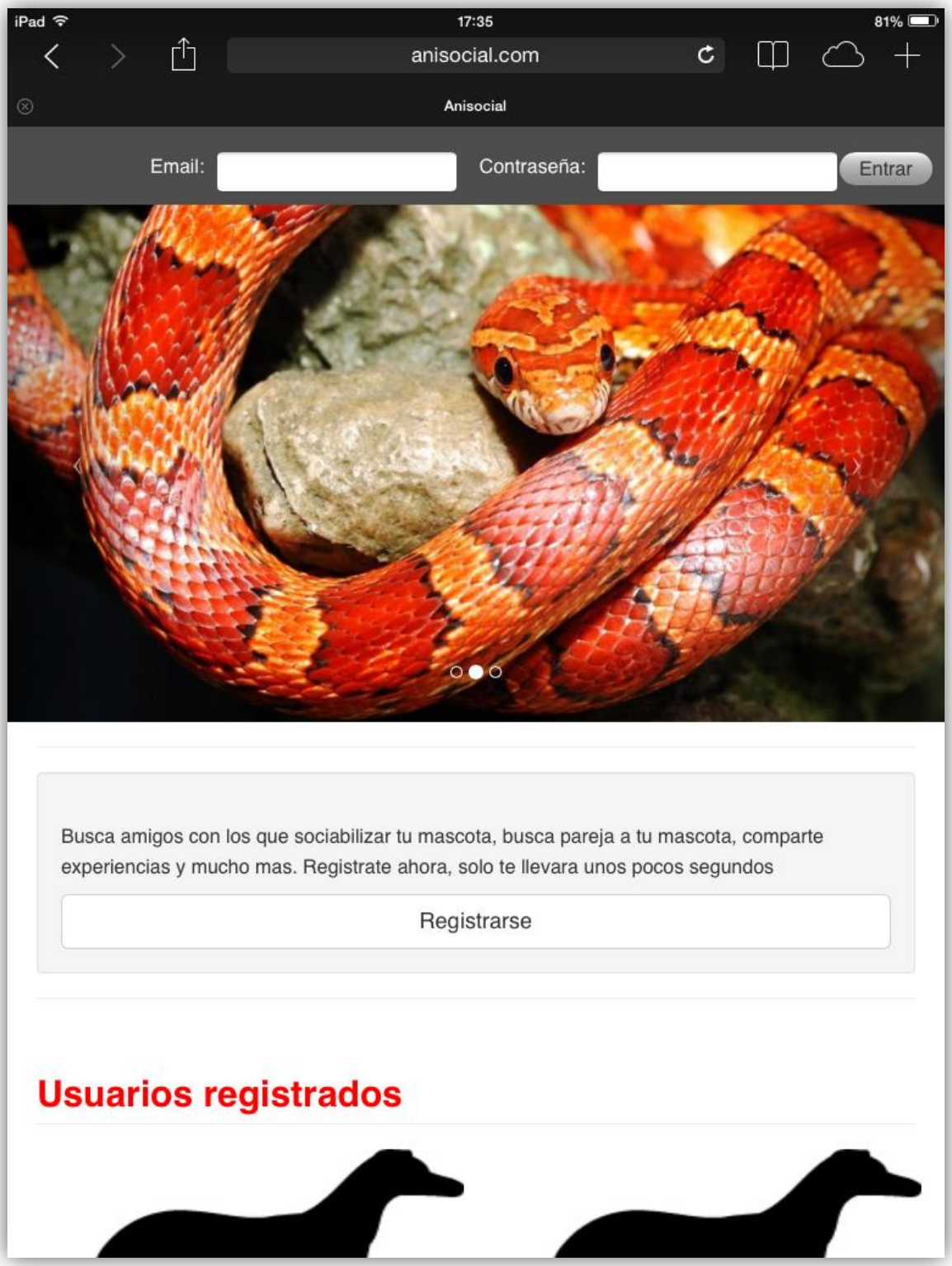

Figura 42 Captura de la portada usando Safari en un ipad 2

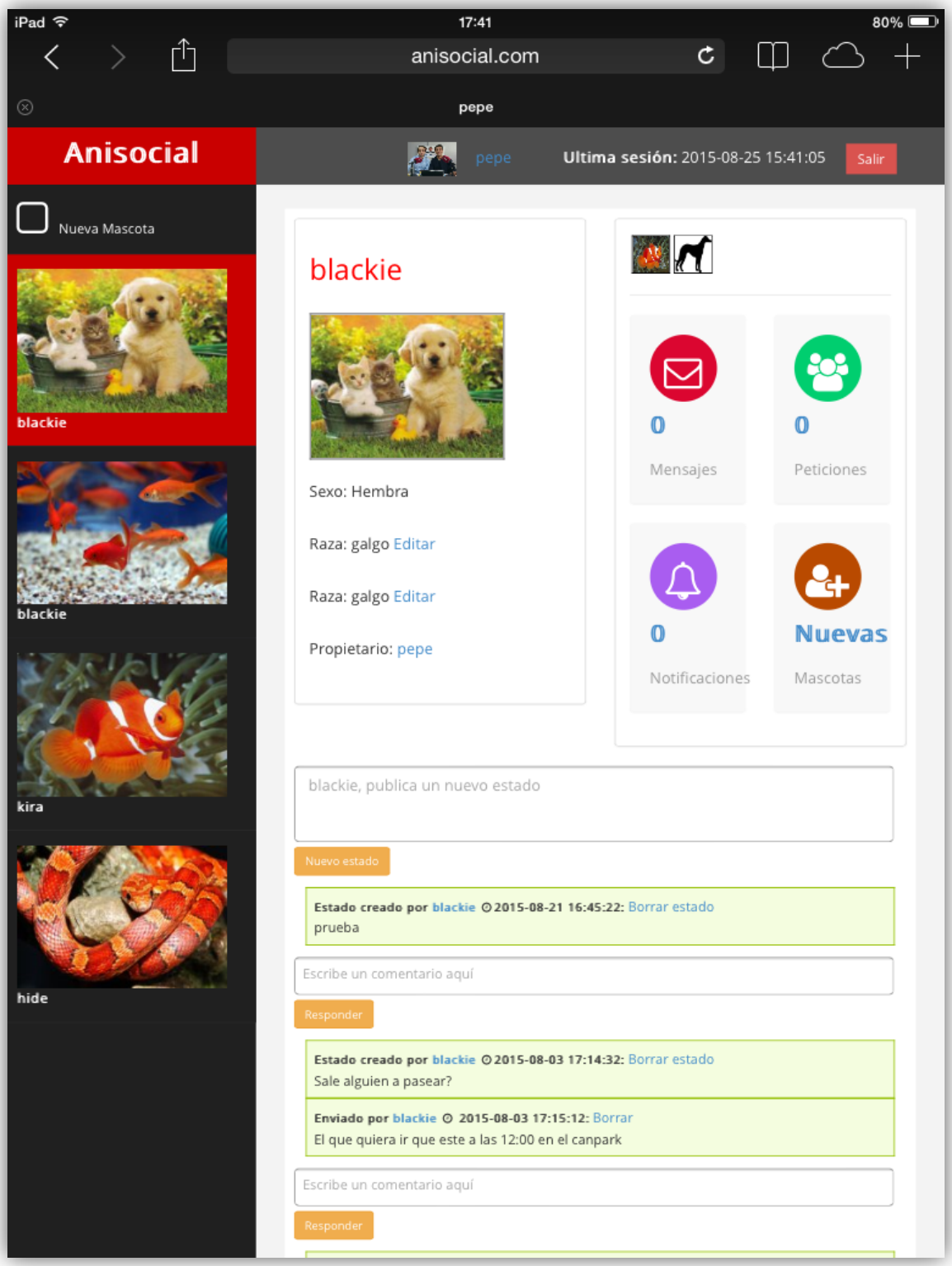

Figura 43 Captura de la interfaz usando Safari en un ipad 2

Explorador Chorme en un Smartphone Android

Figura 44 *Captura de la portada usando un Smartphone Android con chrome*

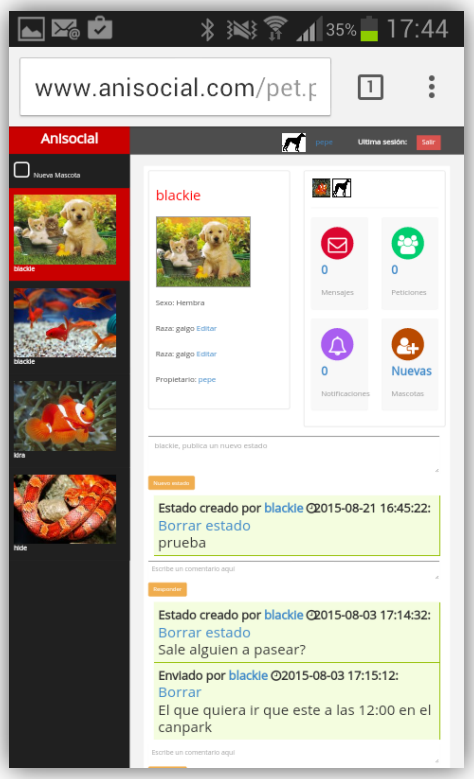

Figura 45 *Captura de la interfaz usando un Smartphone Android con chrome*

6.2. Pruebas de uso

A continuación, se detallarán las pruebas de uso realizadas mediante un recorrido por la interfaz gráfica, para ello se probará con diferentes pantallas y dispositivos, para comparar si la visualización es la que corresponde en cada caso, así como mostrar las diferencias que pudiesen existir entre la interfaz de escritorio y la móvil. He de destacar que Bootstrap facilita mucho la creación de las interfaces ya que sus hojas de estilo son responsive, y por lo tanto existen diferentes configuraciones para cada tamaño de pantalla, haciendo que el diseño se adapte a distintos interfaces, algo muy importante dado el auge del uso de smartphones y tablets.

Las siguientes pruebas se han realizado sobre el explorador Google Chrome en su versión 44.0.2403.125 m.

Pantalla inicio

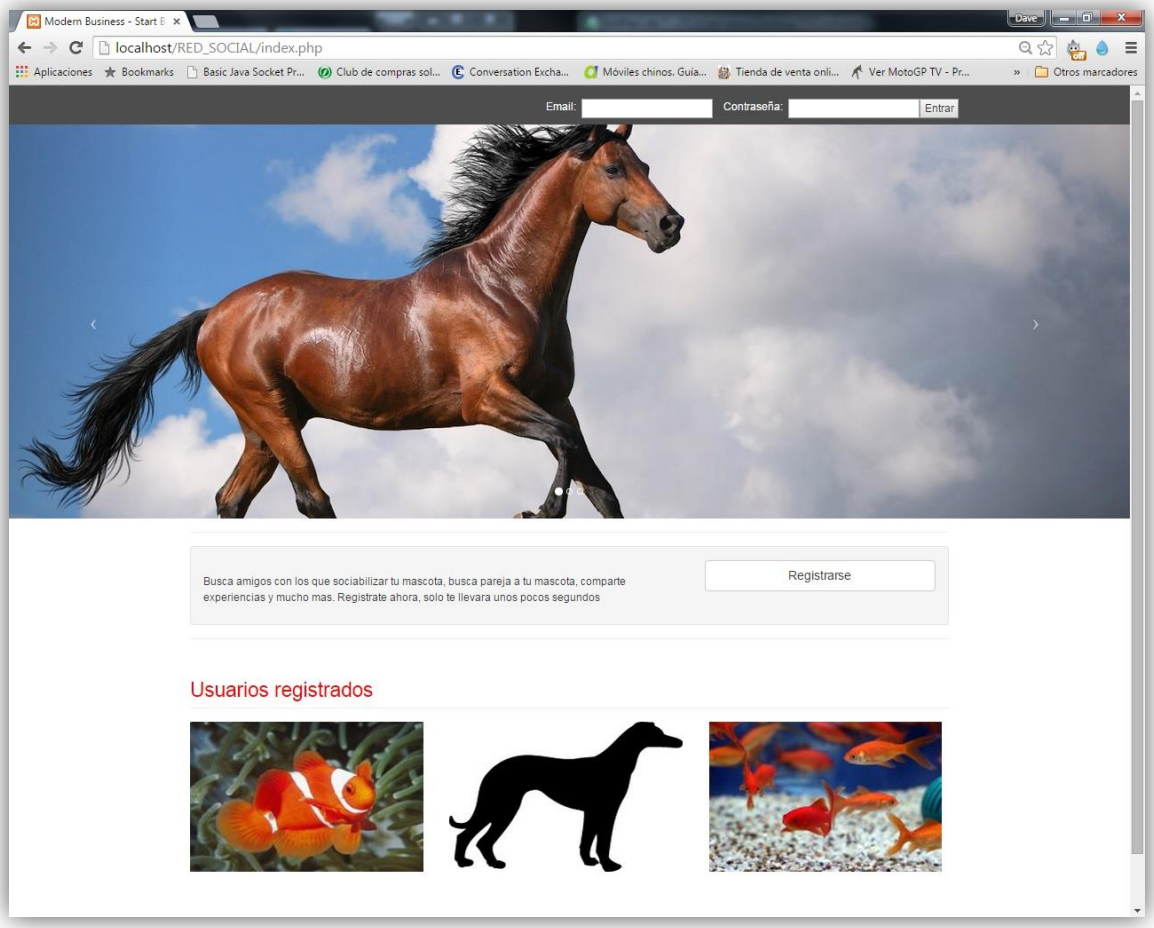

Figura 46 *Pruebas de uso portada de la red social anisocial.com*

Empezaremos mostrando la versión web visualizada con una pantalla de 15". En el inicio observamos que disponemos de la parte de login situada en la parte superior, imitando el estilo de muchos sitios web que optan por ponerlo en primer plano para que no estorbe al usuario y dejando espacio dentro de la portada para publicar la información que se quiera mostrar al visitante. Se escogió esta forma porque se podría ampliar la portada fácilmente añadiendo más subapartados en la parte inferior de la misma, un ejemplo de ampliación es la unión de un blog o un pequeño apartado de noticias. También a modo de mejora, se podría añadir noticias sobre el carrusel de imágenes pudiendo ser usado a modo de propaganda de eventos.

Pantalla Registro

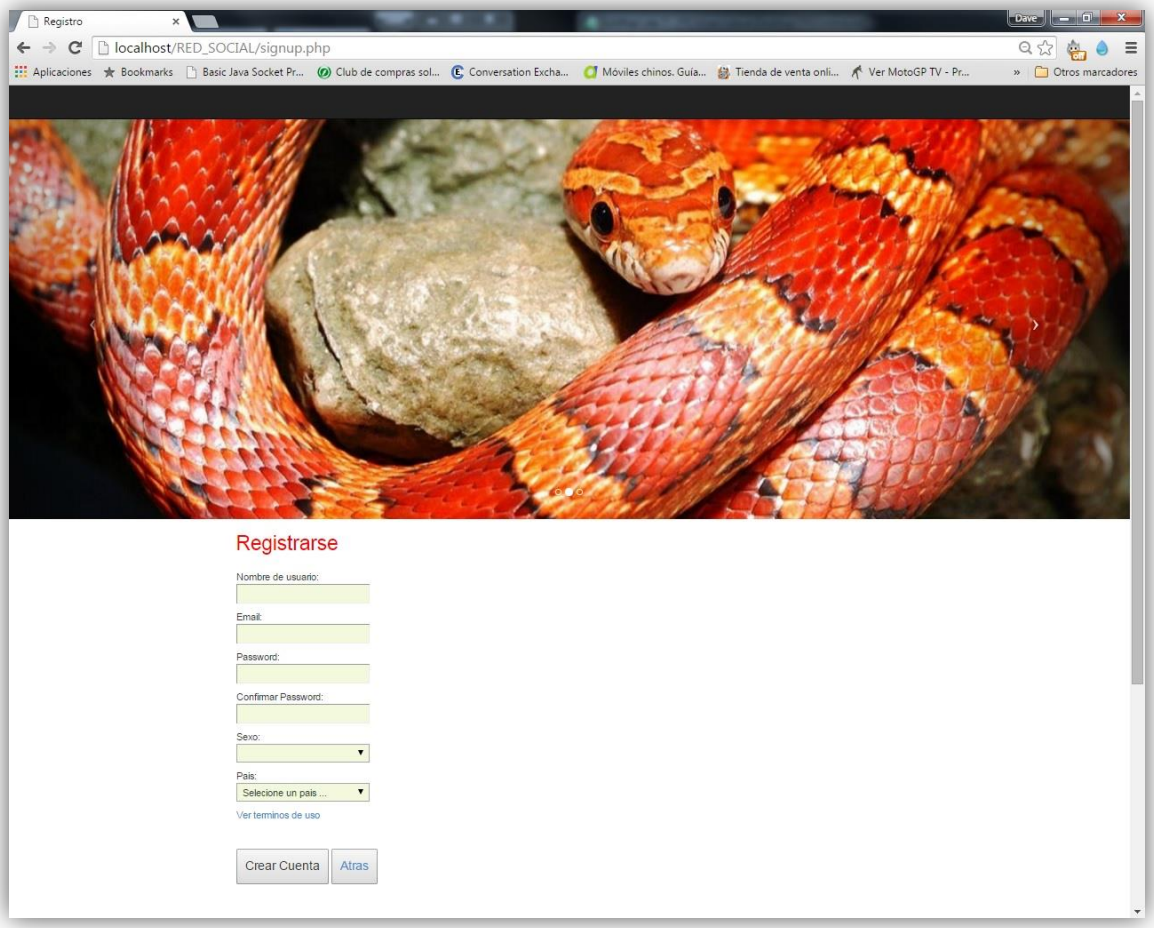

Figura 47 *Pruebas de uso, registro usuario de la red social anisocial.com*

En la pantalla de inicio, al pulsar sobre Registro se abrirá la siguiente ventana donde podremos darnos de alta como usuario. El apartado de registro se ha realizado con la mayor simplicidad posible, ya que de haber incluido muchos sistemas de verificación, o incluir demasiados campos, el usuario podría desistir en el registro y por lo tanto conviene que sea simple y con el menor número de campos a rellenar, aun así, el usuario deberá visitar el campo ver términos, el cual lleva una cláusula que indica que se aceptan las condiciones expuestas al registrarse, evitando de esta forma tener que incluir un checkbox de acepto las condiciones.

Email activación

Una vez finalizado el registro del usuario, se nos enviará un email donde deberemos confirmar el registro. Para ello pulsaremos sobre el enlace que se nos proporciona, el cual nos activará la cuenta y a partir de ese momento podremos usar nuestro usuario.

Este método de validación es quizás uno de los más usados, ya que a diferencia de los captcha, éste hace que el usuario proporcione un email válido, algo de utilidad para en un futuro poder enviar emails al usuario y recordarle las notificaciones pendientes o cualquier otra información que pensemos que es de utilidad para el usuario.

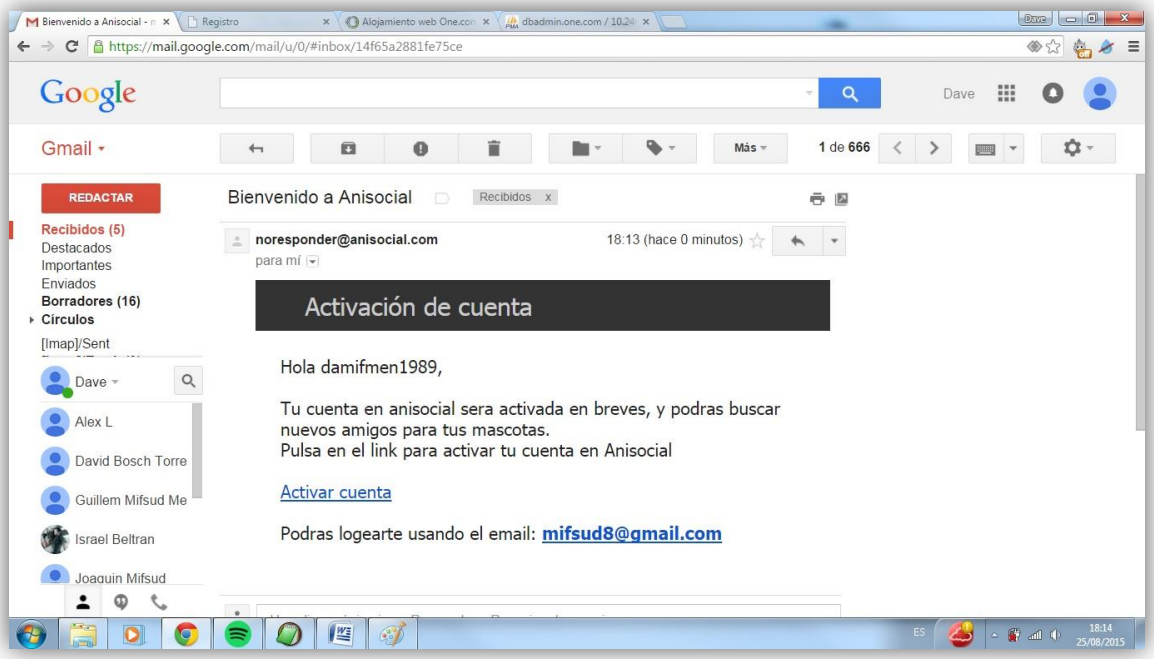

Figura 48 Captura email de activación de la cuenta

Pantalla principal usuario

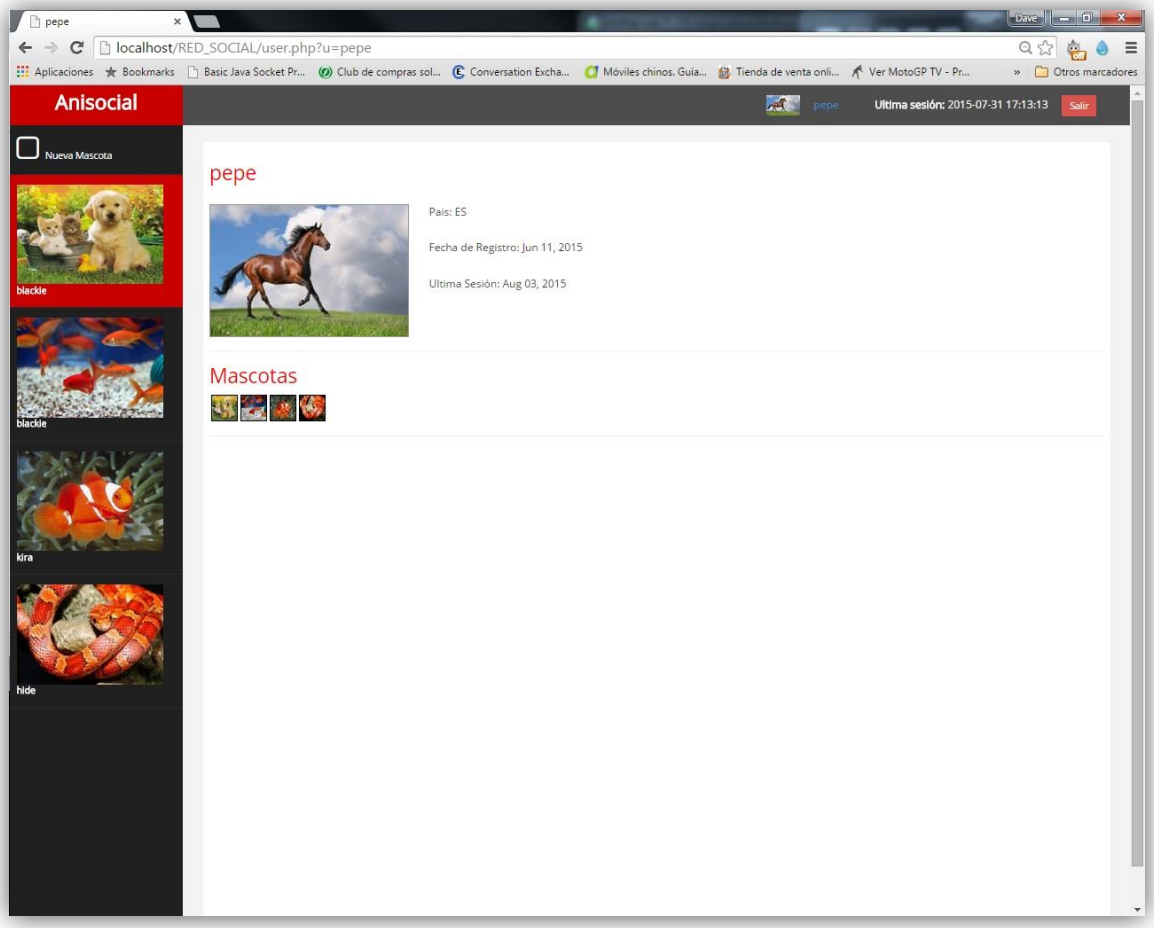

Figura 49 *Pruebas de uso, pantalla principal del usuario de la red social anisocial.com*

Una vez registrados, volveremos al inicio de la aplicación y nos podremos logear en el sistema, la primera ventana que veremos será la del usuario. A pesar que el usuario es un mero trámite y no es de interés para este proyecto, he pensado en incluir sus datos en la portada así como mostrar sus mascotas. Si en un futuro alguien quisiera ampliar el proyecto podría mostrar estadísticas o cualquier otra información que pudiese ser de interés al usuario nada más iniciar sesión.

Como se puede observar en esta captura de pantalla, disponemos de las mascotas registradas en el menú lateral, al pulsar sobre alguna de ellas, nos autenticaremos como esa mascota y accedemos a su muro donde encontraremos sus publicaciones su menú y todo lo relacionado con nuestra mascota.

Pantalla registrar mascot

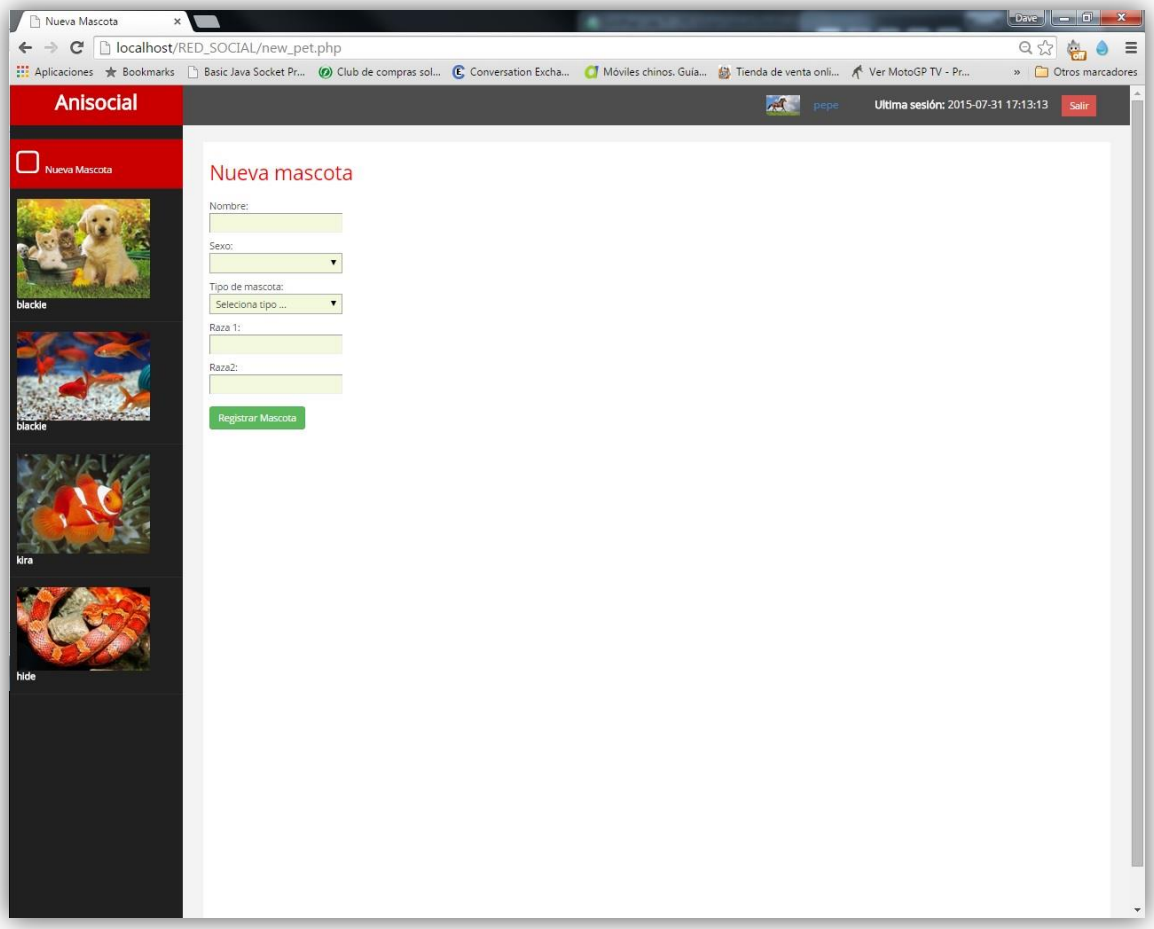

Figura 50 *Pruebas de uso, registro de una nueva mascota en anisocial.com*

Los pasos para registrar a una mascota son muy simples, solo se ha de rellenar el formulario y posteriormente la nueva mascota aparecerá en nuestro menú lateral, no hace falta ningún tipo de confirmación.

He de destacar que en caso de que el explorador en el que se esté visualizando la página web no soporte Javascript no supondrá ningún problema, ya que hay una doble comprobación, por una parte se comprueba la información mediante Javascript y una vez se ha visto que los datos son correctos se envían al servidor, el cual los vuelve a comprobar. Este método puede parecer ineficiente, pero no lo es, ya que de esta forma la primera comprobación de los datos recae sobre el cliente y no sobre el servidor, de esta forma se evita saturar a nuestro servidor web.

Pantalla mascot

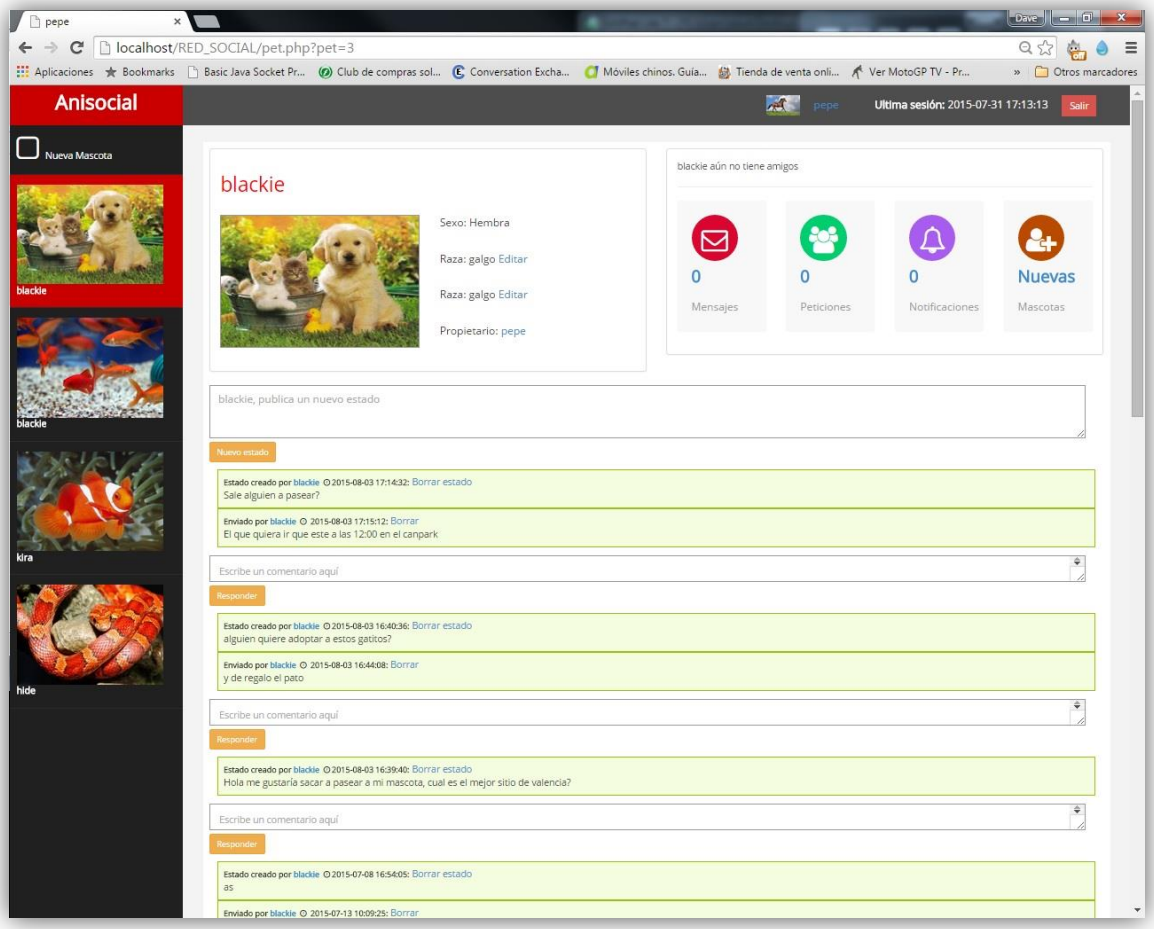

Figura 51 *Pruebas de uso, muro de la mascota en la red social anisocial.com*

Ahora pasamos a una de las partes que más interesan en este proyecto, ya que se centra en las mascotas de los usuarios cuya finalidad es esta red.

Por una parte disponemos de la misma barra lateral con los animales que tenemos registrados, esta barra estará siempre presente en toda la interfaz.

Como se puede observar se le da más importancia a las mascotas que al propio usuario. Un claro ejemplo es el tamaño de las imágenes, estas son 5 ó 6 veces más grandes que la del propietario de las mascotas, esto también facilita que la red pueda ser usada por gente con alguna discapacidad visual.

Por la parte central disponemos del muro de la mascota, el cual se compone de las publicaciones y la información de la mascota, esta última puede ser modificada en la misma página, sin necesidad de acceder a otro apartado, facilitando la modificación de la misma y evitando tener que cargar una nueva página.

En la siguiente imagen se puede observar que para cambiar la imagen debemos pulsar sobre el enlace que aparece sobre la imagen, con lo cual podremos visualizar el formulario para cambiar la imagen de perfil de nuestra mascota.

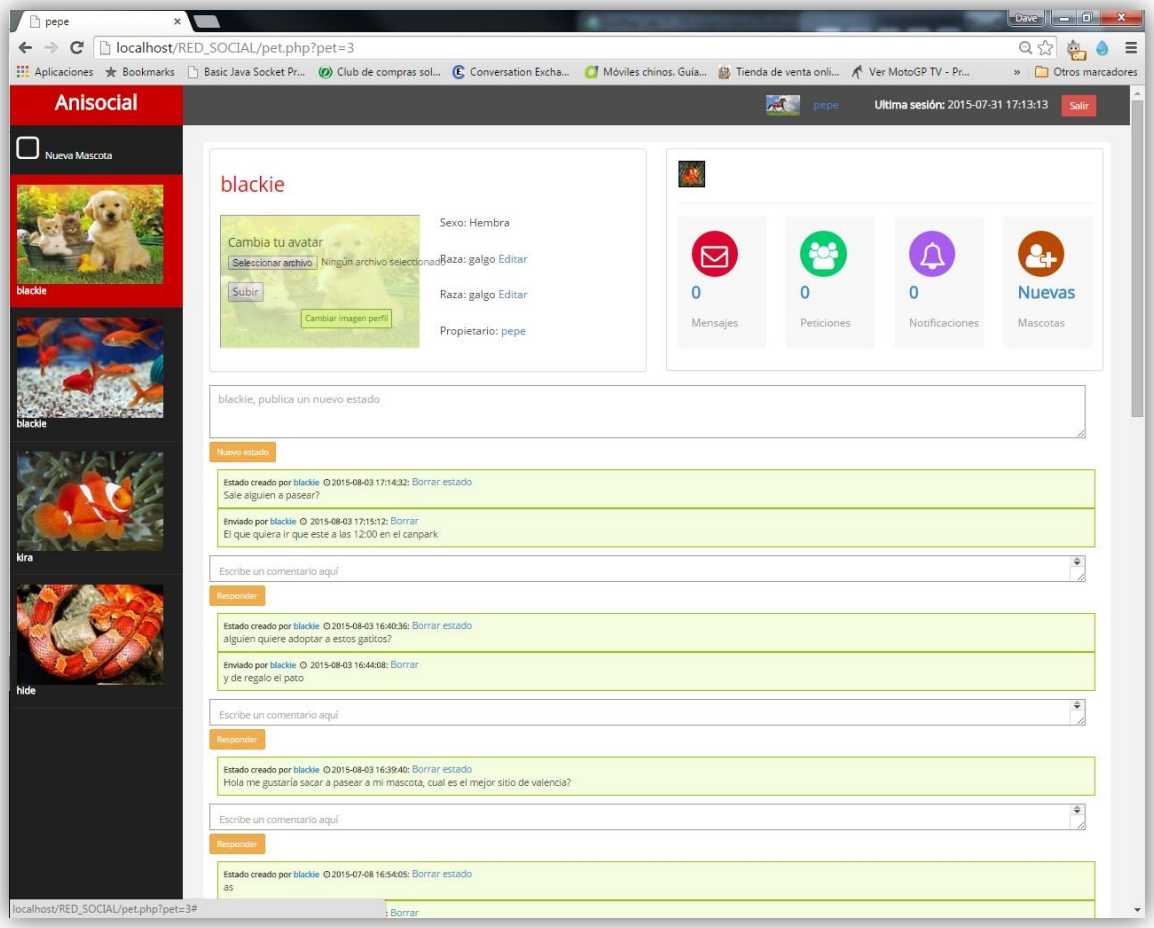

Figura 52 *Pruebas de uso, cambio de imagen de perfil de la mascota en anisocial.com*

Por otra parte el menú de la mascota siempre estará en el lateral derecho, el cual pertenecerá siempre a la mascota autenticada, en caso de que estuviésemos en el perfil de un amigo, este menú también aparecerá siendo éste el de nuestra mascota. Dentro del mismo menú se muestran los avisos de si se ha recibido algún mensaje, petición o notificación, y en la parte superior se mostrarán los amigos de la mascota con la que estamos autenticados.

A continuación se explorarán las opciones del menú de la mascota, en primer lugar iremos al apartado de mensajes, el cual accedemos desde el menú pulsando sobre el primer icono de color rojo.

Pantalla mensajes

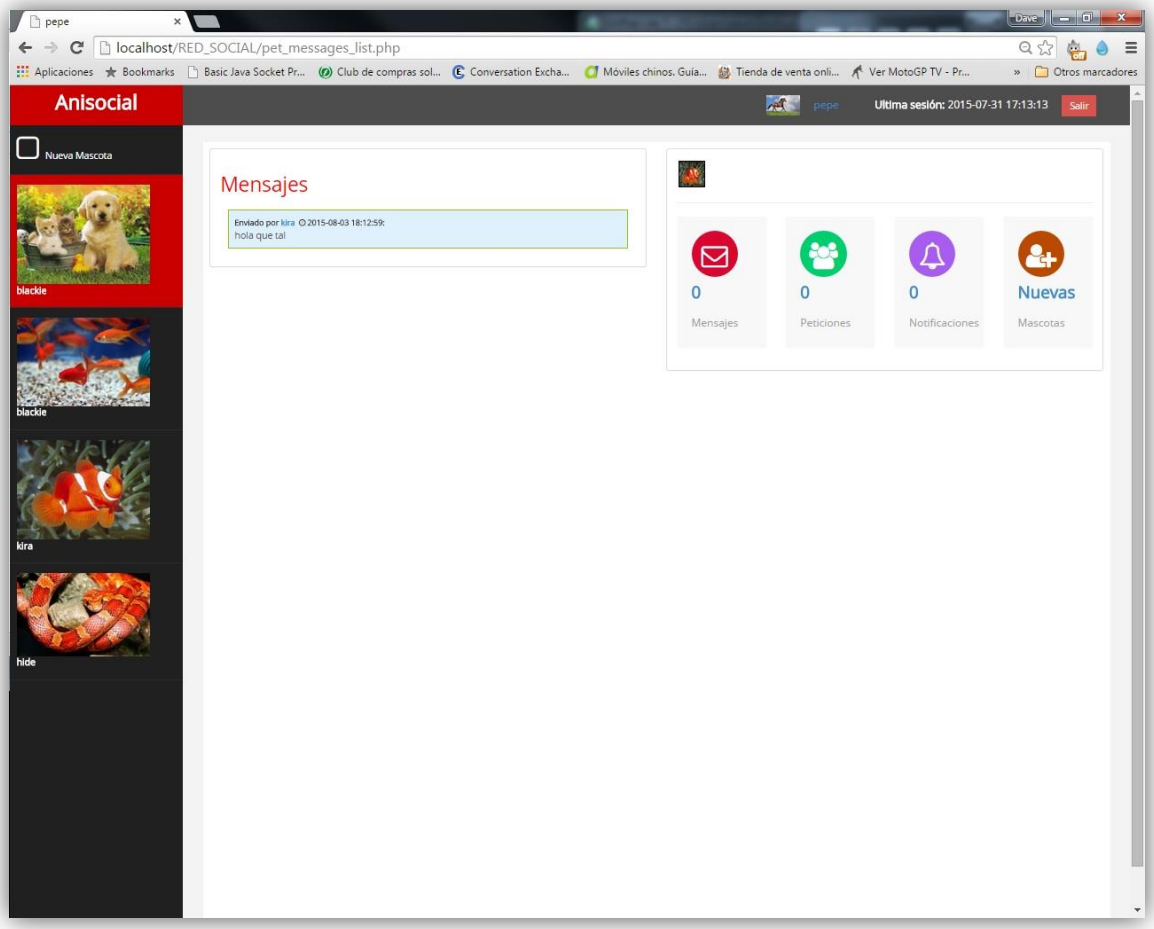

Figura 53 *Pruebas de uso, apartado de mensajes en anisocial.com*

El sistema de lectura de mensajes es muy simple, ya que la principal finalidad de las redes sociales es la de enviar publicaciones. En este caso los mensajes aparecen en primer plano, no como los emails que llevan un encabezado y un título. Se optó por esta visualización por simplicidad y por el hecho de que los mensajes pueden ser revisados con mayor rapidez, además en la versión móvil este tipo de visualización facilita la usabilidad, los mensajes nuevos se visualizarán de color azul quedando éstos más resaltados que los leídos.

Pantalla peticiones

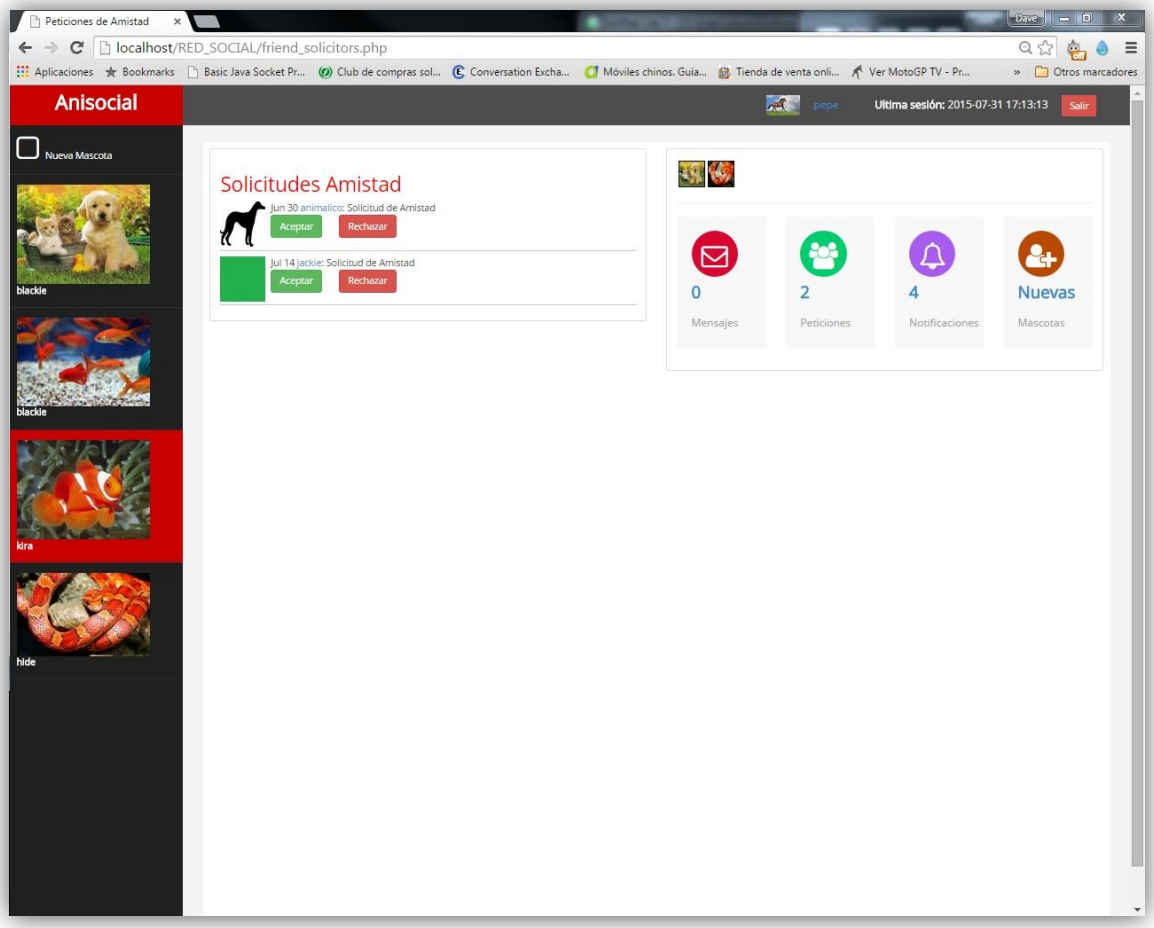

Figura 54 *Pruebas de uso, apartado de solicitudes de amistad en anisocial.com*

Siguiendo el orden del menú exploramos el apartado de peticiones, el cual accedemos a través del icono verde, las mascotas podrán recibir solicitudes de amistad de otras mascotas y éstas serán evaluadas en el apartado correspondiente. El sistema de solicitudes de amistad es básicamente como el de cualquier red social, excepto el hecho de que no existe ningún sistema de bloqueo de usuarios no deseados, ya que esta función no tenía mucho sentido en este proyecto.

Pantalla notificaciones

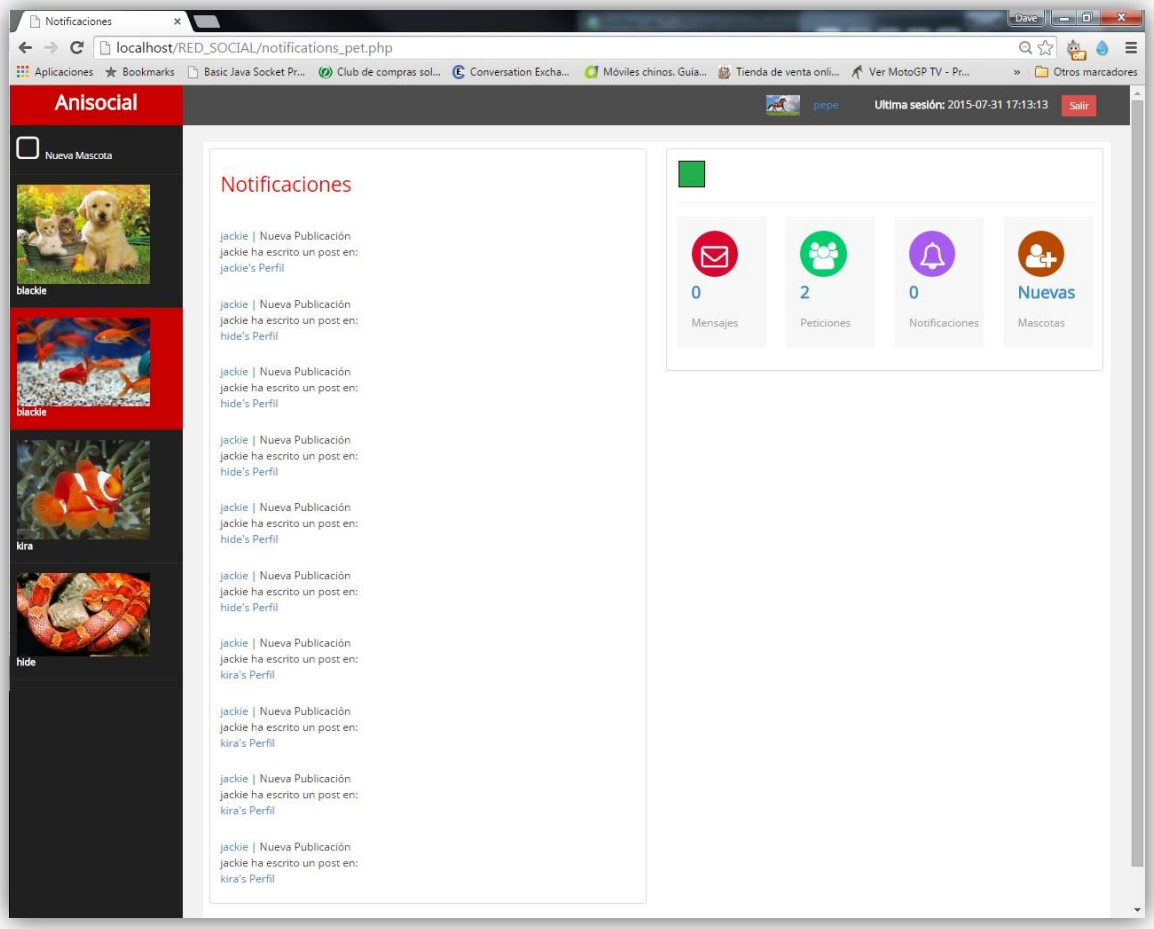

Figura 55 *Pruebas de uso, apartado notificaciones de la mascota en anisocial.com*

Siempre que se realice una publicación en los perfiles de nuestros amigos o en el nuestro, recibiremos una notificación en la cuenta de nuestra mascota, éste es un método simplificado de un timeline.

Pantalla nuevas mascotas

En este apartado la mascota podrá enviar solicitudes de amistad a otras mascotas, una cosa a destacar en este apartado es el hecho que desaparece el menú, en este caso consideré quitarlo para dejar más espacio a las imágenes de las mascotas.

La búsqueda se realiza mediante el uso del formulario alojado en la parte superior, los animales que cumplan con los requisitos de búsqueda serán mostrados de forma agrupada de 3 en 3 en cada fila.

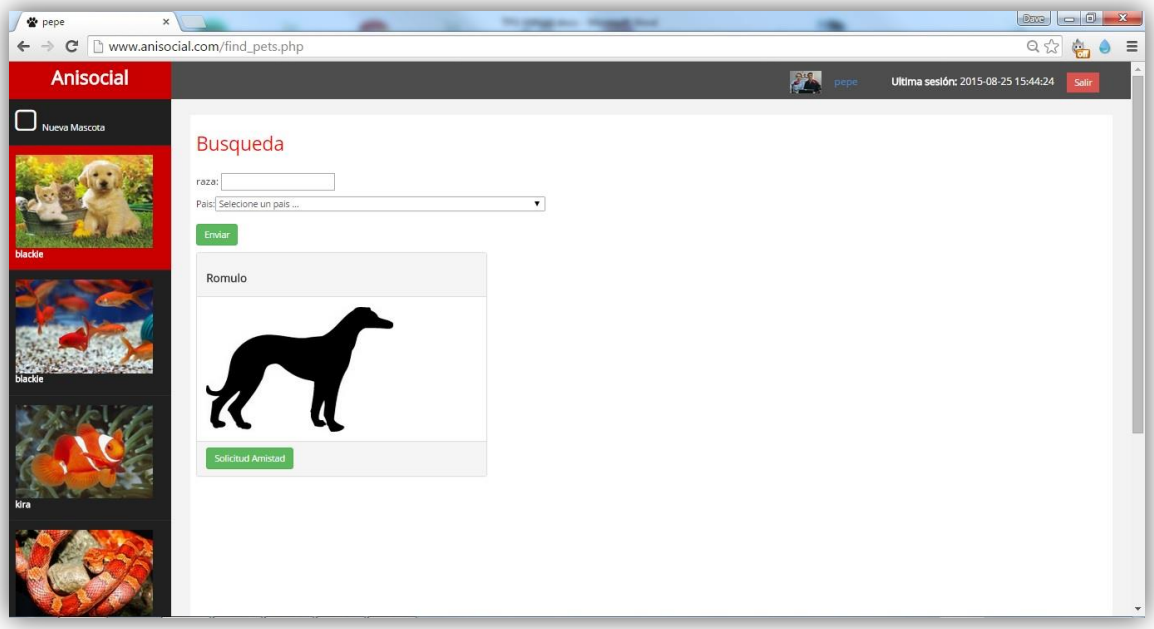

Figura 56 Pruebas de uso, apartado nuevas mascotas en anisocial.com

7. Conclusiones

Finalmente se exponen mis conclusiones acerca de este TFG, las cuales quedan separadas entre técnicas y personales. También he añadido algunas posibles ampliaciones acerca de este proyecto, que podrán ser realizadas por otros alumnos o usarse como material para otras asignaturas sobre programación de webs dinámicas.

7.1. Técnicas

Durante la realización de este TFG he aprendido lo importante que es el uso de las técnicas de planificación a la hora de realizar un software, puesto que la mitad del trabajo depende de esa planificación.

También he tenido un primer contacto con el framework Symfony 2. A pesar de haber abandonado la realización del proyecto con este framework a causa de su más que pronunciada curva de aprendizaje, me ha servido para tener un primer contacto. Quizás en un futuro vuelva a usarlo, aunque personalmente me pareció un framework destinado más a un ámbito muy profesional y no aconsejaría su uso como primer framework. También he de destacar que el uso de frameworks para desarrollos web tiene bastantes ventajas, como son la reutilización de los bundles que otros usuarios ponen a tu disposición agilizando así la programación.

También aprendí a usar el framework CodeIgniter, el cual es mucho más intuitivo y fácil de aprender.

He aprendido que el uso de Javascript en las aplicaciones webs es de vital importancia para reducir el tamaño de la carga que soporta el servidor ya que se pueden hacer múltiples comprobaciones en la maquina del usuario, evitando que todas éstas se produzcan en nuestro servidor y de esta forma ahorrar potencia computacional.

Por último y no menos importante descubrí como automatizar la gestión de un sitio web mediante el uso de tareas programadas o cronjobs, algo de gran importancia en proyectos de gran tamaño.

7.2. Personales

A nivel personal realizar este proyecto me ha dado una visión de la importancia de una de las ramas de la informática como es la ingeniería de software, para poder planificar un proyecto de tamaño medio o grande, ya que sin la debida planificación de

los proyectos, la inmensa mayoría quedan abocados al fracaso o al abandono del mismo.

También debo destacar la necesidad de la realización de más asignaturas sobre programación web, así como asignaturas que ayuden al incremento de la seguridad de esta clase de sistemas.

Este proyecto ha supuesto a nivel personal un crecimiento en mi capacidad de autoaprendizaje, ya que al inicio del mismo no poseía todos los conocimientos necesarios sobre el lenguaje php para llevarlo a cabo. Este incremento de mi capacidad de autoaprendizaje será de agradecer ahora que me acerco al mundo laboral.

Otra cosa que me hubiese gustado, es que en el grado de ingeniería informática hubiese alguna asignatura que impartiera la utilización de un framework PHP ya que de esta forma se acercaría más la realización de estudios universitarios al mundo laboral.

7.3. Futuras ampliaciones

Durante mi investigación previa a la realización del proyecto encontré algunas funcionalidades interesantes, muchas de las cuales me hubiese gustado implementar, pero por desgracia, la falta de tiempo me ha hecho imposible incluirlas dentro de este TFG, por lo cual pongo a disposición de quien quiera continuar con la ampliación de este proyecto un listado de posibles ampliaciones.

- Añadir en el formulario de registro si el usuario es un particular o si por el contrario es una protectora de animales, en el caso de que sea una protectora que se indique en todos sus animales que están en proceso de adopción. También se podría añadir la funcionalidad “adopción de mascota” para usuarios particulares, pero solo cuando lo soliciten expresamente.

- Un apartado en el que se cuente alguna curiosidad acerca de la raza a la que pertenece la mascota.

- Añadir más seguridad, añadiendo un *captcha* en el registro de usuario, o cuando se produzca un número determinado de intentos de acceso se limite al usuario el acceso en 15 minutos o que aparezca otro *captcha* para asegurarse que no es un usuario malicioso que intenta acceder al sistema mediante ataques de diccionario.

- Crear la funcionalidad “petición de crianza”, ésta puede ser implementada simplemente mediante mensajes a la bandeja de entrada con un color distinto o bien en otro apartado de la red.

- Integrar una tienda online para la venta de productos para las mascotas.

- Integración de un servicio de veterinaria con geolocalización con *Google maps* que muestre los veterinarios más cercanos.

- Integración con las cuentas de otras redes sociales, de esta forma, el usuario no necesitaría registrarse, pudiendo utilizar su cuenta de Facebook, Google+ o cualquier otra red social.

- Añadir la funcionalidad de transferir un animal a otro usuario.

- Crear un sistema de invitaciones vía correo electrónico por el cual el usuario pueda enviar un número limitado de invitaciones.

- Traducción de esta red social al inglés.

- Crear los denominados "retos", es decir, crear retos temporales en el tiempo para crear expectación, por ejemplo: Consigue 10 amigos en 1 semana, sube 10 fotos a tu galería o comenta 5 publicaciones. La finalidad sería hacer que los usuarios se sientan atraídos por esta web y terminen usándola regularmente.

- Creación de la parte administradora de la web, de esta forma se puede poner en manos de personal sin la necesidad que éstos tengan conocimientos sobre administración de webs. Así se podrán resolver incidencias y eliminar fácilmente usuarios que incumplan ciertas condiciones, como medida de seguridad para la parte de administración web, que debería ser administrada solo en local o bajo ciertas medidas de seguridad.

- Crear un *timeline* donde aparezcan todas las publicaciones de los usuarios y sus amistades recientes.

- Apartado de sugerencia de amistad con otras mascotas basándose en las otras mascotas que tienen los usuarios amigos o con las mascotas amigas.

- Crear una versión para invidentes de esta web, en la cual se tendrán que realizar estudios, pruebas con usuarios reales.

- Sección de páginas amigas para promocionar la red social.

- Añadir botones de compartir en redes sociales populares como pueden ser: Facebook, Twitter, Meneame…

- Implementar una sección de anuncios de mascotas perdidas o un sistema que envíe un correo electrónico a todos los propietarios que tengan como amigos a alguna de tus mascotas.

7.4. Problemas y soluciones

A continuación realizaré una descripción de algunos de los problemas que me surgieron al realizar este proyecto, en ellos explicaré como solucionarlos y por qué se tomaron ciertas acciones para poder finalizarlo a tiempo.

- El proyecto lo empecé a realizar en Symfony 2, lo cual fue un error, ya que la curva de aprendizaje me ocupó un tiempo del que no disponía, decidí realizar el proyecto con PHP y finalmente inicie una migración parcial a CodeIgniter, ya que este framework es más sencillo y altamente recomendable para programadores que se están iniciando con la programación MVC.

- Otro problema al que me enfrenté era la ausencia de conocimientos de diseño, algo importante para la realización de un proyecto web, ya que el usuario solo ve la interfaz y no el trabajo realizado debajo de la interfaz, es decir las comprobaciones son importantes para que el sistema funcione correctamente, pero la importancia que los usuarios le dan al aspecto visual es mucho mayor. Para solucionar el problema del diseño hice uso de Bootstrap, el cual resolvió esas deficiencias en la interfaz.

8. Bibliografía

CakePHP. *Cookbook 1.3: Autenticación*. <http://book.cakephp.org/1.3/es/The-Manual/Core-Components/Authentication.html> [Consulta: 17 de Julio de 2015].

Countries lists. *Create your own countries list*.<http://www.countries-list.info/Download-List> [Consulta: 23 de Junio de 2015].

Drupal al Sur. *Enviar correos con XAMPP en local*. <http://drupalalsur.org/videos/enviar-correos-con-xampp-en-local > [Consulta: 18 de Junio de 2015].

Hernández, Ariel . Codigofacilito: Curso CodeIgniter. <https://www.youtube.com/watch?v=w1Cn-CiL-E8&list=PLuaU46Q2O0VojYJHvEahzGgtSIwz9uUD4 > [Consulta: 20 de Julio de 2015].

Khoury, Adam. Web Intersect 2.0: Social Network Production. <https://www.developphp.com/> [Consulta: 15 de Junio de 2015].

Startbootstrap. *Binary theme*. <http://www.binarytheme.com/bootstrap-admin-template-binary-admin/> [Consulta: 21 de Junio de 2015].

Startbootstrap. *Modern Business*. <http://startbootstrap.com/template-overviews/modern-business/>[Consulta: 20 de Junio de 2015].

Tello Montejo, Juan. Especificación de requerimientos de software. <http://www.academia.edu/8929757/Especificaci%C3%B3n_de_requerimientos_de_software_Proyecto_Sistema_para_Gesti%C3%B3n_de_Art%C3%ADculos_Deportivos_LSI_03_Gesti%C3%B3n_de_Ventas>[Consulta: 1 de agosto de 2015].

Wikipedia. *W3C Markup Validation Service*. <https://en.wikipedia.org/wiki/W3C_Markup_Validation_Service> [Consulta: 10 de agosto de 2015].

9. Anexos

9.1. Implementación servidor definitivo

Al finalizar el desarrollo, procederemos a migrar el proyecto a un servidor de *hosting* para que esté disponible al público. Para ello revisaremos la sección portabilidad dentro del apartado requisitos de rendimiento de este mismo documento, el apartado portabilidad nos indica cuáles son los archivos que se deben trasladar dentro del nuevo servidor.

A continuación se muestra una lista de las tareas que debemos realizar para la completa migración del proyecto en el nuevo servidor.

- Alquiler servidor, existen numerosas empresas que se dedican al *hosting* de servidores entre ellas están one o 1and1, ambas son empresas fiables donde alojar nuestro sitio web.

- Registrar dominio, esta fase es una de las más delicadas ya que si no se hace con tiempo pueden surgir problemas, ya que el nombre de dominio puede no estar disponible.

- Crear las BD y trasladar los datos que pudiese contener, para ello exportaremos la BD de datos desde nuestro phpMyadmin, ya que ésta contiene la última versión de la tabla y los datos que hemos insertado para hacer las pruebas, one.com trabaja con phpMyadmin, así que los pasos a seguir serán los mismos que cuando creamos la BD en local. El primer paso será crear una BD en phpMyadmin, una vez creada ejecutaremos el script sql para que nos cree las tablas en nuestra nueva BD. Una vez finalizado este paso, pondremos la contraseña que queramos a nuestra BD desde el panel de control del proveedor de *hosting*.

- Modificar los archivos que contienen la configuración de la BD, para ello deberemos editar el archivo db_conx.php y poner los nuevos datos de la BD, entre ellos pondremos la URL del servidor MySQL del nuevo servidor, la contraseña, el usuario y el nombre de la BD.

- Subir archivos, esta parte consistirá en exportar los archivos guardados en htdocs a nuestro servidor.

- Configurar cronjobs, este apartado no pudo ser configurado en nuestro caso a causa de que los servidores de la empresa no soportaban esta opción, pero

en otros sitios se configuraría mediante un archivo php, el cual contendrá las acciones a llevar a cabo y indicándole la frecuencia con la que queremos que se realice la acción.

- Desactivar errores de PHP. Esta parte deberá hacerse por motivos de seguridad, ya que el atacante puede ayudarse por los errores que reporta PHP. En la siguiente imagen puede verse donde se encuentra esta opción, además deberemos comentar en el código todos los posibles reportes de errores que hayamos puesto, por ejemplo en la conexión de la base de datos.

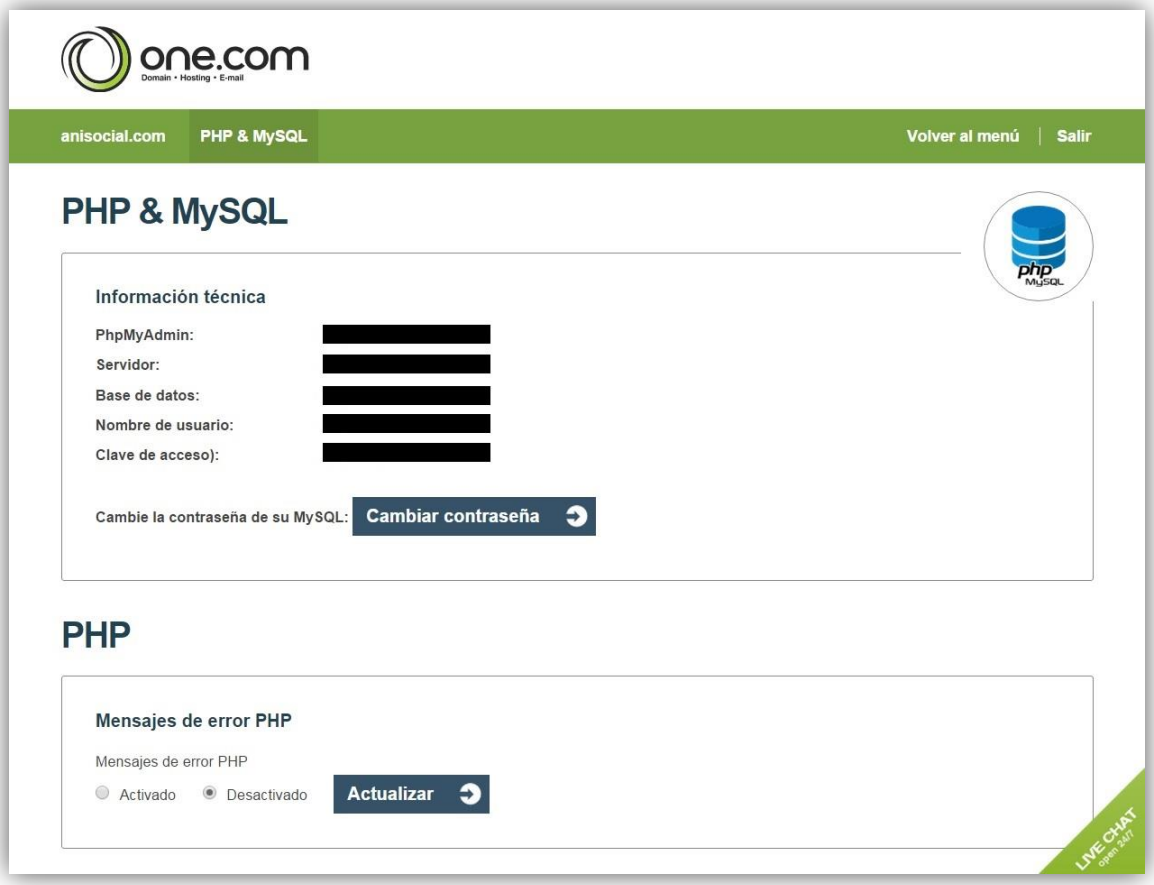

Figura 57 Apartado php&MySQL

9.2. Rentabilizar la aplicación

A continuación se expondrán algunas formas de hacer rentable esta red social, la mayor parte de las redes sociales actuales usan algún tipo de publicidad para obtener fondos y mantener en marcha el sitio web.

- Buscar sponsors, para ello podremos negociar con propietarios de negocios referentes a los animales a los cuales podremos alquilar una sección dedicada a mostrar anuncios referentes a su página web. Si no se nos da bien encontrar patrocinadores, existen agencias dedicadas a la búsqueda de éstos, las cuales reciben una parte del patrocinio a modo de compensación.

- Subcontratar publicidad mediante anuncios de Google, éste quizás sea uno de los métodos más populares de conseguir fondos. Google actualmente es uno de los gigantes del sector de la publicidad online, algo que nos asegurará el pago de nuestros servicios.

- Crear una tienda online para vender productos propios, para ello deberemos crear una tienda online donde vender nuestro merchandaising y de esta forma también servirá para patrocinar nuestra red social.

- Apartado de descuentos, el hecho de ser participante en nuestra red social puede aportar a los usuarios algún tipo de descuento, algo que incrementará las ventas de las tiendas.

- Apartado de tiendas recomendadas, algunas empresas pueden buscar este tipo de servicios de publicidad.

9.3. Re-implementación CodeIgniter

En un principio se optó por no usar un framework por el hecho de querer aprender PHP y porque el hecho de que empecé con symfony, el cual es un framework demasiado profesional para alguien que no había usado un framework PHP anteriormente.

Una vez acabado el proyecto, decidí hacer una reimplementación parcial del proyecto para comprobar si los tiempos eran menores. Para ello se eligió el framework CodeIgniter, el cual considero mucho mejor que Symfony, ya que CodeIgniter facilita la tarea a diferencia de Symfony.

He de decir que escoger un buen framework no debe realizarse en base a cual es el que tiene más trabajos, en ocasiones incluso haciendo un estudio de cuál es el mejor framework nos podemos equivocar en nuestra elección. Para escoger uno deberemos

probar varios y escoger el que nos sea más fácil de usar y/o se adapte a nuestras necesidades.

Para la implementación parcial me centré en crear la gestión de las relaciones de amistad. Primero definí el modelo y posteriormente el controlador, a continuación se muestra el código implementado.

```php
<?php

class M_friendship extends CI_Model{

   function __construct() {
       parent::__construct();
       $this->load->database();


       //CODIGO DE EJEMPLO PARA INICIAR SESSION

       session_start();
       $_SESSION['pet_id']=89;
       setcookie("pet_id",89, strtotime( '+30 days' ), "/", "", "", TRUE);


    }

   function friendship_solicitor($solicitor_id,$id){
               $data = array(
               'pet_id_1' => $solicitor_id,
               'pet_id_2' => $id,
               'accepted' => '0');

               $this->db->insert('friends',$data);

   }



   function decline_petition($my_id,$id){
               $this->db->where('pet_id_2', $my_id);
               $this->db->where('pet_id_1', $id);
               $this->db->where('accepted','0');
               $this->db->delete('friends');
   }

   function accept_petition($my_id,$id){
               $this->db->where('pet_id_2', $my_id);
               $this->db->where('pet_id_1', $id);
               $this->db->set('accepted','1');
               $this->db->update('friends');
   }

   function remove_friendship($my_id,$id){
               $this->db->where('pet_id_2', $my_id);
               $this->db->where('pet_id_1', $id);
               $this->db->where('accepted','1');
               $this->db->delete('friends');

               $this->db->where('pet_id_1', $my_id);
               $this->db->where('pet_id_2', $id);
               $this->db->where('accepted','1');
               $this->db->delete('friends');

   }
```

```
    function get_petitions($my_id){
                $this->db->select('pet_id_2');
                $this->db->where('pet_id_1',$my_id);
                $this->db->where('accepted','0');
                $Q = $this->db->get("friends");

       if ($Q->num_rows() > 0){
           foreach ($Q->result() as $row){
               $data[] = $row->pet_id_2;
           }
       }else{
           $data = array();
       }
       $Q->free_result();

       return $data;
    }



    function get_friends($pet_id_1){
                $this->db->select('pet_id_2');
                $this->db->where('pet_id_1',$pet_id_1);
                $this->db->where('accepted','1');
                $Q = $this->db->get("friends");

       if ($Q->num_rows() > 0){
           foreach ($Q->result() as $row){
               $data[] = $row->pet_id_2;
           }
       }else{
           $data = array();
       }
       $Q->free_result();
       return $data;
    }

}//end class

?>
```

Figura 58 *Modelo amistades*

```
<?php
defined('BASEPATH') OR exit('No direct script access allowed');

class Friendship extends CI_Controller {


        function __construct() {
                parent::__construct();

                $this->load->helper('form');
                $this->load->helper('html');
                $this->load->helper('url');


                $this->load->model('m_friendship');
```

```
        }



        function index(){
            //SOLO TESTING
           $data['title'] = 'prueba peticiones amistad';
           $data['main_view'] = 'friendship/friendship';

                $this->load->vars($data);
           $this->load->view('template');
    }

         function petitions(){
            //SOLO TESTING
                $data['title'] = 'prueba peticiones amistad';
           $data['main_view'] = 'friendship/petitions';
               $data['petitions'] = $this->get_petitions();
               $this->load->vars($data);
           $this->load->view('template');
    }

    function friendship_petition($id){
       $myid = $_SESSION['pet_id'];
       $this->m_friendship->friendship_solicitor($myid,$id);
       redirect("agilan/index", 'refresh');
    }

    function cancel($id){
       $myid = $_SESSION['pet_id'];
                $this->m_friendship->decline_petition($myid,$id);
       redirect("agilan/index", 'refresh');
    }

        function accept($id){
       $myid = $_SESSION['pet_id'];
                $this->m_friendship->accept_petition($myid,$id);
       redirect("agilan/index", 'refresh');
    }

        function remove_friendship($id){
           $myid = $_SESSION['pet_id'];
                $this->m_friendship->remove_friendship($myid,$id);
           redirect("agilan/index", 'refresh');
    }

        function get_petitions(){
           $myid = $_SESSION['pet_id'];
                $data['petitions']=$this->m_friendship->get_petitions($myid);
                $this->load->vars($data);
                return $data;

    }

}
```

Figura 59 *Controlador de la gestión amistades*