Universitat Autònoma de Barcelona

escola d'enginyeria

48899-1: Servicio Web para el cuidado de animales domésticos

Memòria del Projecte Fi de Carrera
d'Enginyeria en Informàtica
realitzat per
Raúl Blasco Rodríguez
i dirigit per
Diego Javier Mostaccio Mancini
Bellaterra,15 de Juny de 2012

Universitat Autònoma de Barcelona

etse
Escola Tècnica Superior d'Enginyeria

El sotasignat, Diego Javier Mostaccio Mancini,

Professor/a de l'Escola Tècnica Superior d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en

I per tal que consti firma la present.

Signat:

Bellaterra, 15 de Juny de 2012

Resumen del proyecto

La crisis económica en la que España sigue inmersa ha afectado al abandono y la adopción de animales. Las protectoras de animales están desbordadas debido a la recogida diaria de animales y así como la reducción de la adopción de mascotas. La adopción disminuye ya que es un gasto más para la familia.

El estado actual de la economía no ha hecho más que agravar el problema de abandono de mascotas por parte de la sociedad española. Un estudio realizado por la Fundación Affinity sobre el abandono de animales demuestra que España es uno de los países de la Comunidad Europea con mayor abandono de animales.

Los motivos del abandono de animales suele ser por la llegada de un bebe a la familia, por nacimiento de camadas o bien porque en realidad han perdido al animal y no llevaba chip identificador. Aunque lamentablemente las mascotas son parte de la sociedad de consumo en la que vivimos y por lo tanto las familias abandonan a sus mascotas porque se cansan, pasan de moda o bien molestan a los vecinos.

El objetivo del proyecto es crear un software que permita dar servicio a familias con mascotas que necesiten encontrar de una manera rápida residencias cercanas, protectoras o bien veterinarios.

En caso de no poder mantener a la mascota, tener la posibilidad de anunciarlo para que otra familia pueda hacerse cargo de nuestra mascota adoptándola.

En esta etapa de crisis económica se dispone de un servicio que permite a familias con mascota ponerse en contacto con personas de su provincia que se responsabilicen de su mascota durante un periodo de tiempo llegando a un acuerdo. De esta manera los dueños de la mascota obtienen un cuidado temporal más económico que una guardería de verano.

Tabla de contenidos

Tabla de figuras

1 Introducción

Es habitual que en transcurso de nuestras vidas las persones se enfrenten a cambios de vida que nunca antes habían considerado. En estos casos muchos propietarios de mascotas encuentran necesario el hecho de tener que abandonar a “uno de la familia”, es decir dejar a las mascotas que ya no pueden mantener.

Este suceso se ha producido aun estando o no estando en crisis y esto ha desembocado a que gente de todo el mundo abandone millones de mascotas.
En la Fig.1-1 se puede observar como en España desde 1998 hasta 2007 ha incrementado 14.833 el número de abandonos únicamente de perros.

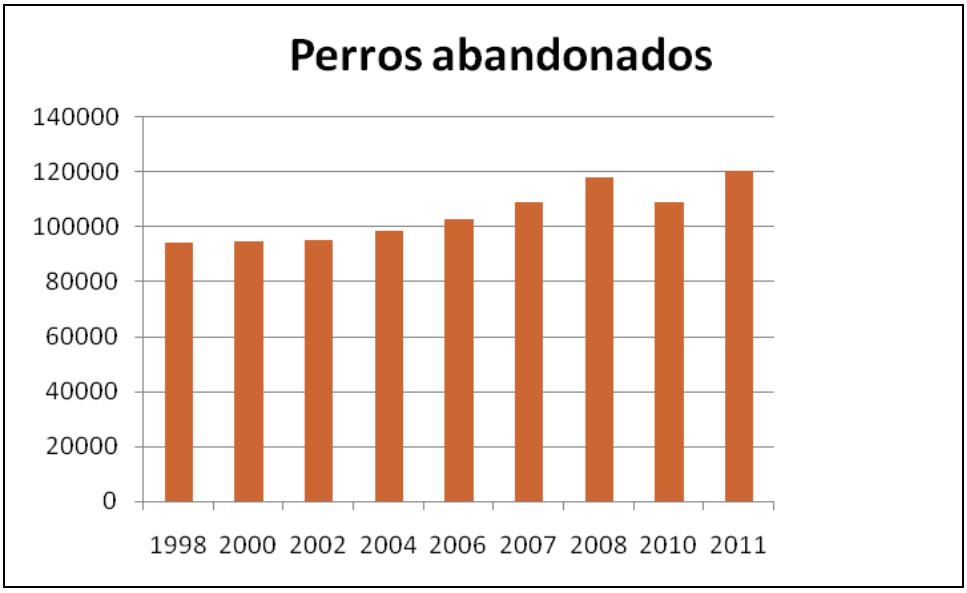

Figura 1-1: Evolución de abandono de perros en España.

Por difícil que sea, los dueños deben responsabilizarse de la mascota que tienen a cargo para resolver el problema.
Es cierto que las personas ven a las mascotas como un producto social que a la que se pierde el interés por el animal o bien aparece un animal novedoso, deshacemos del animal para más tarde cambiarlo por esta nueva mascota más atractiva y novedosa.

Con los estudios realizados se observa en la Fig.1-2 que en la mayoría de casos el abandono del animal viene dado por la pérdida de interés sobre la mascota, por el nacimiento de un hijo que reduce el dinero dedicado en el cuidado de la mascota para invertirlo en el nuevo miembro de la familia. Pero hay que tener en cuenta que en la época que vivimos actualmente lo más normal es que se produzcan cambios de hogar donde anteriormente disponíamos de un espacio suficientemente grande para poder tener nuestra mascota a vivir en un piso reducido para poder reducir gastos y esto implica que los propietarios abandonen a su mascota.

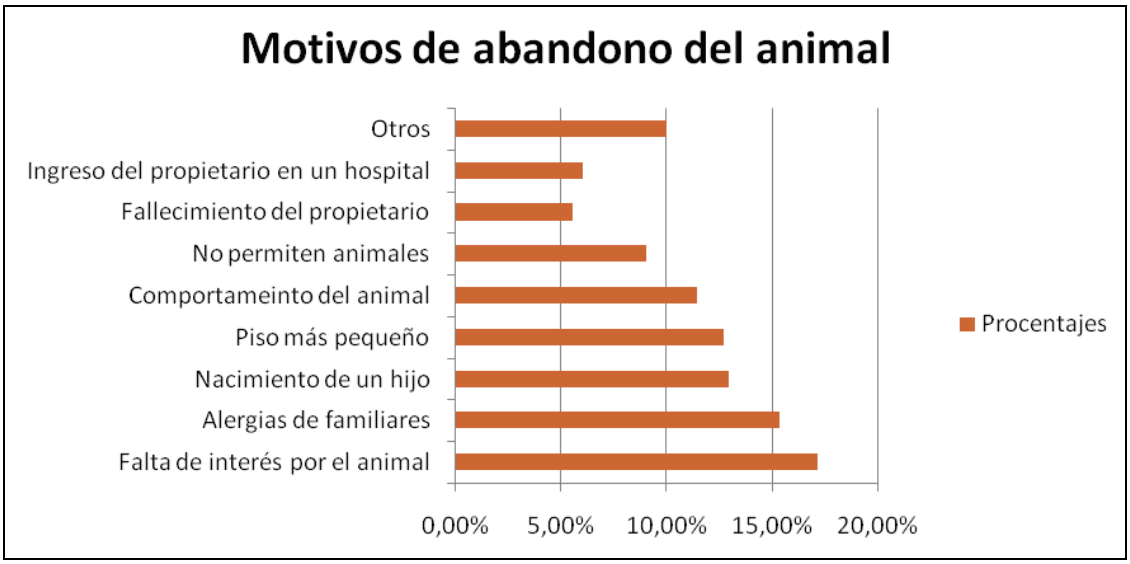

Figura 1-2: Motivos de abandono del animal.

Entro otros motivos habituales también hay que destacar que dueños irresponsables no se informan de la posibilidad de esterilizar a su mascota para evitar la aparición de camadas que o bien acabarán abandonados en la calle o si tienen un poco de suerte acabarán en una residencia canina donde si con el paso del tiempo nadie puede

hacerse cargo de estos animales, lamentablemente como son un gasto serán sacrificados.

Es evidente que ninguna razón puede llegar a ser la excusa del abandono del animal, únicamente dueños sin ética e irresponsables.

1.1 Consecuencias del abandono

El hecho que en España miles de animales sean abandonados por una parte de la sociedad obliga a numerosas sociedades y personas a dedicar tiempo a solucionar de alguna manera este problema y esto se ve afectado económicamente.

Podemos diferenciar dos tipos de consecuencias, sociales y para la salud del animal

Consecuencias sociales:

- Accidentes de tránsito
- Coste económico
- Suciedad de la vía pública(desecho con los que se alimentan, excrementos)
- Riesgo de agresiones a personas o bien otros animales

Consecuencias a los animales abandonados:

- Accidentes y muerte de muchos de estos animales
- Problemas psicológicos: Depresión, desorientación pérdida de confianza
- Desnutrición

Observando estos datos las personas a cargo de mascotas o bien que estén decididas a tener una mascota, inicialmente han de tener claro que es una gran responsabilidad que no pueden desechar de un día para otro. Que a la hora de conseguir una mascota deberían optar por ir a protectoras para adoptar su nueva mascota y se debe de tener muy presente que una solución no es enviar a nuestra mascota a una protectora porque puede dar la casualidad de que nadie la adopte y deban practicarle la eutanasia lamentablemente.

1.2 Estado del arte

Actualmente para dar soporte al problema, existen diferentes residencias, protectoras y hoteles de verano que pretenden evitar de algún modo que las personas abandonen a sus mascotas. Estas entidades como [1] y [2] ofrecen sus servicios mediante publicidad vía internet, televisiva y anuncios en carteles.

La mayoría de estas entidades disponen de un acceso vía aplicación web donde nos describen los servicios que ofrecen. Y existen aplicaciones que recogen estas entidades y las ponen al servicio del público dependiendo la provincia que la que se sitúen y facilita de alguna manera que un usuario pueda informarse rápidamente de residencias, veterinarios o hoteles cercanos de una manera cómoda y organizada.

Empresas como Facilísimo Interactive S.L [3] disponen del servicio de poder obtener por provincias y tipo de mascota, residencias cercanas.

Diferentes empresas físicas se encargan de publicitar o dejar publicar anuncios en sus aparadores para que personas de este ámbito puedan informarse sobre animales abandonados o bien sobre adopciones.

1.3 Estructura del documento

Un estudio de la situación actual de estos servicios de mascotas permitirá valorar de manera objetiva el problema con el abandono de animales e investigar diferentes soluciones del problema (Capitulo 2, p 11). Una vez estudiado la viabilidad de las posibles soluciones se trabajara con la propuesta más factible y se analizara detalladamente (Capitulo 3, p 25-29). Tras el análisis se estudiará el diseño de la propuesta (Capitulo 4, p 30-53) y finalmente se implementara (Capitulo 5, p 55-69).

2 Estudio de viabilidad

En esta sección se determina si el proyecto es viable. Consta de las siguientes partes:

- Descripción de la situación actual del problema donde se explica que tipos de servicio ofrecen otras aplicaciones web y qué tipo de servicio ofrecemos con este proyecto.
- Descripción del sistema que se quiere implementar para desarrollar el proyecto, requerimientos y organización del proyecto.
- Estudio de la inversión y la financiación del proyecto. Estimación de costes en la adquisición de equipos y mano de obra.

2.1 Descripción del proyecto

2.1.1 Situación actual

Aplicaciones como Mascotas Facilismo.com [3] permiten a usuarios poder encontrar sus residencias pero no de una manera muy organizada ya que no permite filtrar por zonas y la estructura de la web visualmente no es muy intuitiva ya que se compone de muchos accesos y es poco claro.

Por otro lado existen muchas aplicaciones dedicadas a un solo tipo de mascota como Planeta Can que se centran en el área canina y dan servicios como adopciones, protectoras, hoteles y artículos caninos.

El problema de estos servicios es que se centran únicamente en un animal y se podrían aun incluir más servicios respecto a este tipo de mascota. Muchas aplicaciones se dedican a anunciar entidades pero no permiten a usuarios poder anunciarse como particulares y poder informar sobre animales en adopción, sobre cuidados en verano no solo paseadores.

2.1.2 Objetivo

Se propone realizar una aplicación que permita anunciar residencias, hoteles, veterinarios, cruces de mascotas, mascotas perdidas y la posibilidad de comunicar personas particulares de una misma provincia que se encargue del cuidado de nuestra mascota durante un determinado intervalo de tiempo.

Con esta nueva opción, en esta situación de crisis, los dueños de mascotas que no vean una opción el hecho de dejar a su mascota en residencia por un asunto económico, tendrán la posibilidad de encontrar a una persona de su misma provincia y poder determinar un tipo de acuerdo económico que se adecue más a sus gastos. De este modo el dueño de la mascota tiene la opción de no estar atado a su mascota y poder obtener un servicio más económico y por el otro lado el cuidador tendrá la posibilidad de obtener un sueldo extra en estos momentos tan difíciles.

2.1.3 Entorno de desarrollo

En la investigación sobre que tecnologías que podrían ser útiles y necesarias para realizar el proyecto nos centramos en el patrón **Modelo Vista Controlador (MVC)**.
Este patrón de programación viene a ser antiguo pero actualmente se está poniendo de moda ya que es fácil y de flexible estructuración del código teniendo de manera independiente datos, implementación e interfaz.

Para el desarrollo de este patrón se ha utilizado **Struts** bajo la plataforma Java como herramienta de soporte para el desarrollo de la aplicación. Su carácter de software libre, la compatibilidad con todas las plataformas Java y el interés de desarrollar la aplicación con tecnología Java la hacen una herramienta potente. La información referente a esta herramienta la encontramos en [4].

La necesidad de una herramienta de persistencia que mapeara la sentencias SQL y de software libre hizo realizar un estudio sobre herramientas como **Hibernate** y **MyBatis**. El hecho de que **MyBatis** permitiera de una manera sencilla optimizar sentencias SQL, saber siempre lo que se está ejecutando en la base de datos y dado que es muy estable y facilita el hecho de encontrar donde surgen los problemas decantó para finalmente utilizar esta tecnología. La información referente a estas dos herramientas la encontramos en [5] y [6].

Finalmente para complementar estas dos herramientas **Struts** y **MyBatis**, se estudió la posibilidad de utilizar tecnología **Java Server Faces** o bien **Java Server Pages**.
Dada la experiencia utilizando estas dos tecnologías de programación obtuve que para el desarrollo más elaborado y más detallado el contenido **Java Server Faces** no era muy útil aunque para aplicaciones estándar de tablas era una herramienta muy sencilla de utilizar, y con Frameworks que hacían de ella una herramienta que permite al programador desarrollar de una manera muy rápida. Por otro lado **Java Server Pages** nos permite implementar una lógica de funcionalidad más compleja y es más amigable a la hora de realizar interfaces con CSS. En conclusión **Java Server Pages** permite la implementación de aplicaciones más dinámicas que **Java Server Faces**.

En el sistema de gestión de datos se investigó sobre dos bases de datos, **MySQL** o bien **Oracle**.

Visto que actualmente ambos gestores son de software libre, podemos encontrarnos con que la versión gratuita de **Oracle** es mucho más limitada que la de **MySQL**.
Oracle tiene el titulo de ser utilizado por grandes empresa y que **MySQL** no podría soportar esta magnitud de datos pero podemos ver que empresas como Facebook utilizan este gestor y tiene millones de usuario cada día haciendo peticiones a su base de datos. Finalmente, dado que la previsión de datos no va a ser a gran escala y que **MySQL** es un gestor más rápido, nos decantamos por esta herramienta para implementar nuestro servicio.

Finalmente para el entrono de desarrollo de la aplicación se propuso hacerla o bien en **NetBeans** o **Eclipse**. El hecho de que el **IDE NetBeans** permite una facilidad importante a la hora de crear rápidamente un esqueleto de una aplicación dada cualquier tipo de **MVC** y que su configuración es notablemente mas amigable y rápida que la **IDE Eclipse**, dio peso a la elección de la herramienta que utilizaría para el desarrollo del servicio. La información sobre tos entornos de desarrollo se pueden encontrar en los siguientes vínculos [7] y [8].

El servidor que se utiliza para el desarrollo de la aplicación es **GlassFish** dado que es más completo aunque menos flexible que un **Tomcat** ya que este último es más amigable pero se compone de únicamente un Web **container**, en cambio **GlassFish** contiene una colección Java EE donde uno de ellos es un **Tomcat**. Cualquiera de los dos sería una buena elección pero si se quiere desarrollar más aplicaciones en un futuro es aconsejable escoger un **GlassFish**.

2.2 Descripción del sistema a realizar

Análisis de Requerimientos

Requerimientos funcionales

- Acceder mediante un usuario y contraseña a la aplicación web.
- Consultar en la Base de Datos.
- Gestionar usuarios.
- Generar anuncios mediante las peticiones.

Requerimientos no funcionales

- Utilización de las herramientas que proporciona el software de GlassFish.
- La utilización de API de Facebook.
- Implementación de la aplicación en el entorno Windows.

2.2.1 Módulos del sistema

El sistema esta compuesto por tres módulos independientes que funcionan como ruedas dentadas ya que un modulo transmite datos al siguiente modulo para crear un flujo de funcionamiento (Fig.2-1). En el funcionamiento de la aplicación el usuario se comunica con la interfaz de usuario que realiza una acción. El controlador recibe la notificación de una acción del usuario y este se encarga de actualizar en el modelo de datos lo que corresponda a esa acción. Finalmente el controlador delega el objeto de datos a la interfaz. La vista recibe los datos del modelo y esta los presenta en la interfaz del usuario de forma que se represente los cambios que se han realizado en el modelo.

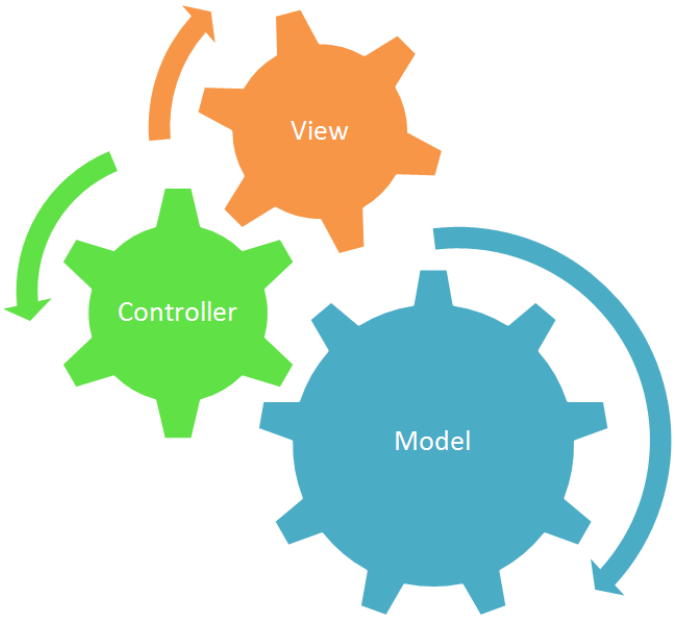

Figura 2-1: Módulos del sistema

2.2.2 Arquitectura del Proyecto

El **Modelo** es la representación específica de la información con la que el sistema opera. Se encarga de acceder a la capa de almacenamiento de datos, define las reglas de negocio. Lleva un registro de las vistas y controladores del sistema. En el caso de la aplicación, se encarga la herramienta MyBatis de gestionar mediante consultas los datos que fluyen por la aplicación hacia la base de datos

El **Controlador** es el encargado de gestionar las acciones del usuario e invocar peticiones al modelo y a la vista.

La herramienta Struts se encarga de gestionar las acciones del usuario. Llegada una acción de un usuario mediante la interfaz, este dirige el flujo hacia la acción siguiente.

La **Vista** presenta los datos que recibe del Modelo al usuario y puede encargarse del servicio de actualización para que lo invoque el controlador. En el caso de la aplicación, la interfaz esta implementada en HTML, JavaScript y Java Server Pages. Mediante las acciones que realiza el usuario sobre la interfaz, JSP utiliza sus herramientas para comunicarse con el controlador para que este tome las acciones pertinentes.

Utilizar este patrón de programación permite la posibilidad de tener varias vistas sin necesidad de modificar el modelo subyacente y permite un mecanismo de configuración de complementos complejos de una manera más tratable.

2.2.3 Recursos

Para el correcto funcionamiento de la aplicación podemos utilizar los siguientes navegadores como, IExplorer, Mozilla FireFox o Google Chrome.

Las tecnologías de programación con el que se ha desarrollado la aplicación es en Java Server Pages, HTML, JavaScript, Framework jQuery 1.7.1 y CSS para el diseño de la interfaz de la aplicación.

Aplicación utilizada para el diseño de la aplicación es Adobe PhotoShop CS5.

El hardware mínimo como requisitos es una Memoria RAM de 512MB con un procesador Pentium 4 3000Mhz. HDD de 30GB de disco duro. Monitor SVGA, Mouse, teclado, DVD-ROM, Tarjeta de red.

Vision Web Hosting [9] que se encargara de ofrecernos servicios de espacio, estabilidad y seguridad en la aplicación mediante las siguientes prestaciones:

- Espacio en disco 50GB
- Ancho de Banda Ilimitada
- 1 Dominio
- Soporte técnico 24h
- Servicio SMTP
- Spam & Virus protection
- Servidor GlassFish 3.0
- Java JDK 5.0 & 6.0
- JSP, Struts y Java

- SSH
- SSL Certificate
- Base de datos MySQL 5.0.67

2.3 Análisis de Coste-Beneficio

Dado que el objetivo de este proyecto es ofrecer un servicio gratuito a los usuarios, no se contempla ningún beneficio económico. No implica que en un futuro no se amplíe el proyecto y se desarrollen más servicios que proporcionen un beneficio.

En la implementación del proyecto podemos visualizar gastos por el software utilizado y gastos por el hardware utilizado.

Costes de Hardware/Software

Computadora de desarrollo	(*)350€
Dominio aplicación web	6€
Servidor Hosting	135€
Adobe PhotoshopCS5 (Free Trial)	0€
NetBeans6.9.1	0€
Framework jQuery1.7.1	0€
Total 1ªAño	**491€**
Total Anual	**141€**

*Gasto del primer año.

Figura 2-3: Tabla de costes

2.4 Evaluación de riesgos

Los riesgos que pueden surgir durante el desarrollo del proyecto son los siguientes:

- En la actualización de software como Internet Explorer o Mozilla pueden ocurrir riesgos de incompatibilidades de la aplicación web.
- La tecnología que utilicemos no proporcione les herramientas necesarias para el desarrollo del proyecto, esto provocaría realizar un nuevo estudio sobre el proyecto y la investigación de otra tecnología por lo tanto un incremento de tiempo en el proyecto.
- Un riesgo común es durante el diseño del a Base de Datos darse cuenta de que haría falta incluir algún factor más (p.e unos datos que no hemos tenido en cuenta al estudiar el diseño de la Base de Datos). Esto haría ralentizar el proyecto ya que tendríamos que modificar las tablas de la Base de Datos.

Los riesgos que podemos encontrar en un futuro son:

- Pretender mejorar el proyecto aportando nuevas ideas, y encontrar que la tecnología utilizada para el proyecto no puede proporcionar estas mejoras.
- Las actualizaciones de los lenguajes utilizados y de los navegadores no sean compatibles.

2.5 Alternativas

En el caso que alguna tecnología no ofreciera las herramientas que necesitemos para cumplir con los objetivos del proyecto buscaríamos otras tecnologías o lenguajes que nos las ofrecieran, previamente se haría un estudio de posibles candidatos.

Durante la decisión de tecnologías que formarían parte del proyecto se valoraron muchas otras que podrían ser compatibles con el objetivo que tiene la aplicación.

Para el caso de alternativas a Struts encontramos Frameworks como **Spring** y **Java Server Faces** que darían soporte al modelo MVC de la aplicación ya que son

Frameworks se software libre, agiles, fáciles de utilizar y se dispone de mucha información y guías de uso.

En el mapping de sentencias SQL tenemos como alternativa utilizar Hiberante que es un software también de software libre y existe mucha información sobre el uso del Framework.

Como lenguajes de desarrollo alternativo a Java Server Pages tenemos el propio lenguaje de Java Server Faces o bien JSTL.

Como alternativas a IDE NetBeans disponemos el Conocido IDE Eclipse.

Pero en caso de que fallasen estas alternativas, el proyecto bien se podría desarrollar en lenguaje PHP utilizando como MVC el Framework de Symphony lanzado en un servidor Apache y trabajando sobre un IDE como NetBeans.

2.6 Planificación del proyecto.

En la Fig.2-4 podemos observar que el proyecto está compuesto por tres etapas. El proyecto se inicia el 10 de Enero y se propone terminar el 12 de Mayo. El proyecto se desarrolla en seis meses, donde la primera se desarrolla en un tiempo de 29 días, la segunda etapa en 46 días y finalmente la tercera de publicar el proyecto se desarrolla en 6 días.

Nombre de tarea	Duración	Comienzo	Fin
Proyecto	**90 días**	**mar 10/01/12**	**vie 11/05/12**
Brainstorming de las funcionalidades	6 días	lun 16/01/12	lun 23/01/12
Investigación de tecnologías	**29 días**	**mar 24/01/12**	**vie 02/03/12**
Desarrollo del proyecto	**46 días**	**sáb 03/03/12**	**jue 03/05/12**
Release Proyecto	**6 días**	**vie 04/05/12**	**vie 11/05/12**

Figura 2-4: Tareas

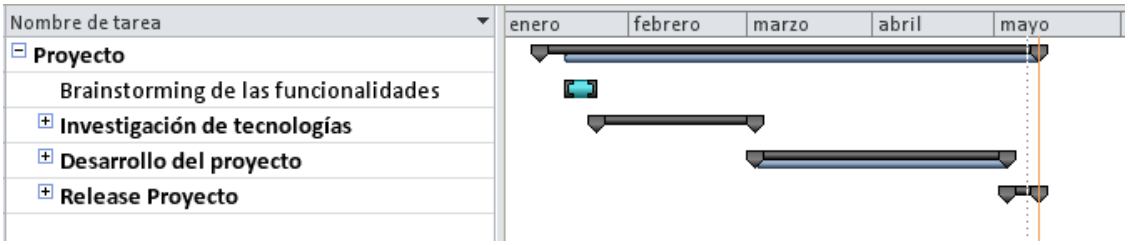

Figura 2-5: Diagrama de Gantt

Etapa 1: Investigación de tecnologías

Tareas

Nombre de tarea	Duración	Comienzo	Fin
Proyecto	**90 días**	**mar 10/01/12**	**vie 11/05/12**
Brainstorming de las funcionalidades	6 días	lun 16/01/12	lun 23/01/12
Investigación de tecnologías	**29 días**	**mar 24/01/12**	**vie 02/03/12**
Estudio de patrones MVC	4 días	mar 24/01/12	vie 27/01/12
Estudio de herramientas de mapping	4 días	lun 30/01/12	jue 02/02/12
Estudio del patron Struts	7 días	vie 03/02/12	lun 13/02/12
Estudio de la herramienta MyBatis	7 días	mar 14/02/12	mié 22/02/12
Estudio del lenguaje JavaServer Pages	7 días	jue 23/02/12	vie 02/03/12
Desarrollo del proyecto	**46 días**	**sáb 03/03/12**	**jue 03/05/12**
Release Proyecto	**6 días**	**vie 04/05/12**	**vie 11/05/12**

Figura 2-6: Etapa 1: Tareas

Diagrama de Gantt

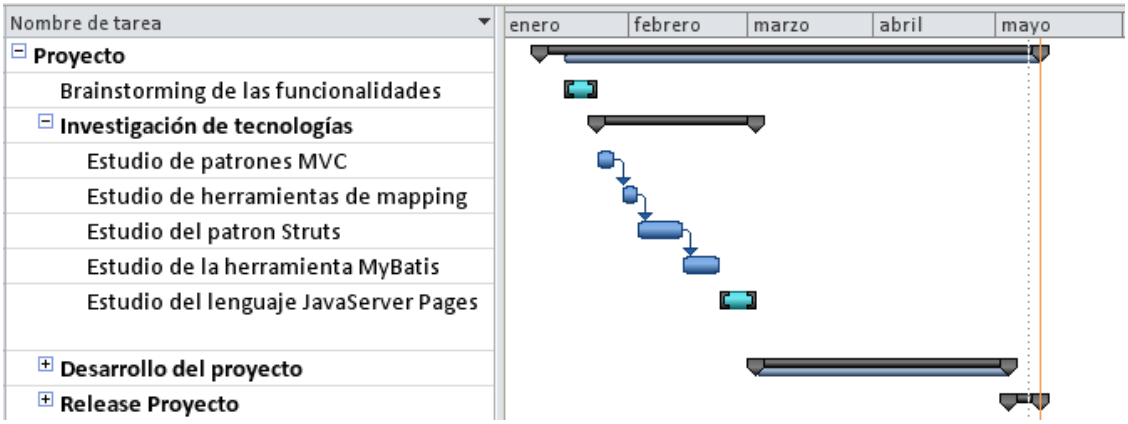

Figura 2-7: Etapa 1: Diagrama de Gantt

Etapa 2: Desarrollo del proyecto

Tareas

Nombre de tarea	Duración	Comienzo	Fin
Desarrollo del proyecto	**46 días**	**sáb 03/03/12**	**jue 03/05/12**
Generación de la Base de datos	**8 días**	**sáb 03/03/12**	**mié 14/03/12**
Estudio de la Base de datos	4 días	sáb 03/03/12	mié 07/03/12
Creación de la Base de datos	5 días	jue 08/03/12	mié 14/03/12
Estudio del diseño	**8 días**	**jue 15/03/12**	**lun 26/03/12**
Estudio de Photoshop CS5	4 días	jue 15/03/12	mar 20/03/12
Estudio de Illustrator CS4	4 días	mié 21/03/12	lun 26/03/12
Implementación	**29 días**	**mar 27/03/12**	**jue 03/05/12**
Home Page	**4 días**	**mar 27/03/12**	**vie 30/03/12**
Diseño de imágenes	3 días	mar 27/03/12	jue 29/03/12
Implementación	1 día	vie 30/03/12	vie 30/03/12
Crear usuarios y empresas	**6 días**	**sáb 31/03/12**	**vie 06/04/12**
Diseño de imágenes	3 días	sáb 31/03/12	mar 03/04/12
Implementación	3 días	mié 04/04/12	vie 06/04/12
Crear espacio personal de usuarios y empresas	**10 días**	**sáb 07/04/12**	**jue 19/04/12**
Diseño de imágenes	3 días	sáb 07/04/12	mar 10/04/12
Implementación	7 días	mié 11/04/12	jue 19/04/12
Crear visualización de anuncios	**6 días**	**vie 20/04/12**	**vie 27/04/12**
Diseño de imágenes	3 días	vie 20/04/12	mar 24/04/12
Implementación	3 días	mié 25/04/12	vie 27/04/12
Crear pantalla de Términos y contacta	**2 días**	**sáb 28/04/12**	**lun 30/04/12**
Diseño de imágenes	1 día	sáb 28/04/12	sáb 28/04/12
Implementación	1 día	dom 29/04/12	dom 29/04/12
Redes sociales	**4 días**	**lun 30/04/12**	**jue 03/05/12**
Estudio API de Facebook	1 día	lun 30/04/12	lun 30/04/12
Implementación	1 día	mar 01/05/12	mar 01/05/12
Desarrollo pagina de Facebook	2 días	mié 02/05/12	jue 03/05/12

Figura 2-8: Etapa 2: Tareas

Diagrama de Gantt

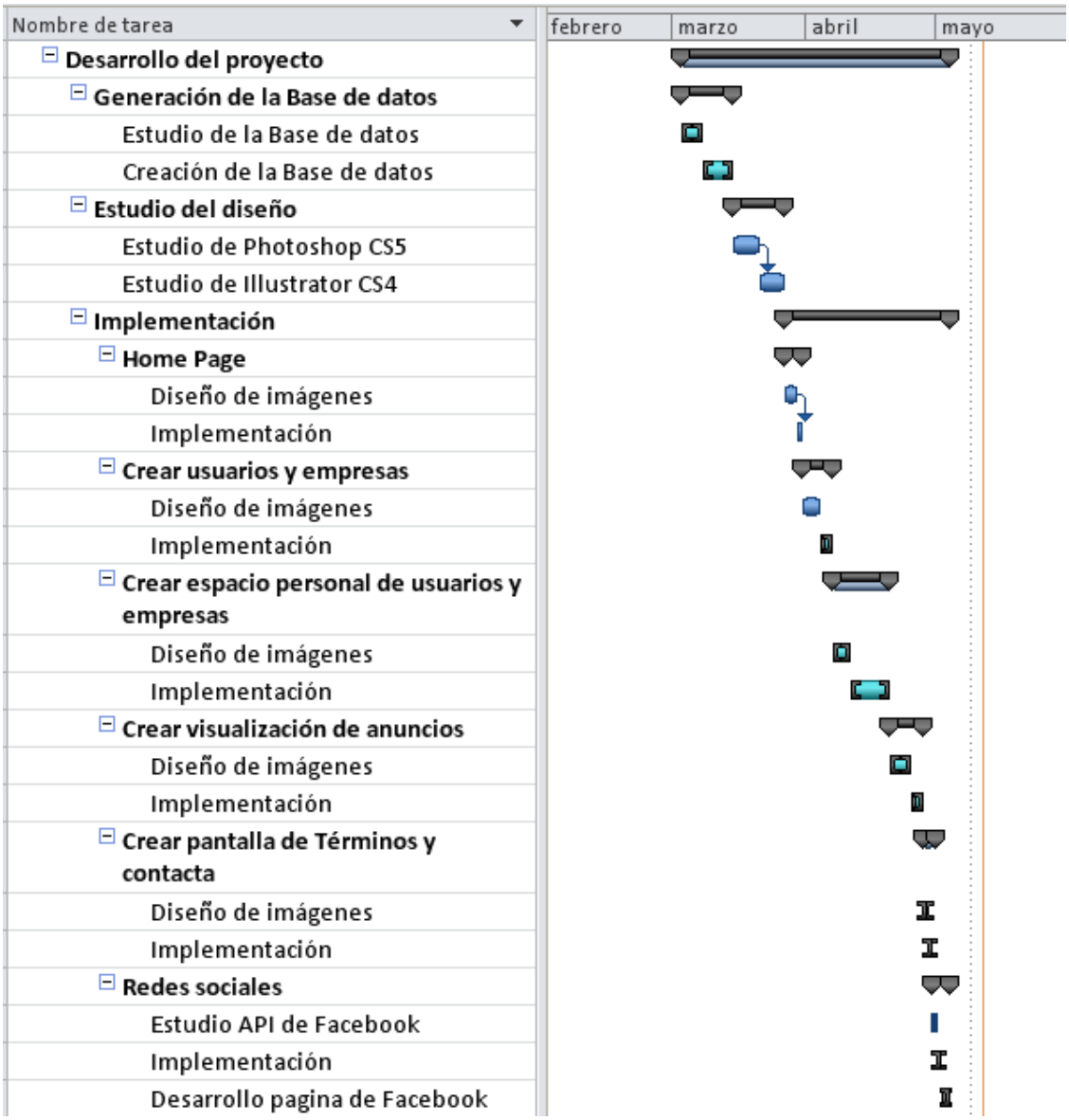

Figura 2-9: Etapa 2: Diagrama de Gantt

Etapa 3: Release del proyecto

Tareas

Nombre de tarea	Duración	Comienzo	Fin
Proyecto	**90 días**	**mar 10/01/12**	**vie 11/05/12**
Brainstorming de las funcionalidades	6 días	lun 16/01/12	lun 23/01/12
Investigación de tecnologías	**29 días**	**mar 24/01/12**	**vie 02/03/12**
Desarrollo del proyecto	**46 días**	**sáb 03/03/12**	**jue 03/05/12**
Release Proyecto	**6 días**	**vie 04/05/12**	**vie 11/05/12**
Estudio de un Hosting	3 días	vie 04/05/12	mar 08/05/12
Configuración	3 días	mié 09/05/12	vie 11/05/12

Figura 2-10: Etapa 3: Tareas

Diagrama de Gantt

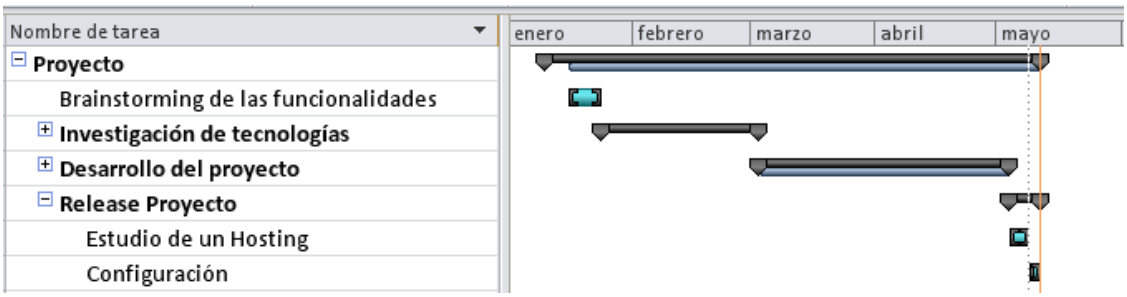

Figura 2-11: Etapa 3: Diagrama de Gantt

2.7 Conclusiones

De acuerdo con los objetivos del proyecto, se ha previsto desarrollar la aplicación en seis meses. El proyecto se ha estructurado en 3 etapas, en la primera se ha realizado un estudio de las tecnologías adecuadas para implementar la aplicación. Se han investigado patrones que formen parte de la base del proyecto y herramientas que complementen a este patrón.

La segunda etapa trata de la implementación del código del programa y el diseño de la interfaz de usuario. Para ello esta segunda etapa se ha dividido en tres sub etapas

donde encontramos la creación de la Base de datos el diseño de la interfaz y la implementación. Así mismo la implementación se divide en seis sub etapas de las cuales se encuentra cada pantalla que encontrara el usuario navegando por la aplicación junto a las redes sociales en las que está registrada la aplicación.

3 Análisis

Una aplicación web es una aplicación software que es codificado en un lenguaje mediante los navegadores web (HTML, JavaScript, Java, asp.net, etc.) en la que se confía la ejecución al navegador.

Estas aplicaciones web permiten ser actualizadas de manera ágil y permite a los usuarios interactuar de manera sencilla y rápida con los servicios que las aplicaciones les ofrecen.

Existen numerosos??? de lenguajes de programación para el desarrollo de aplicaciones web, estos son algunos de los más utilizados:

- HTML 5
- PHP
- Microsoft .NET
- JAVA
- Ruby & Rails

La estructura de una aplicación web consta de tres capas. En la primera capa se encuentra el navegador web que interpreta lenguajes dados como HTML o bien Javascript y un motor capaz de interpretar lenguajes como JavaServer Pages o bien ASP.Net es la segunda capa de la aplicación. Finalmente en la tercera capa encontramos la base de datos donde podemos encontrar tipos como Oracle, MySQL o bien PostgresSQL.

Una vez determinada la tecnología que va implementar la aplicación que queremos desarrollar, es necesario tener una visión del flujo que tendrá nuestro servicio, mediante la navegación de un usuario a través de la interfaz web.

3.1 Casos de uso

A continuación cada caso de uso proporciona escenarios que indican cómo debería interactuar el sistema con el usuario.

Usuario no registrado: Este perfil tiene como funcionalidades poder buscar residencias para su mascota, buscar un cuidador de su ciudad, informarse sobre animales perdidos, adoptar mascotas abandonadas y buscar mascota para cruzar con la suya propia.

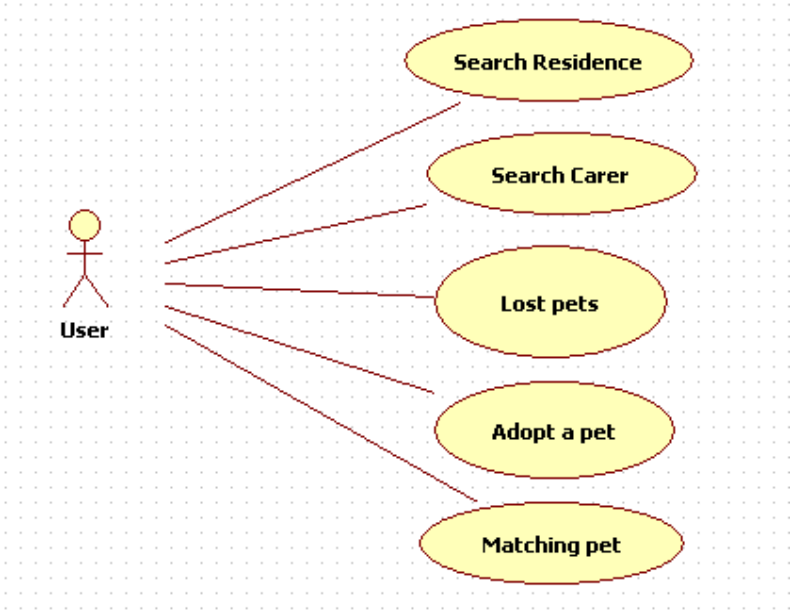

Figura 3-1: Caso-Uso:usuario

A parte de los casos de uso de búsqueda, un usuario no registrado tiene opción de registrarse en la aplicación para disponer de mas servicios, contactar con el administrador de la aplicación, añadir la aplicación a favoritos y unirse a la red de facebook de la aplicación.

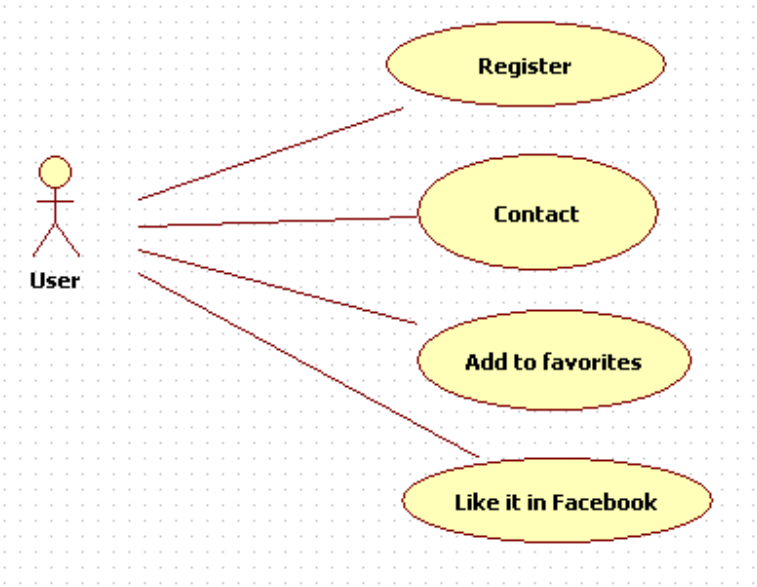

Figura 3-2: Caso-Uso: Usuario

Como usuario no registrado tiene dos opciones a la hora de formar parte de la aplicación que son registrarse como empresa o bien como particular.

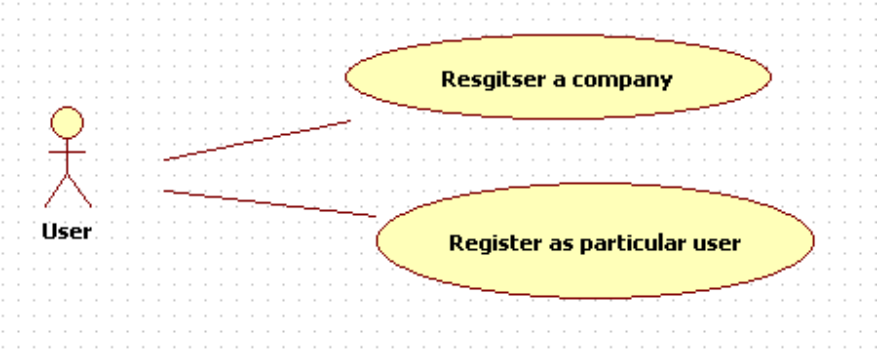

Figura 3-3: Caso-Uso: Usuario

Usuario registrado: Un perfil de usuario registrado dispone de los casos de uso de un **usuario no registrado** y además puede anunciarse en la aplicación. Puede gestionar su perfil de usuario, ver y gestionar los anuncios creados por él y darse de baja de la aplicación.

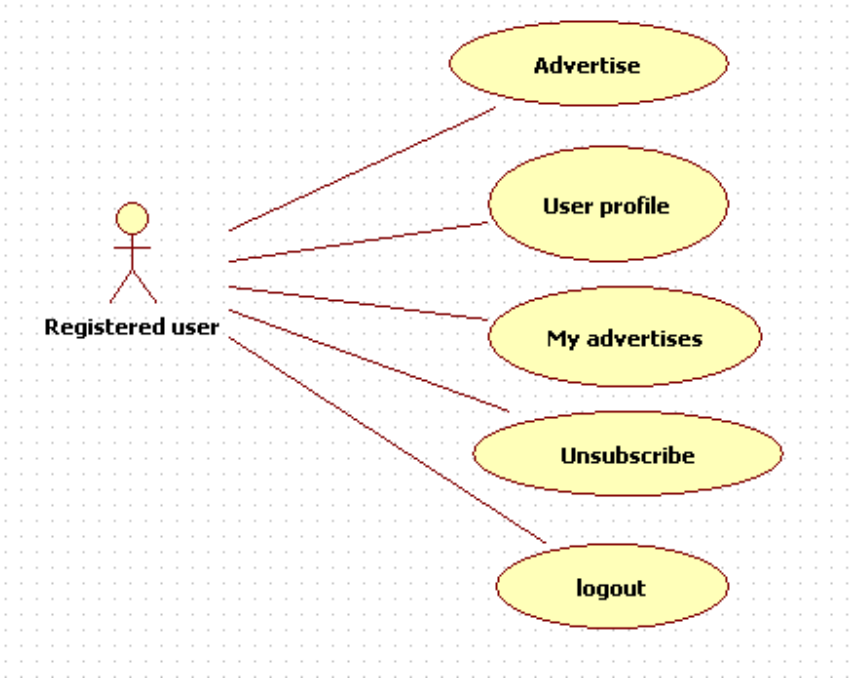

Figura 3-4: Caso-Uso: Usuario Registrado

3.2 Resumen

Una vez analizado el proyecto que se quería desarrollar, se analizaron los casos de uso que debería incluir el servicio, el tipo de usuarios que podrían acceder a la aplicación y la manera de estructurar el proyecto.

Después de un estudio sobre los patrones a utilizar para la aplicación y la tecnología de programación con la cual se desarrollaría el proyecto se determino en utilizar Struts como patrón de MVC complementado con MyBatis para el mapeo de las consultas SQL. El lenguaje de programación se ha utilizado Java Server Pages, HTML, JavaScript, jQuery. Como base de datos MySQL.

Factores que determinaron la elección son la experiencia obtenida en estas tecnologías mencionadas y la necesidad de dominar cada una de ellas de manera profesional de cara a un futuro. Factores importantes que dieron pie a la elección fueron las ventajas que tenían estas tecnologías y herramientas frente a las alternativas que podrían también haber sido parte de la aplicación.

4 Diseño

Después de realizar un estudio de los objetivos que tiene que cumplir la aplicación y después de investigar sobre el software del que se quería utilizar por motivos de conocimientos y familiarización, se analizo la manera de implementar un software que abordara cada uno de los objetivos con el fin de facilitar al usuario el servicio.

Una vez conocido el funcionamiento que tendría el software, se han estudiado para cada pantalla diferentes maneras de distribuir la información y de hacer más amigable el flujo entre pantallas. Para ello se ha utilizado Photoshop CS5 para crear diferente borradores de las vistas de cada pantalla.

4.1 Diseño de la aplicación Web

4.1.1 Diseño de la Base de datos

Previamente al diseño de la aplicación WEB se ha estudiado y propuesto diferentes estructuras y clases que contendrá la Base de Datos.

Se ha tenido en cuenta que no únicamente usuarios particulares querrían formar parte del servicio, sino que también empresas emprendedoras podrían animarse a ofrecer sus servicios en esta aplicación. Estas clases se construyeron después de estudiar cómo sería más lógico la clasificación de datos y como deberían estar relacionadas las clases entre si y sus dependencias.

Se introdujeron clases como por ejemplo la del usuario para poder gestionar el acceso de diferentes usuarios o empresas. Otra característica de la primera versión de la Base de datos fue crear una clase para diferenciar entre usuario y empresa con el objetivo de no tener que recorrer tantos registros mediante una consulta, pero finalmente por temas logísticos se ha determinado que se unificarían estos dos tipos de usuarios en una misma entidad. Finalmente se crearon doce entidades.

En la entidad usuario disponemos de toda la información que el usuario debe de introducir para poder describirse y autentificarse en la aplicación, esta entidad es la misma para empresa y para particulares.

En la entidad anuncios_empresa y anuncios_usuario se contiene la información sobre el titulo, el contenido y la imagen del determinado anuncio, y el identificador que hace referencia al usuario que lo ha publicado.

Seguidamente disponemos de tablas como imágenes_perfil_usuario, imágenes_anuncio_usuario y imágenes_anuncio_empresa, que contienen el path de la imagen asociada al anuncio.

Se han creado tablas como ciudades, tipo de animal, tipo de empresa para poder tener múltiples opciones al crear un anuncio ya que podemos tener cualquier animal al que queremos anunciar, disponemos de todas las provincias de España ahora anunciarnos y disponemos de varios tipos de empresas que ofrecen sus servicios.

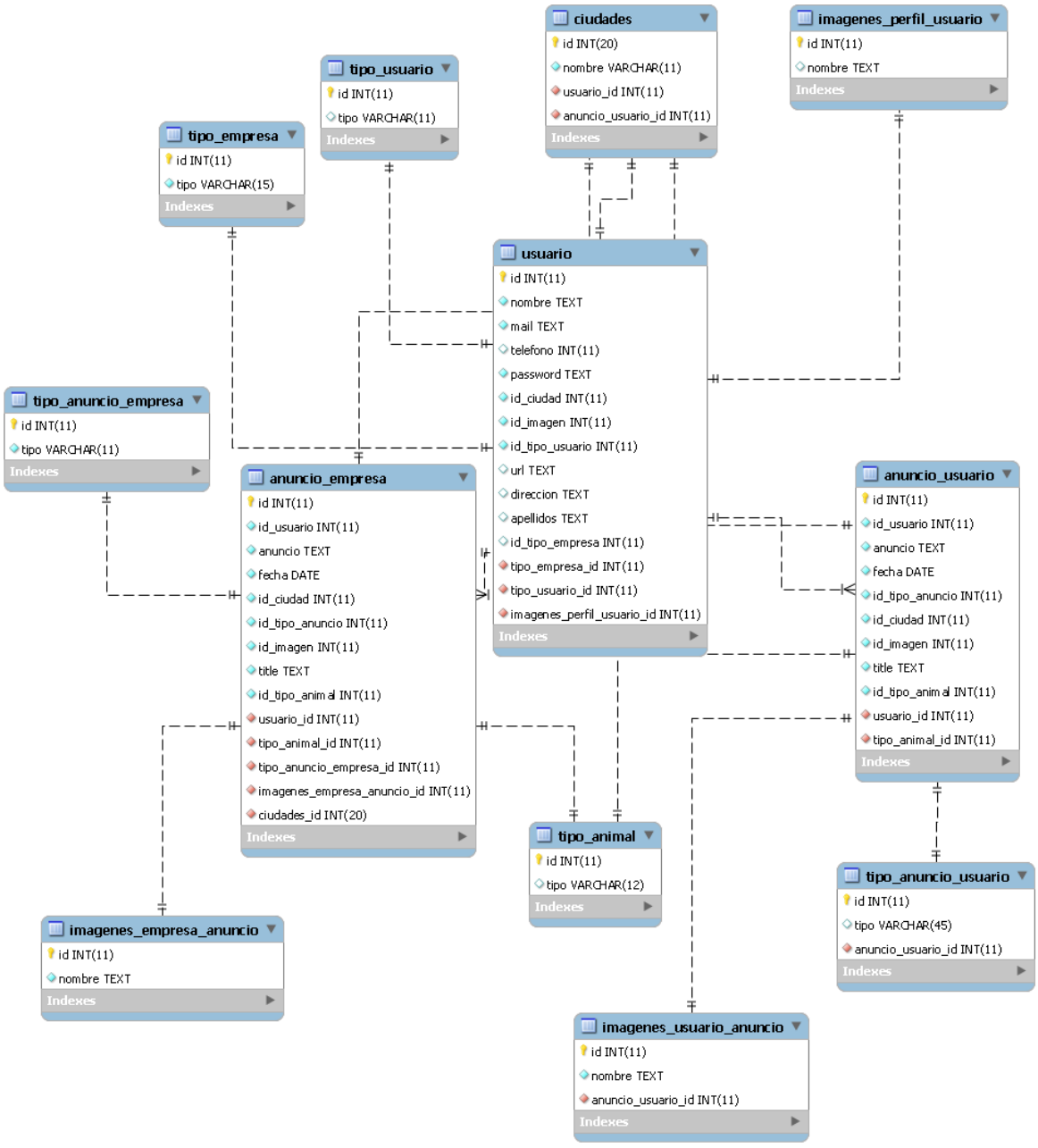

Figura 4-1: Estructura de la Base de datos

4.1.2 Diseño del Modelo Vista Controlador (MVC)

4.1.2.1 Flujo del funcionamiento de Struts

En la Fig. 4-2, se muestra el flujo que realiza la arquitectura Struts internamente mediante sus ficheros de configuración (**web.xaml** y **struts-config.xml**) y las acciones realizadas por el usuario a través de la aplicación web que se comentan en el apartado 4.1.2.2.

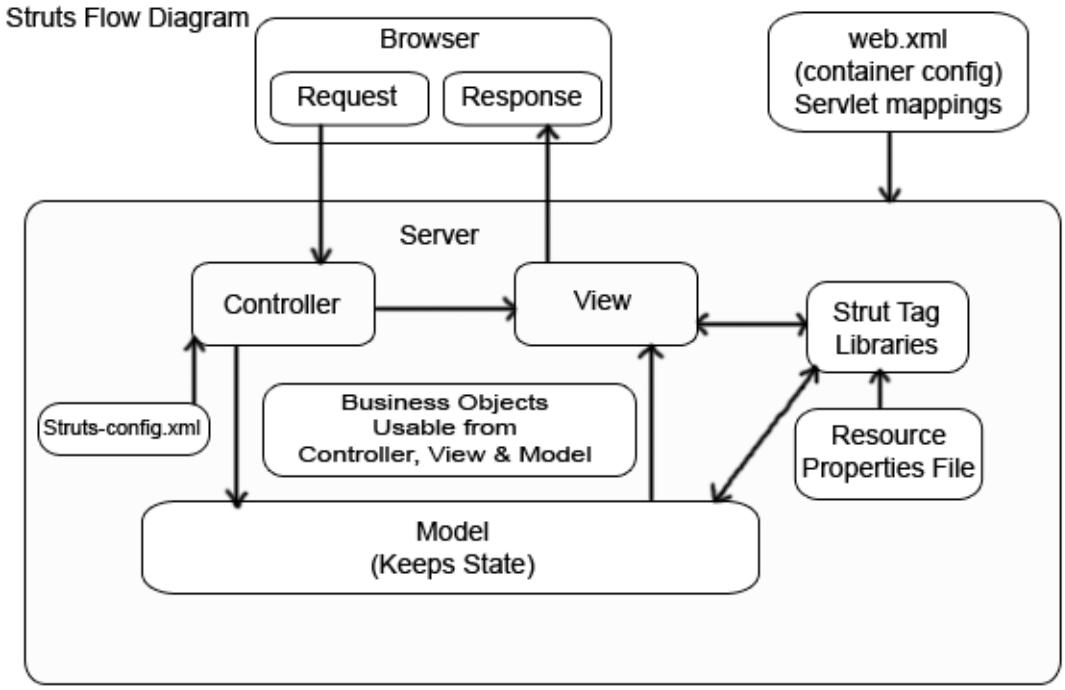

Figura 4-2: Diagrama de flujo de Struts

Como se observa en la Fig. 4-2, el flujo de Struts consta de dos partes principales, el **Navegador Web (Browser)** y el **Servidor**.

El **Navegador Web** es cualquier explorador web como Internet Explorer, Google Chrome o bien Mozilla Firefox, y en esta arquitectura se comunica con el Controlador para gestionar las peticiones que realiza el usuario. Una vez procesada la petición del usuario, la Vista es la encargada de comunicarse de nuevo con el Navegador para mostrar los resultados al usuario de la petición que ha realizado previamente.

En el modulo del **Servidor**, se encuentra alojado el patrón MVC. Como se puede observar en la Fig. 4-2, el patrón MVC contiene un archivo de configuración de Servlets (FIg.4-3) donde se configura n los parámetros iníciales del patrón, el directorio donde se alojan las acciones del controlador, y los Servlet-Mapping.

El Navegador Web se comunica con el Controlador para realizar las peticiones del usuario. El Controlador contiene un fichero de configuración XML (apartado 4.1.2.2) donde se determinan las acciones que debe realizar el controlador para obtener el resultado a la petición que le ha entrado por parte del usuario. Una vez determinada la acción que debe realizar, se comunica con el Modelo (apartado 4.1.2.3) para consultar en la Base de datos la petición recibida y esta se comunica con la Vista que finalmente muestra el resultado al usuario mediante el Navegador Web.

```
<servlet>
    <servlet-name>action</servlet-name>
    <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
    <init-param>
        <param-name>config</param-name>
        <param-value>/WEB-INF/struts-config.xml</param-value>
    </init-param>
    <init-param>
        <param-name>debug</param-name>
        <param-value>2</param-value>
    </init-param>
    <init-param>
        <param-name>detail</param-name>
        <param-value>2</param-value>
    </init-param>
    <load-on-startup>2</load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>action</servlet-name>
    <url-pattern>*.do</url-pattern>
</servlet-mapping>
<session-config>
    <session-timeout>
        30
    </session-timeout>
</session-config>
<welcome-file-list>
    <welcome-file>home.jsp</welcome-file>
</welcome-file-list>
```

Figura 4-3: Configuración web.xml

4.1.2.2 Estructura de archivos

Para realizar el proyecto, se ha utilizado el patrón MVC mediante el IDE NetBeans 6.9.1 que construye la siguiente estructura de ficheros (Fig.4-4).

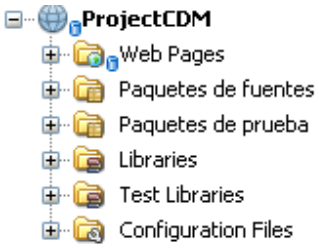

Figura 4-4: Estructura de ficheros MVC

Para la utilización del patrón MVC son necesarias un conjunto de librerías. NetBeans por defecto te crea las referencias a las librerías de Struts, pero además para este proyecto se ha utilizado las librerías de la herramienta de persistencia MyBatis (apartado 4.1.2.4) y un servidor GlassFish Server 3 (Fig. 4-7).

Una vez referenciadas todas las librerías necesarias para el correcto funcionamiento del patrón MVC mediante Struts, es necesario realizar las configuraciones pertinentes del patrón mediante los ficheros de configuración web.xml y struts-config.xml, comentados en el apartado 4.1.2.1, situados dentro del directorio Web Pages/WEB-INF (Fig.4-5).

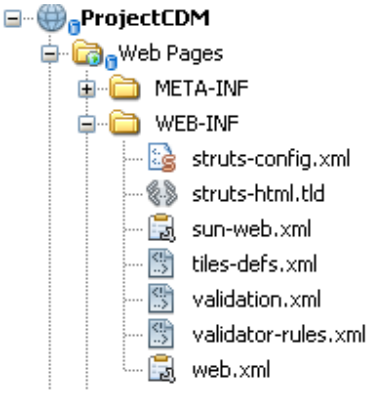

Figura 4-5: Directorio de configuración WEB-INF

Para la creación del proyecto se han creado un conjunto de **Java server Pages** que es el modulo Vista desde el cual el usuario recibe los resultados de las peticiones y

desde donde puede realizar sus peticiones (Fig. 4-6). Como podemos observar todas las vistas de usuario se recogen en el directorio Web Pages.

Figura 4-6: Vistas JSP

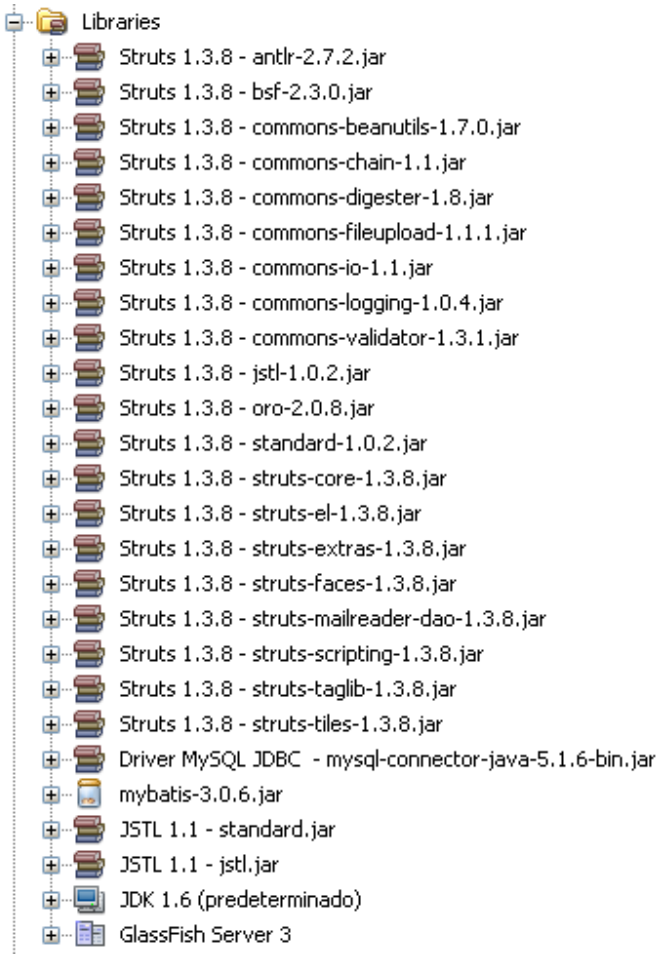

Figura 4-7: Referencia de librerías

En la Fig.4-8 muestra los Paquetes de fuentes, que contienen un seguido de paquetes que realizan las acciones que el controlador a ha pedido, el paquete DB que contiene la conexión entre la base de datos y la herramienta de persistencia MyBatis (apartado 4.1.2.4) y un archivo XML que contiene las sentencias SQL que se realizan durante la utilización de la aplicación web.

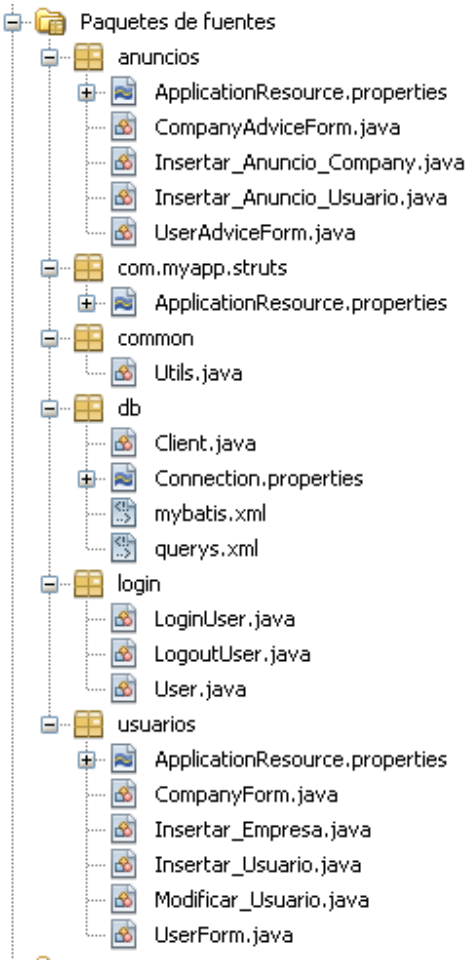

Figura 4-8: Paquetes de fuentes

4.1.2.3 Arquitectura Struts

El framework de Struts tiene tres objetos clave:

- **Request Handler** que permite mapear un URI particular.
- **Response Handler** que es el encargado de dar el control a otro recurso que permita terminar la petición del usuario.
- **Tag library** que permite al desarrollador generar formularios interactivos basados en server pages.

Cuando un usuario realice una petición, esta petición es controlada por el **Struts Action Servlet**. Cuando el Action Servlet recibe la petición, intercepta la URL y dependiendo e nuestro fichero de configuración, el Action Servlet envía la petición a un Action Class. Este Action Class es parte del controlador y es el encargado de completar la petición del usuario y comunicarse con la capa de modelo.

El modelo interactúa en Back-End contra la Base de datos que finalmente devuelve los datos que ha pedido el usuario y se comunica con el controlador de la vista que en este caso es una presentación en Java Server Pages (JSP).

En una aplicación Struts encontramos otro fichero XML de configuración llamado **struts-config.xml**. En este fichero están configuradas todas las acciones que debe realizar la aplicación dependiendo de la petición realizada por el usuario. El fichero se utiliza para asociar los componentes de los controladores con los path (i.e **<action path ="/login" type = "LoginAction">**). Este tag informa al ActionServlet que cuando reciba una URL con extensión **/login.do** invoque al controlador **LoginAction**.

Por cada tag action contiene la propiedad **forward** que permite dependiendo del resultado que devuelva el action del controlador, realizar una u otra acción. En caso de que el resultado del la acción sea **Success**, puedes visualizar una vista o si bien es **Failure**, presentar al usuario otra vista, Fig.4-9.

```
<action path="/LoginUser" type="login.LoginUser">
      <forward name="success" path="/espacio_Personal.jsp" />
      <forward name="failure" path="/login_error.jsp" />
</action>
```

Figura 4-9: Tag Action

Normalmente un Action está asociado a un JavaBean en el fichero de configuración de Struts. Esta clases Java se encarga de invocar el método valídate del ActionForm, Fig.4-10.

```
<form-beans>
   <form-bean name="UserForm" type="usuarios.UserForm"/>
   <form-bean name="CompanyForm" type="usuarios.CompanyForm"/>
   <form-bean name="UserAdviceForm" type="anuncios.UserAdviceForm"/>
   <form-bean name="CompanyAdviceForm" type="anuncios.CompanyAdviceForm"/>
</form-beans>
```

Figura 4-10: Form Beans

En la clase UserForm que extiende a ActionForm, tenemos las propiedades definidas para esta clase y la implementación del método valídate() que se encarga de validar los campos que introduce el usuario para esta clase (Fig. 4-11).

```
public class UserForm extends org.apache.struts.action.ActionForm {

    private String message = "";
    private String name = "";
    private String surname = "";
    private String mail = "";
    private String password = "";
    private String cpassword = "";
    private String city = "";
    private String phone = "";
    private FormFile file;
    private String check = "";
    private static String imagesPath;
```

Figura 4-11 JavaBean UserForm

Struts dispone de tag libraries que permiten interactuar con el controlador que tenemos definido. En la Fig. 4-12, observamos que este **<html:form>** hace referencia a la clase **UserForm**. Por lo tanto una vez realizado el submit de este formulario, automáticamente se rellenaran los campos que contiene la clase UserForm con la información cumplimentada a través de este formulario.

```
<html:form action="Insertar_Usuario" method="post" enctype="multipart/form-data">
    <table>
        <table align="left" style="padding-right: 80px;padding-top: 40px;">
            <tr>
                <td>
                    Nombre<br/>
                    <html:text name="UserForm" property="name"/>
                </td>
                <td style="padding-left: 30px">
                    Apellidos<br/>
                    <html:text name="UserForm" property="surname"/>
                </td>
            </tr>
            <tr>
                <td >
                    Correo electrónico<br/>
                    <html:text name="UserForm" property="mail"/>
                </td>
            </tr>
```

Figura 4-12: Tag library Insertar usuario

4.1.2.4 Herramienta de persistencia MyBatis

A diferencia de un ORM (Mapeo Objeto-relacional), esta herramienta se encarga de mapear métodos a sentencias SQL. Estas sentencias pueden ser almacenadas mediante anotaciones o bien en un fichero XML. En este caso se ha utilizado ficheros XML.

MyBatis contiene un fichero de configuración XML donde se introduce los parámetros necesarios para la conexión a la base de datos y el fichero XML donde están alojadas las sentencias SQL (Fig. 4-13). En este caso el fichero **query.xml** contiene todas las sentencias SQL de las que dispone la aplicación (Fig.4-14) y están identificados mediante un ID para que desde la vista JSP se pueda determinar que sentencia ejecutar.

Una vez configurada la herramienta MyBatis, tenemos una clase cliente que contiene un método donde se definen los parámetros necesarios para comunicarse con las sentencias SQL. De esta manera desde la vista JSP se realizan llamadas a la clase cliente pasando por parámetro el nombre que hace referencia al sentencia SQL que se quiere ejecutar contra la base de datos.

```
<configuration>
    <properties resource="db/Connection.properties" />
    <environments default="development">
        <environment id="development">
            <transactionManager type="JDBC"/>
            <dataSource type="POOLED">
                <property name="driver" value="${driver}"/>
                <property name="url" value="${url}"/>
                <property name="username" value="${user}"/>
                <property name="password" value="${pass}"/>
            </dataSource>
        </environment>
    </environments>
    <mappers>
        <mapper resource="db/querys.xml"/>
    </mappers>
</configuration>
```

Figura 4-13: Configuración MyBatis

```
    <mapper namespace="querys">
        <select id="selectUsers" resultType="Map">
            select
                id,
                nombre,
                apellidos
            from usuario
        </select>
        <select id="selectAnimalType" resultType="Map">
            select
                id,
                tipo
            from tipo_animal
        </select>
        <select id="selectCities" resultType="Map">
            select
                id,
                nombre
            from ciudades
        </select>
```

Figura 4-14: Sentencias SQL

4.2 Diseño de la interfaz WEB

Previamente al diseño de la interfaz web, se estudió los tipos de usuario que existirían en la aplicación web y que funciones podrían realizar cada uno de los perfiles. Los perfiles de usuario son los siguientes:

Perfiles	Buscar Residencia	Buscar cuidador	Buscar apareador	Adoptar Mascota	Mascotas perdidas	Anunciar
Empresa	✓	✓	✓	✓	✓	✓
Usuario	✓	✓	✓	✓	✓	✓
Usuario /Empresa No registrado	✓	✓	✓	✓	✓	✗

Figura 4-15: Tabla de privilegios

Una vez diseñada la Base de datos, definido los perfiles de usuario y teniendo conocimiento del trabajo que ha de realizar la aplicación web, se ha estudiado la estructura, Fig.4-16, que ha de seguir la nueva aplicación.

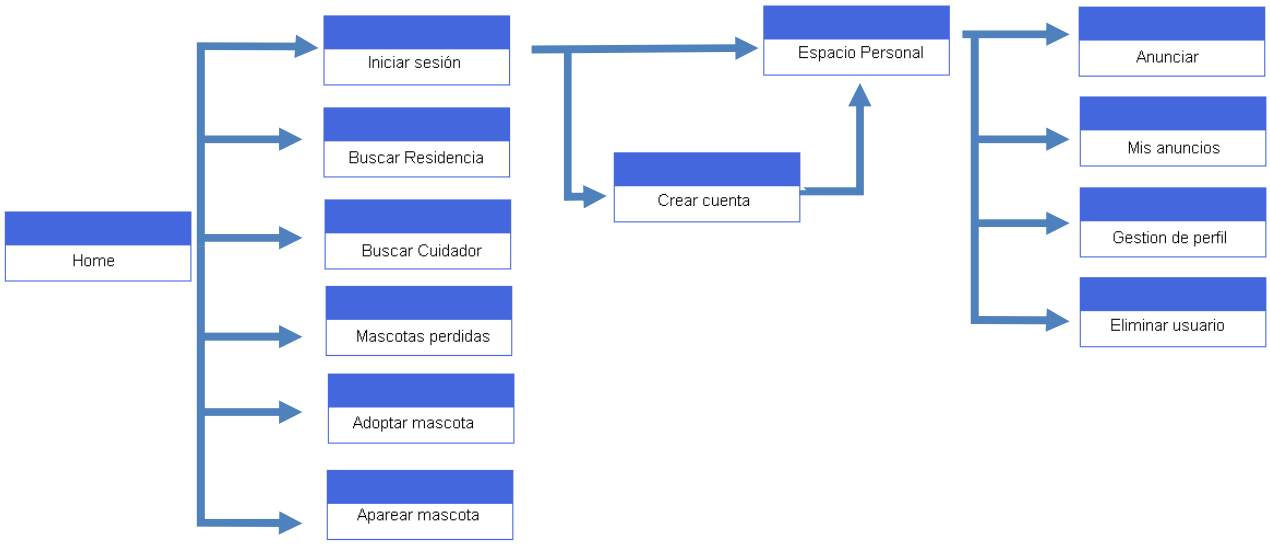

Figura 4-16: Esquema de la interfaz de usuario

En el esquema de la interfaz usuario podemos observar las diferentes opciones que dispone el usuario una vez accedida a aplicación. Una vez dentro de la aplicación el

usuario se encuentra la página principal (**Home**) donde se explica brevemente el servicio y donde nos encontramos el menú y la visualización de un grupo de mascotas que puede adoptar. En esta pantalla el usuario tiene también la posibilidad de unirse al grupo de Facebook de este servicio.

Una vez introducido al usuario dentro del contexto de la aplicación se le ofrece navegar por diferentes servicios:

- **Buscar Residencia** en su provincia
- **Buscar cuidadores** en su provincia
- Ver **anuncios mascotas perdidas**
- Ver **anuncios para adoptar una mascota**
- Ver anuncios de dueños que desean **aparear a su mascota**

Una vez dentro de la opción **Buscar Residencia**, al usuario se le mostrara una vista donde podrá ver como elemento principal un seguido de anuncios que se componen por un titulo, un contenido, una imagen asociada, la provincia de donde se realiza el anuncio y la fecha de creación del anuncio. Dentro de este anuncio encontramos un acceso directo al **detalle del anuncio** donde se encuentra el anuncio en si con la información del anunciante. En esta sub pantalla el usuario puede tener a su disposición el teléfono o correo electrónico del anunciante y así poderse poner en contacto con él.

Volviendo a la pantalla de **Buscar Residencia** el usuario puede ver un seguido de filtros para buscar entre los muchos anuncios de la aplicación, el usuario puede filtrar por provincia y mascota.

Desde esta pantalla el usuario puede navegar a otros menús como **Buscar cuidadores**, ver **anuncios de mascotas perdidas**, ver **anuncio para adoptar mascota** y ver **anuncios para cruzar sus mascotas**. Estas pantallas que vera el usuario tienen la misma mecánica que la explicada anterior mente en **Buscar Residencia.**

Todas estos servicios ofrecen en sus pantallas la posibilidad de permitir al usuario **iniciar sesión** o bien **anunciarse**. Para ello si el usuario quiere **iniciar sesión** navegara hacia una pantalla donde se le pedirá que ingrese su usuario y contraseña, en caso de olvidarse se le proporcionara dicha información por correo electrónico. En Caso de ingresar correctamente se le dirigirá a su **espacio personal**. Si el usuario no dispone de cuenta de la aplicación se le dispone de un enlace a una pantalla donde se le pedirá **registrarse** como empresa o usuario.

Una vez accede a su cuenta, se encontrará en su espacio personal donde dispone de opciones para administrar su **perfil de usuario** y **sus anuncios**. Desde el espacio personal es desde donde el usuario podrá **crear sus anuncios** a través de un formulario y finalmente publicarlo.

Todas estas pantallas comparten los vínculos para poder ver los Términos legales del servicio, contactar con el administrador por correo electrónico o bien añadir el servicio sus favoritos.

4.2.1 Propuesta de diseño

Inicialmente se propuso que directamente se mostrara el menú de buscar residencia sin tener una página inicial, pero finalmente se optó por tener una página donde explicar al usuario lo que iba a encontrar dentro de la aplicación.

Se puede observar que en esta página principal no hay posibilidad aun de anunciarse ni de iniciar sesión ya que se utiliza únicamente como entrada a la aplicación y como introducción informativa.

La página esta dividida entre los menús principales que son loe menús del 1 al 5 y los menús de contacto y de información que se sitúan en la parte inferior de la aplicación siendo los menús del 6 al 8, representación en la Fig.4-17.

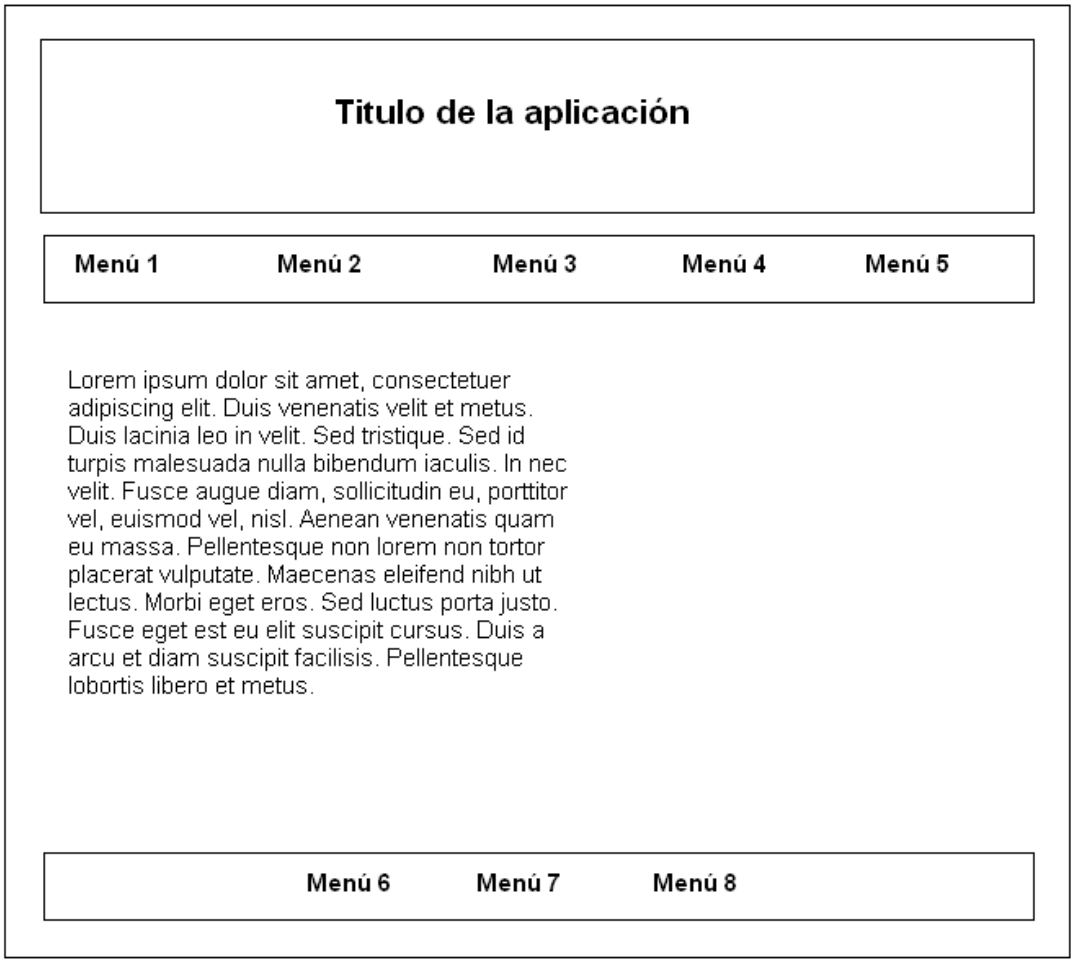

Figura 4-17: Propuesta página principal

Una vez seleccionado unos de los menús situados en la franja superior de la aplicación que se componen entre los menús del 1 al 5, nos dirigimos a la siguiente pantalla.

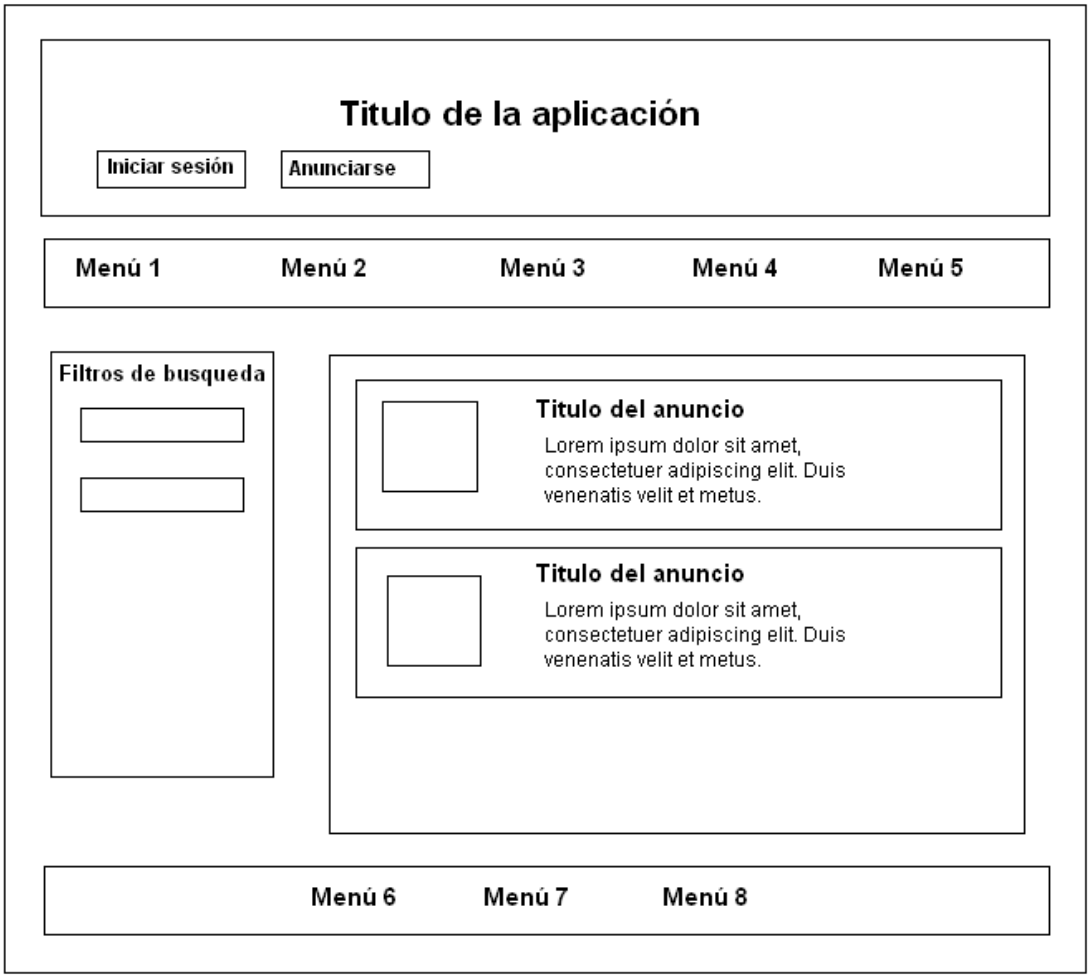

Figura 4-18: Propuesta de menú de la interfaz

En la Fig.4-18, se muestra la propuesta de la página de vista de anuncios. Desde un principio ya se tuvo bastante claro esta distribución ya que nos basamos en diferentes aplicaciones existentes. Se ha diseñado una pantalla que permita al usuario interpretar de manera simple y sencilla, las acciones que puede realizar. Esta plantilla de pantalla es la sigue la misma estructura para los Menús 2 al 5 ya que se pretende que el usuario no tenga una impresión muy grande en la transición de pantallas.

En la Fig. 4-19, se muestra como se propuso hacer la vista de iniciar la sesión de un usuario.

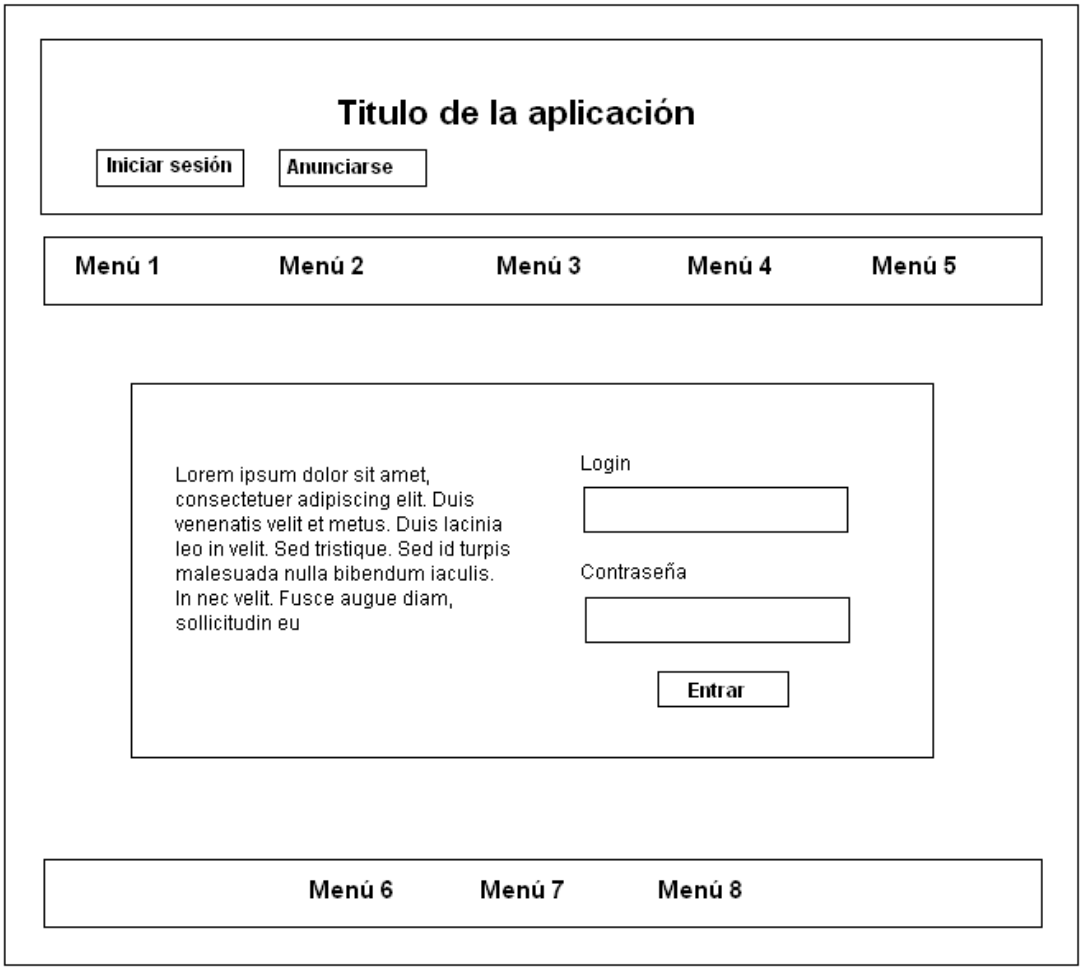

Figura 4-19: Propuesta login de usuario

En esta vista se valida si el usuario esta registrado en la base de datos y si la contraseña corresponde a este usuario en caso contrario se le indica que pruebe a introducir la de nuevo.

Una vez iniciada la sesión correctamente, la aplicación nos derivara al espació personal donde no encontraremos la siguiente vista. Cada submenú de mi perfil, mis anuncios y anunciarme se despliega dentro del recuadro del centro para que el usuario puede realizar las operaciones correspondientes a la opción elegida.

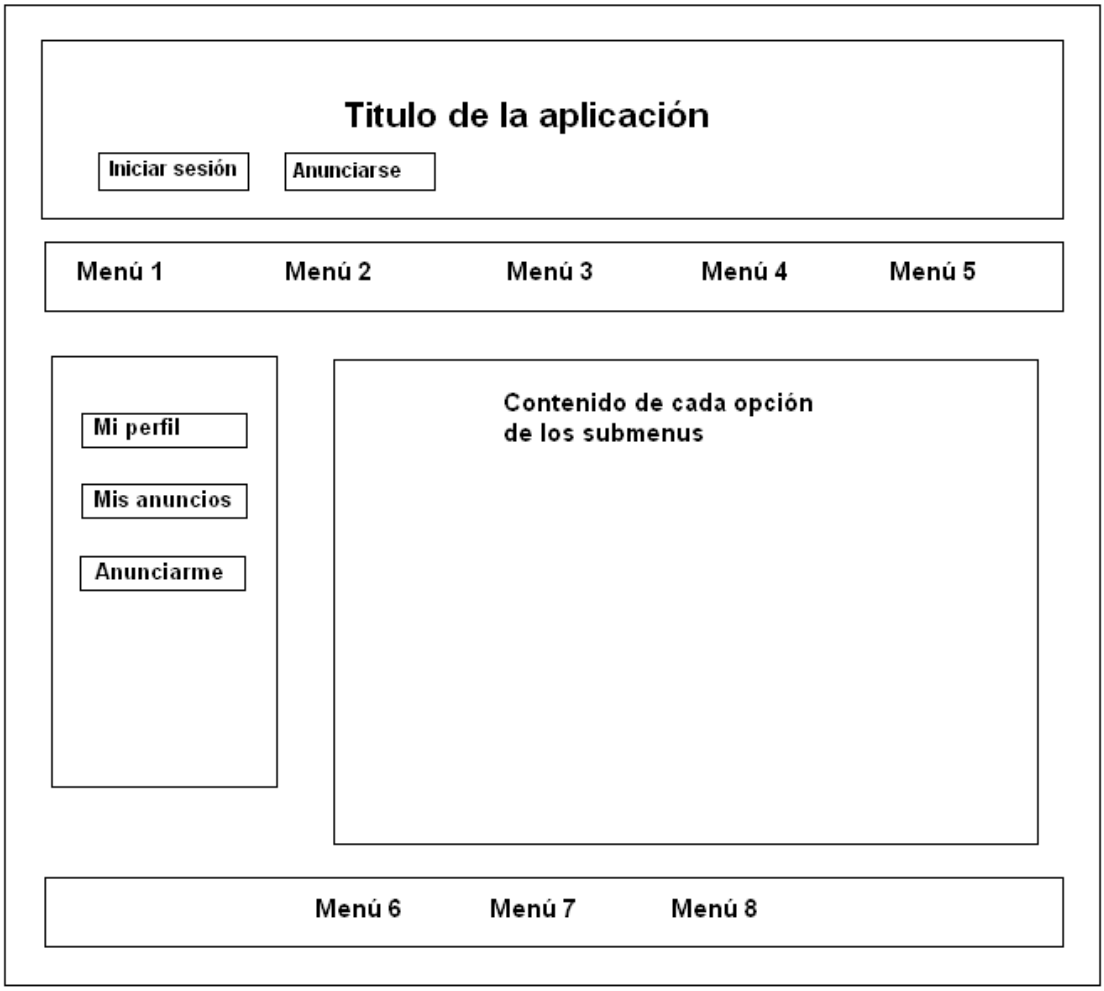

Figura 4-20: Propuesta de espacio personal

En el caso de que un usuario registrado quiera anunciarse, se le aparecerá la siguiente vista (Fig.4-21), en la cual tendrá que completar el siguiente formulario, teniendo en cuenta los campos obligatorios para que se proceda bien operación.

También se le explicara para que sirve este servicio mediante un texto que aparecerá en la parte izquierda del formulario.

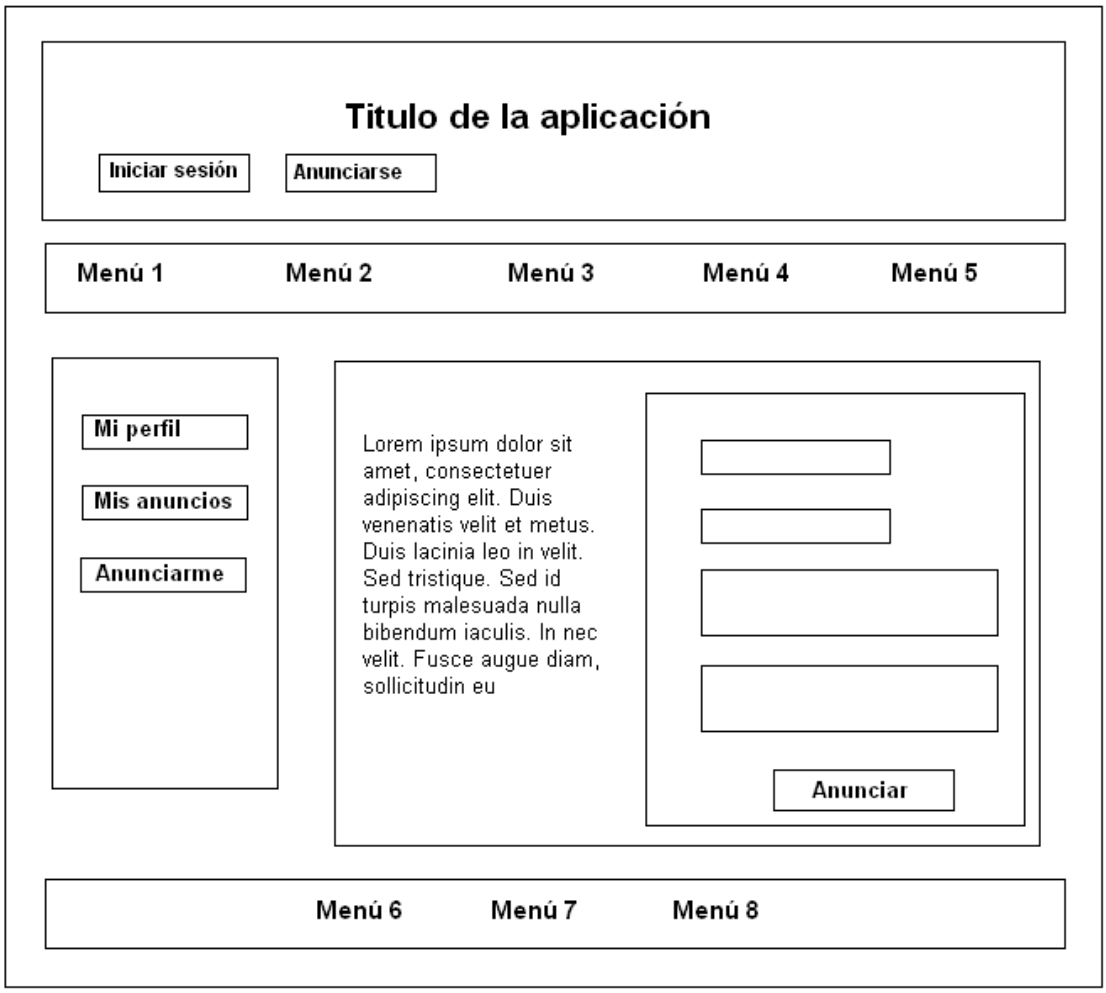

Figura 4-21: Propuesta de vista Insertar anuncio

En caso de que un nuevo usuario quiera registrarse para anunciarse le aparecerá la siguiente pantalla donde deberá escoger entre ser una particular y una empresa (Fig.4-22).

Una vez determinado el tipo de usuario que es, la aplicación le dirigirá al siguiente submenú de registro.

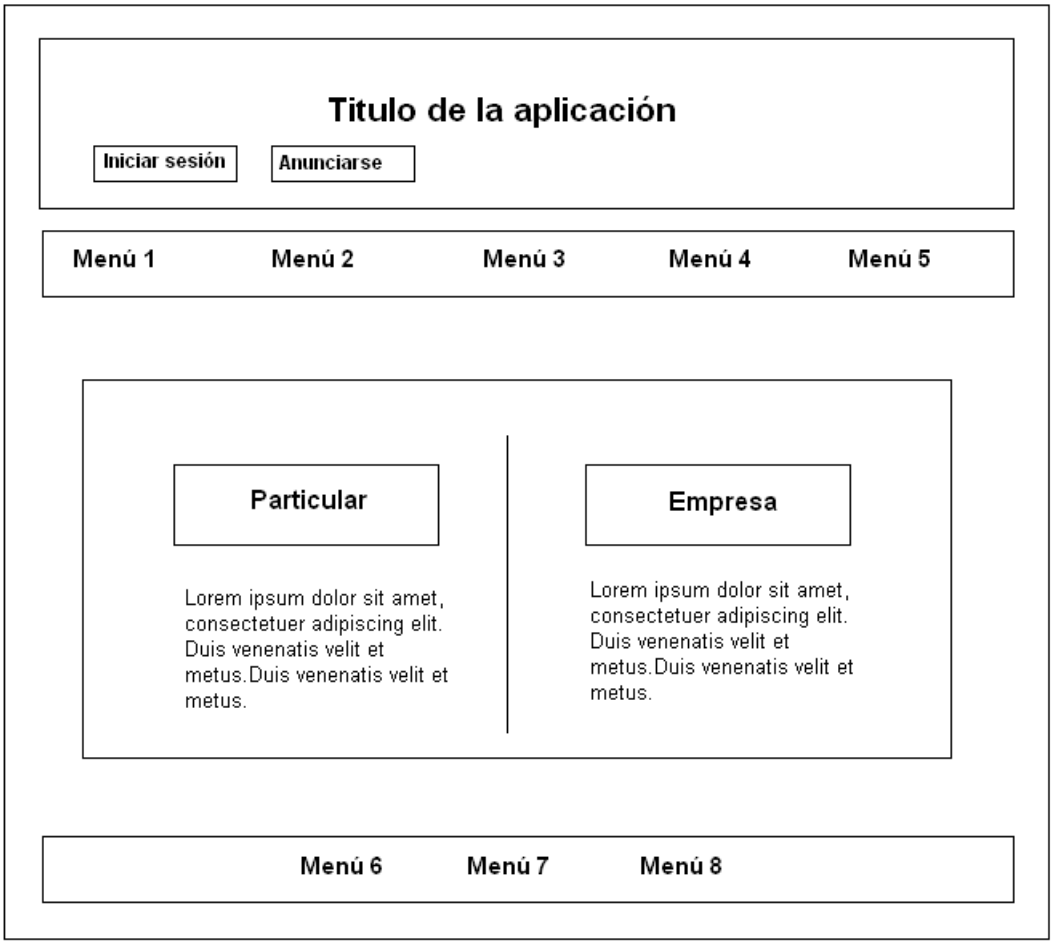

Figura 4-22: Propuesta de vista registrar

En la siguiente pantalla, Fig.4-23, el usuario encontrará un formulario que tendrá que rellenar para poder darse de alta dentro de la aplicación y así poder anunciarse en cualquiera de los apartados de la aplicación. Para ello es necesario rellenar los campos obligatorios.

En la parte izquierda del formulario se encuentra una nota informativa sobre las ventajas de registrarse.

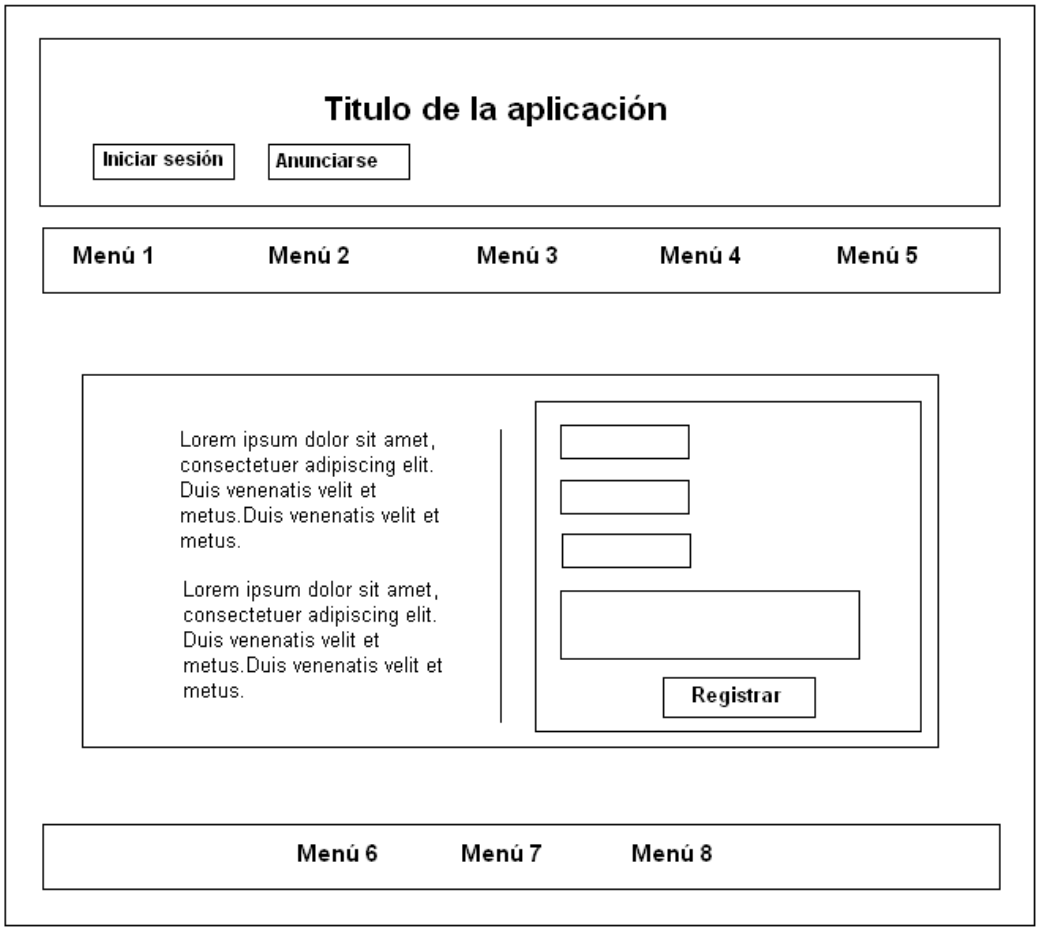

Figura 4-23: Propuesta de vista registrar usuario

4.3 Replanificación

Respecto a la planificación realizada en el estudio de viabilidad han surgido cambios.

- Cambios del diseño de diferentes imágenes de las vistas.
- Se han detallado una nueva organización estructural de diversas pantallas.
- Se han incorporado nuevas pantallas y nuevas imágenes para estas pantallas.

Proyecto

Tareas

Nombre de tarea	Duración	Comienzo	Fin
Proyecto	**110 días**	**mar 10/01/12**	**jue 07/06/12**
Brainstorming de las funcionalidades	6 días	lun 16/01/12	lun 23/01/12
Investigación de tecnologías	**29 días**	**mar 24/01/12**	**vie 02/03/12**
Desarrollo del proyecto	**66 días**	**sáb 03/03/12**	**mié 30/05/12**
Release Proyecto	**6 días**	**jue 31/05/12**	**jue 07/06/12**

Figura 4-24: Tareas

Diagrama de Gantt

Figura 4-25: Diagrama de Gantt

Etapa 1: Investigación de tecnologías

Tareas

Nombre de tarea	Duración	Comienzo	Fin
Proyecto	**110 días**	**mar 10/01/12**	**jue 07/06/12**
Brainstorming de las funcionalidades	6 días	lun 16/01/12	lun 23/01/12
Investigación de tecnologías	**29 días**	**mar 24/01/12**	**vie 02/03/12**
Estudio de patrones MVC	4 días	mar 24/01/12	vie 27/01/12
Estudio de herramientas de mapping	4 días	lun 30/01/12	jue 02/02/12
Estudio del patron Struts	7 días	vie 03/02/12	lun 13/02/12
Estudio de la herramienta MyBatis	7 días	mar 14/02/12	mié 22/02/12
Estudio del lenguaje JavaServer Pages	7 días	jue 23/02/12	vie 02/03/12
Desarrollo del proyecto	**66 días**	**sáb 03/03/12**	**mié 30/05/12**
Release Proyecto	**6 días**	**jue 31/05/12**	**jue 07/06/12**

Figura 4-26: Etapa 1: Tareas

Diagrama de Gantt

Figura 4-27: Etapa 1: Diagrama de Gantt

Etapa 2: Desarrollo del proyecto

Tareas

Nombre de tarea	Duración	Comienzo	Fin
Desarrollo del proyecto	**66 días**	**sáb 03/03/12**	**mié 30/05/12**
Generación de la Base de datos	**8 días**	**sáb 03/03/12**	**mié 14/03/12**
Estudio de la Base de datos	4 días	sáb 03/03/12	mié 07/03/12
Creación de la Base de datos	5 días	jue 08/03/12	mié 14/03/12
Estudio del diseño	**8 días**	**jue 15/03/12**	**lun 26/03/12**
Estudio de Photoshop CS5	4 días	jue 15/03/12	mar 20/03/12
Estudio de Illustrator CS4	4 días	mié 21/03/12	lun 26/03/12
Implementación	**49 días**	**mar 27/03/12**	**mié 30/05/12**
Home Page	**9 días**	**mar 27/03/12**	**jue 05/04/12**
Diseño de imágenes	6 días	mar 27/03/12	lun 02/04/12
Implementación	3 días	mar 03/04/12	jue 05/04/12
Crear usuarios y empresas	**11 días**	**vie 06/04/12**	**vie 20/04/12**
Diseño de imágenes	6 días	vie 06/04/12	vie 13/04/12
Implementación	6 días	sáb 14/04/12	vie 20/04/12
Crear espacio personal de usuarios y empresas	**21 días**	**sáb 07/04/12**	**vie 04/05/12**
Diseño de imágenes	5 días	sáb 21/04/12	jue 26/04/12
Implementación	6 días	vie 27/04/12	vie 04/05/12
Crear visualización de anuncios	**14 días**	**sáb 05/05/12**	**jue 24/05/12**
Diseño de imágenes	6 días	sáb 05/05/12	vie 11/05/12
Implementación	15 días	sáb 05/05/12	jue 24/05/12
Crear pantalla de Términos y contacta	**22 días**	**sáb 28/04/12**	**sáb 26/05/12**
Diseño de imágenes	1 día	vie 25/05/12	vie 25/05/12
Implementación	1 día	sáb 26/05/12	sáb 26/05/12
Redes sociales	**4 días**	**dom 27/05/12**	**mié 30/05/12**
Estudio API de Facebook	1 día	dom 27/05/12	dom 27/05/12
Implementación	1 día	lun 28/05/12	lun 28/05/12
Desarrollo pagina de Facebook	2 días	mar 29/05/12	mié 30/05/12

Figura 4-28: Etapa 2: Tareas

Diagrama de Gantt

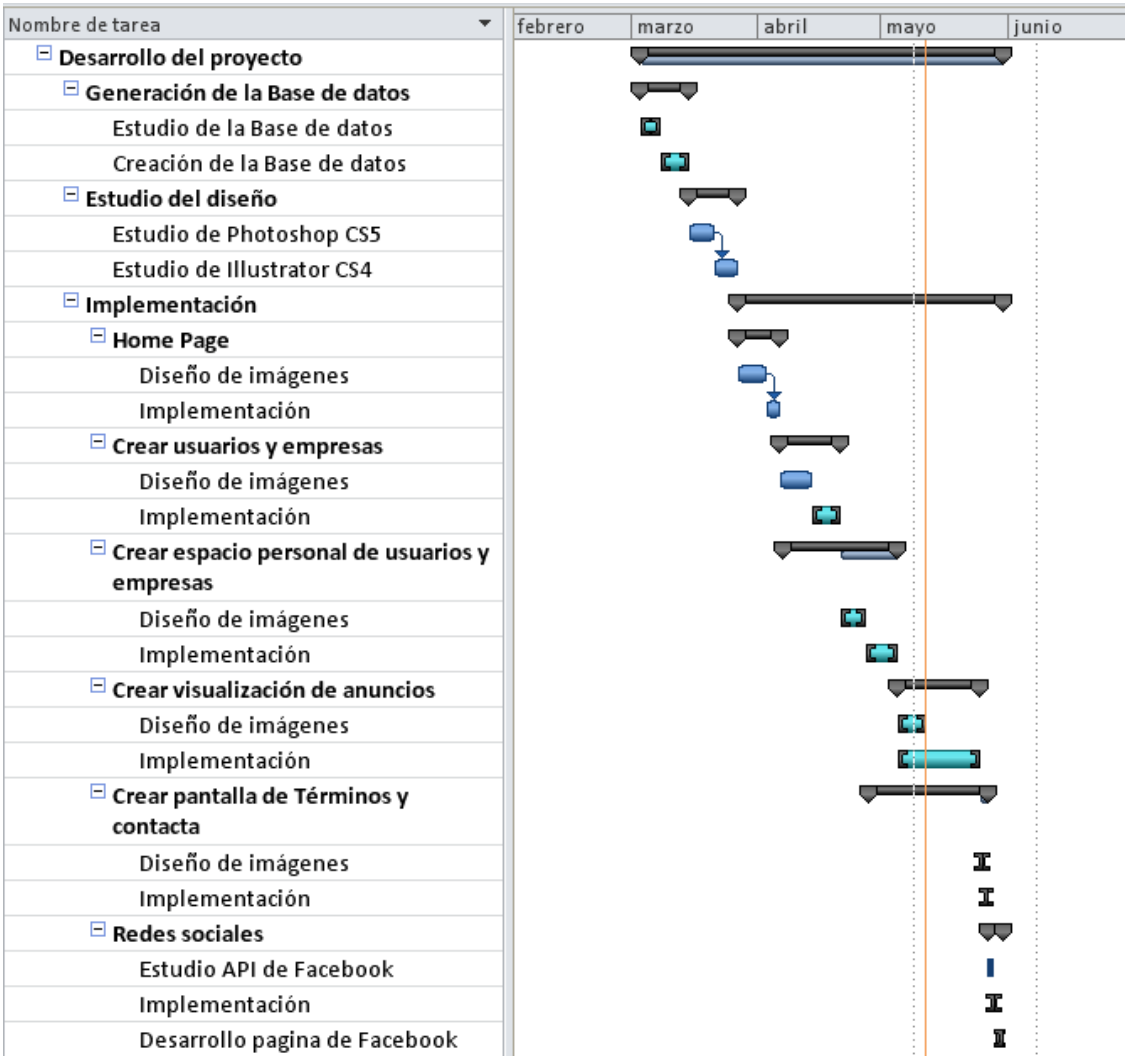

Figura 4-29: Etapa 2: Diagrama de Gantt

Etapa 3: Release del proyecto

Tareas

Nombre de tarea	Duración	Comienzo	Fin
Proyecto	**110 días**	**mar 10/01/12**	**jue 07/06/12**
Brainstorming de las funcionalidades	6 días	lun 16/01/12	lun 23/01/12
Investigación de tecnologías	**29 días**	**mar 24/01/12**	**vie 02/03/12**
Desarrollo del proyecto	**66 días**	**sáb 03/03/12**	**mié 30/05/12**
Release Proyecto	**6 días**	**jue 31/05/12**	**jue 07/06/12**
Estudio de un Hosting	3 días	jue 31/05/12	lun 04/06/12
Configuración	3 días	mar 05/06/12	jue 07/06/12

Figura 4-30: Etapa 3: Tareas

Diagrama de Gantt

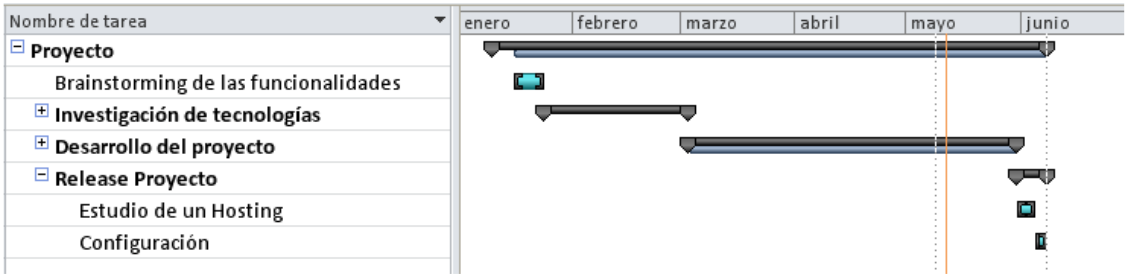

Figura 4-31: Etapa 3: Diagrama de Gantt

5 Implementación

Una vez estudiado el diseño que tendrá la aplicación, se han investigado diferentes tecnologías que implementarían este proyecto. Las partes tecnológicas que se deben determinar antes de empezar a implementar son las siguientes:

- Determinar el entorno de desarrollo de la aplicación.
- Determinar el patrón de la base de la aplicación.
- El lenguaje de programación con el que nos comunicaremos con el servidor.
- El servidor que utilizaremos dependiendo del lenguaje seleccionado.
- Dependiendo del patrón seleccionado, determinar las herramientas que complementen a esta.
- Determinar qué tipo de Base de datos utilizaremos para almacenar los datos que se manipulan dentro de la aplicación.
- Investigación de librerías que puedan darnos soporte en la programación de la aplicación.

5.1 Tecnologías de la aplicación

Una vez realizado las comparativas entre diferentes entornos de desarrollo, se decidió desarrollar el proyecto en Java Web con la intención de aprender a desarrollar aplicaciones mediante esta tecnología. Por lo tanto se realizaron comparativas entre el **IDE NetBeans** y el **IDE Eclipse**. Visto que el entorno de desarrollo de **NetBeans** es mucho más amigable y más rápido de configurar se determino partir de este entorno. Una vez escogido el entorno es imprescindible disponer de una herramienta de control de versiones del proyecto y un historial de modificaciones, por lo tanto de buscaron herramientas gratuitas que realizaran esta labor. Como herramientas de control de versiones se estudiaron entre **Subversion** y **CVS** dado que ambas herramientas deponen de las mismas características y ofrecen el mismo servicio, se escogió SVN por razones de familiarización con la herramienta.

En la búsqueda de un patrón de arquitectura de software, se acabo con implementar el patrón **Modelo Vista Controlador (MVC)** dado que es una arquitectura que separa la interfaz de usuario, de la lógica de programación y la lógica de negocio. De los diferentes **Frameworks MVC** que encontramos que soportaran lenguaje Java encontramos los siguientes:

- Java Apache Grails
- Java GPL Interface Java Objects
- Java LGPL Framework Dinámica
- Java Apache Struts
- Java Apache Beehive
- Java Apache Spring
- Java ApcheTapestry
- Java Apache Aurora
- Java Apache JavaServerFaces

Tras la investigación de los diferentes Frameworks MVC se determino utilizando el patrón **Struts,** ya que es un patrón muy interesante y que actualmente se utiliza mucho en las empresas igual que el patrón **Spring**. Dada la circunstancia de que se tiene un interés concreto en utilizar y poder dominar el Framework de **Struts**, finalmente se desarrollo el proyecto utilizando esta arquitectura.

Una vez escogida la arquitectura del software, se escogieron los complementos que darían soporte al patrón MVC, estos complementos serían **MyBatis** que se encarga de mappear las consultas hacia la Base de datos y **JavaServerPages** con **JSTL** comunicar las acciones del usuario al controlador.

Una vez preparadas las herramientas para la implementación del proyecto, se eligió una Base de datos **MySQL** junto al software **MySQL Workbench 5.2 CE** para administrar de una manera visual la Base de datos.

5.2 Protocolo SSL (Secure Socket Layer)

Se ha implementado este protocolo a toda la aplicación web, esto nos permite tener mayor seguridad para evitar ataques MitM (*Man in the middle*). Esta situación surge cuando un enemigo adquiere la capacidad de leer, insertar y modificar a voluntad, los mensajes entre dos partes sin que ninguna de ellas conozca que el enlace entre ellos ha sido violado. El atacante debe ser capaz de observar e interceptar mensajes entre las dos víctimas.

Tipos de ataque MitM :

- Intercepción de la comunicación incluyendo análisis del tráfico.
- Ataques a partir de textos cifrados escogidos, en función de lo que el receptor haga con el mensaje descifrado.
- Ataques de sustitución.
- Ataques de repetición.
- Ataque que podría bloquear las comunicaciones antes de atacar una de las partes. La defensa en ese caso pasa por el envío de periódico de mensajes de status autenticados.

La mayoría de Hostings ofrecen la opción de poder integrar este servicio de seguridad a tu aplicación por un precio anual.

5.3 Protocolo SMTP (Simple Mail Transfer Protocol)

SMTP es un protocolo de la capa de aplicación que permite el intercambio de correo electrónico entre computadoras o bien dispositivos. Este protocolo es un estándar oficial de internet definido en el RFC 2821.

El protocolo SMTP trabaja por encima del protocolo TCP/IP y utiliza el puerto 25 en el servidor para establecer la conexión. El funcionamiento es de un Cliente-Servidor, donde un cliente envía a uno o a más de un receptores.

Esta herramienta es muy útil en la aplicación para validar usuarios mediante un correo electrónico enviado a su cuenta existente así de esta manera podemos verificar que este usuario existe. De esta manera también podemos enviar información de el usuario y contraseña en caso de que el usuario registrado haya olvidado su identificación y contraseña.

5.4 Implementación de la interfaz de usuario

5.4.1 Interfaz web

La página principal de la aplicación consta de tres partes, la cabecera, el cuerpo y el pie. En la cabecera de esta página encontramos el logo de la aplicación que acompaña al título. Esta cabecera esta implementada en HTML y Java script para poder regresar la página principal dando haciendo click en el título. También contiene las referencias al os servicios que ofrece la aplicación como Cuidamos de tu mascota, He perdido mi mascota o Busco una mascota. Todas estas referencias están creadas en Html.

El cuerpo de esta página esta también echa en HTML y Java script y contiene la descripción de la aplicación y una galería en Java script que permite visualizar algunas de las mascotas a adoptar. El usuario podrá acceder a ese anuncio dando click en la imagen grande de la mascota.

En el pie de página encontramos los vínculos a las páginas de Términos legales, añadir a favoritos o contáctanos.

La cabecera y el pie de la aplicación se comparten en las diferentes pantallas de la aplicación únicamente cambia el cuerpo de esta.

Figura 5-1: Página principal

5.5 Interfaz Cuidamos a tu mascota

5.5.1 Visualizar anuncio

Una vez entramos mediante uno de los vínculos de la parte superior de la aplicación, nos dirigimos directamente a la visualización de anuncios. Esta página contiene la

cabecera y el pie que las demás páginas solo que a partir de ahora el cuerpo funciona con jQuery, Ajax y el controlador de Struts. Por lo tanto cada consulta que se realice dentro de los anuncios con el filtrado, realizara un acción en “background” que generara un HTML plano y lo devolverá a la pagina inicial para que el navegador interprete este texto plano y lo visualice como en la figura siguiente. Gracias a método Post de jQuery podemos lanzar acciones “background” si perder de vista la pagina donde estamos ahora.

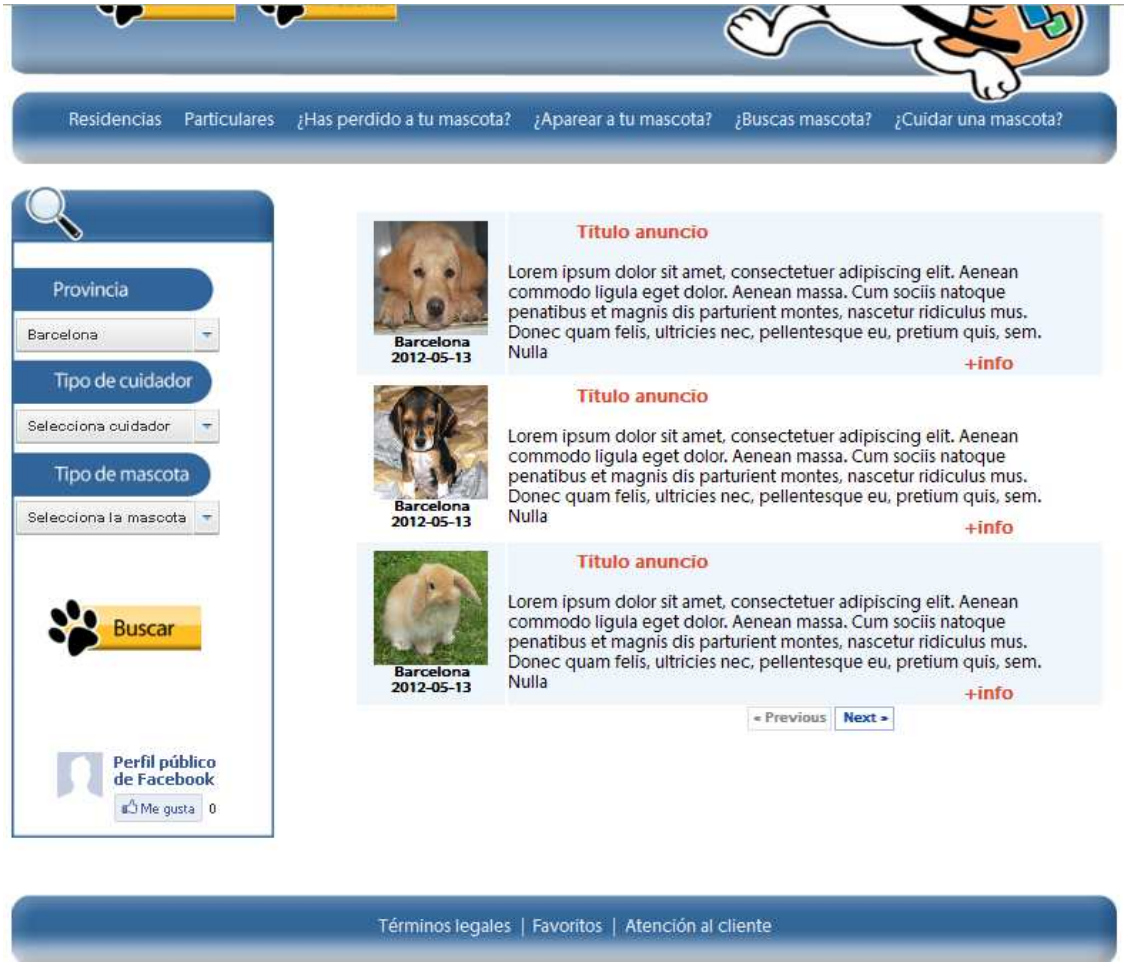

Figura 5-2: Visualización de anuncios

El tratamiento interno de las peticiones es el siguiente, en caso de que un usuario desee buscar filtrando algún campo, internamente se lanzara una acción a un java script que se comunicara con el controlador para que por detrás de nuestra vista de usuario ejecute un Java que realizara la consulta pertinente y en caso de éxito devolverá un texto plano a la función Post en java sctipt que será introducido en un de la pagina

principal que aun estamos visualizando para que el navegador realice la interpretación conveniente.

5.5.2 Crear cuenta de usuario

En caso que un usuario quiera crearse una cuenta encontrara con la siguiente pantalla que le dará la opción de iniciar sesión si ya es un miembro de la aplicación, o bien le pedirá que se identifique como particular o empresa para iniciar el registro.

Figura 5-3: Crear cuenta

Una vez escogida la opción entre particular o empresa, el usuario visualizara la siguiente pantalla donde se le explicara qué ventajas tiene el registrarse y donde deberá rellenar los campos del formulario y acto seguido registrar. Esta pantalla utiliza **HTML Taglib de Struts** ya que se dispone de **Clases Java** que están enlazadas a cada campo de la estructura **HTML Taglib** del formulario y de esa manera es más sencillo y rápido trabajar con los datos introducidos por el usuario y la comunicación con la Base de datos.

Mediante la **Clase Java** de Usuarios se valida que los campos se hayan rellenado correctamente y en caso correcto la aplicación dirigirá al usuario a su espacio personal.

Figura 5-4: Registrar usuario

5.5.3 Iniciar sesión

Una vez un usuario registrado quiera iniciar sesión dentro de la aplicación, accederá ha esta pantalla donde se le pedirá que rellene los campos de usuario y contraseña.
En caso de que el usuario haya olvidad la contraseña o bien el usuario tendrá la opción de realizar una petición mediante los vínculos de Recordar para que el sistema envíe un correo electrónico a este usuario con la información necesaria.

En caso de que un usuario acceda aquí por error tendrá la oportunidad de crear su cuenta mediante un vinculador situado ala izquierda donde se le informa de que puede registrase a través de ese link.

Figura 5-5. Iniciar sesión

5.5.4 Insertar anuncio

Una vez que el usuario este registrado y tenga la intención de anunciarse dentro de alguna sección de la aplicación, se le derivará a través de un vinculo desde la pantalla de espacio personal hacia la pantalla de insertar anuncio.
En esta pantalla tenemos la parte izquierda donde se explica al usuario la ventaja de anunciarse y detalles sobre cómo hacerlo y en la parte derecha se le presenta un formulario que le permite introducir la información necesaria para finalizar el proceso de anunciarse.

Esta pantalla esta creada mediante HTML y TagLib que hacen referencia a una Clase Java llamada Anuncio que se compone de todos los campos aquí presentes en el formulario, de esta manera al realizar la acción de anunciarse, la aplicación enviara la acción al controlador que se encargara de llamar a un Java que con la información introducida insertara en la Base de datos todos los campos que ha rellenado el usuario.

De etsa manera el usuario podrá visualizar sus anuncios insertados en su página personal y poder modificar o eliminarlos.

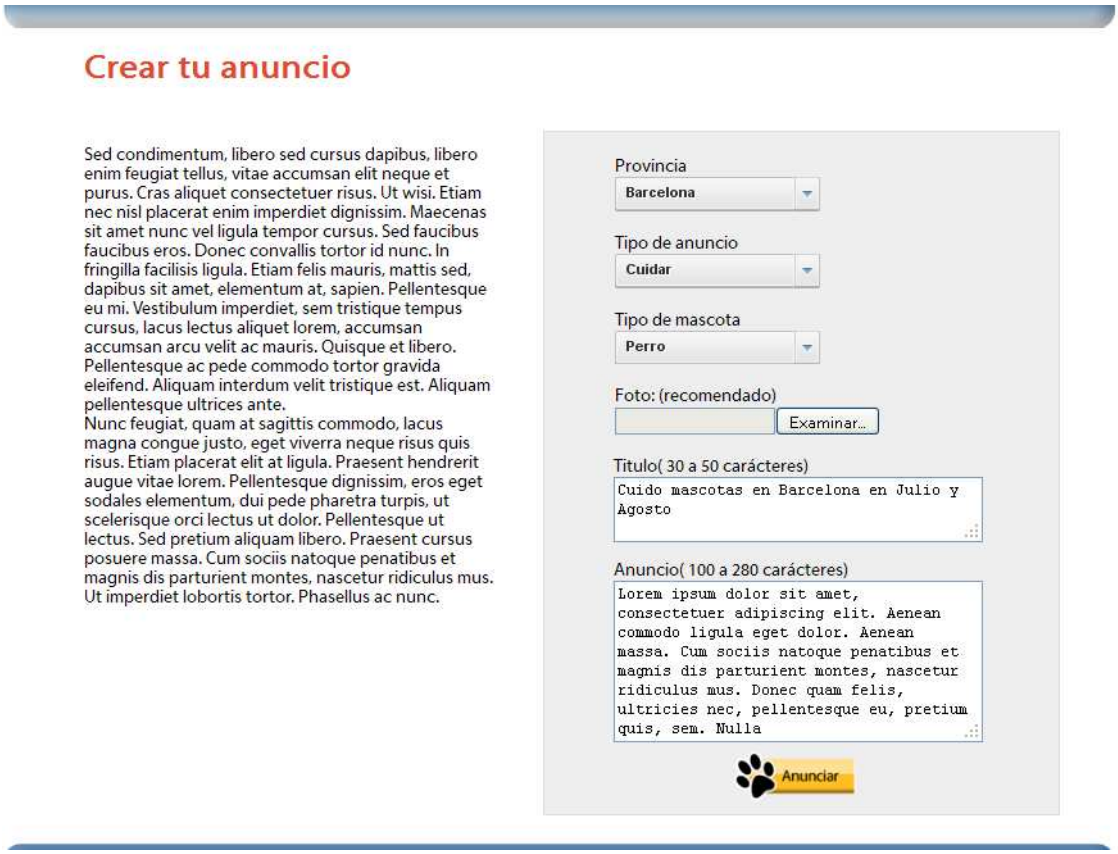

Figura 5-6: Insertar anuncio

5.6 Replanificación

En la planificación real del proyecto ha sufrido cambios de tiempo a causa de:

- El tiempo disponible para realizar el proyecto.
- Aprendizaje de las herramientas que se han ido utilizando.
- Se ha ampliado los días de realización de pruebas.
- La complejidad para añadir el nuevo protocolo de seguridad la aplicación.
- Investigación sobre herramientas útiles para el proyecto y estudio de su funcionamiento.
- Estudiar el servicio SMTP y integrarlo en el proyecto

Nombre de tarea	Duración	Comienzo	Fin
Proyecto	**118 días**	**mar 10/01/12**	**lun 18/06/12**
Brainstorming de las funcionalidades	6 días	mar 10/01/12	mar 17/01/12
Investigación de tecnologías	**29 días?**	**mar 24/01/12**	**vie 02/03/12**
Desarrollo del proyecto	**75 días**	**sáb 03/03/12**	**lun 11/06/12**
Release Proyecto	**5 días**	**mar 12/06/12**	**lun 18/06/12**

Figura 5-7: Tareas

Figura 5-8: Diagrama de Gantt

Etapa 1: Investigación y tecnologías

Tareas

Nombre de tarea	Duración	Comienzo	Fin
Proyecto	**118 días**	**mar 10/01/12**	**lun 18/06/12**
Brainstorming de las funcionalidades	6 días	mar 10/01/12	mar 17/01/12
Investigación de tecnologías	**29 días?**	**mar 24/01/12**	**vie 02/03/12**
Estudio de patrones MVC	4 días	mar 24/01/12	vie 27/01/12
Estudio de herramientas de mapping	4 días	lun 30/01/12	jue 02/02/12
Estudio del patron Struts	7 días	vie 03/02/12	lun 13/02/12
Estudio de la herramienta MyBatis	7 días	mar 14/02/12	mié 22/02/12
Estudio de Hostings			
Estudio de servicio SMTP			
Estudio del lenguaje JavaServer Pages	7 días	jue 23/02/12	vie 02/03/12
Desarrollo del proyecto	**75 días**	**sáb 03/03/12**	**lun 11/06/12**
Release Proyecto	**5 días**	**mar 12/06/12**	**lun 18/06/12**

Figura 5-9: Etapa 1: Tareas

Diagrama de Gantt

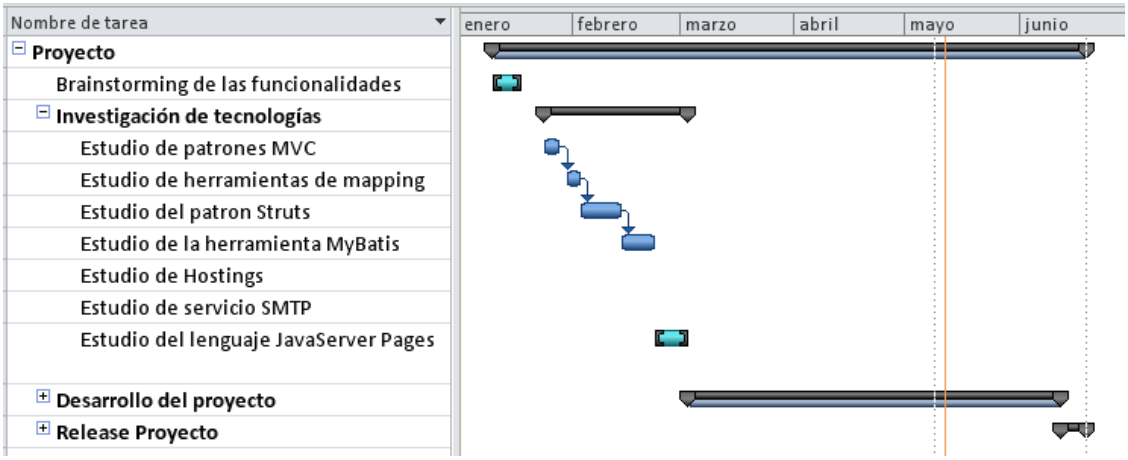

Figura 5-10: Etapa 1: Diagrama de Gantt

Etapa 2: Desarrollo del proyecto

Tareas

Nombre de tarea	Duración	Comienzo	Fin
Desarrollo del proyecto	**75 días**	**sáb 03/03/12**	**lun 11/06/12**
Generación de la Base de datos	**8 días**	**sáb 03/03/12**	**mié 14/03/12**
Estudio de la Base de datos	4 días	sáb 03/03/12	mié 07/03/12
Creación de la Base de datos	5 días	jue 08/03/12	mié 14/03/12
Estudio del diseño	**8 días**	**jue 15/03/12**	**lun 26/03/12**
Estudio de Photoshop CS5	4 días	jue 15/03/12	mar 20/03/12
Estudio de Illustrator CS4	4 días	mié 21/03/12	lun 26/03/12
Implementación	**58 días**	**mar 27/03/12**	**lun 11/06/12**
Home Page	**9 días**	**mar 27/03/12**	**jue 05/04/12**
Diseño de imágenes	6 días	mar 27/03/12	lun 02/04/12
Implementación	3 días	mar 03/04/12	jue 05/04/12
Crear usuarios y empresas	**11 días**	**vie 06/04/12**	**vie 20/04/12**
Diseño de imágenes	6 días	vie 06/04/12	vie 13/04/12
Implementación	6 días	sáb 14/04/12	vie 20/04/12
Crear espacio personal de usuarios y empresas	**21 días**	**sáb 07/04/12**	**vie 04/05/12**
Diseño de imágenes	5 días	sáb 21/04/12	jue 26/04/12
Implementación	6 días	vie 27/04/12	vie 04/05/12
Crear visualización de anuncios	**14 días**	**sáb 05/05/12**	**jue 24/05/12**
Diseño de imágenes	6 días	sáb 05/05/12	vie 11/05/12
Implementación	15 días	sáb 05/05/12	jue 24/05/12
Crear pantalla de Términos y contacta	**22 días**	**sáb 28/04/12**	**sáb 26/05/12**
Diseño de imágenes	1 día	vie 25/05/12	vie 25/05/12
Implementación	1 día	sáb 26/05/12	sáb 26/05/12
Servicio SMTP	**10 días**	**dom 27/05/12**	**jue 07/06/12**
Configuración SMTP	4 días	dom 27/05/12	mié 30/05/12
Implementación del servicio SMTP	6 días	jue 31/05/12	jue 07/06/12
Redes sociales	**3 días**	**vie 08/06/12**	**lun 11/06/12**
Estudio API de Facebook	1 día	vie 08/06/12	vie 08/06/12
Implementación	1 día	sáb 09/06/12	sáb 09/06/12

Figura 5-11: Etapa 2: Tareas

Diagrama de Gantt

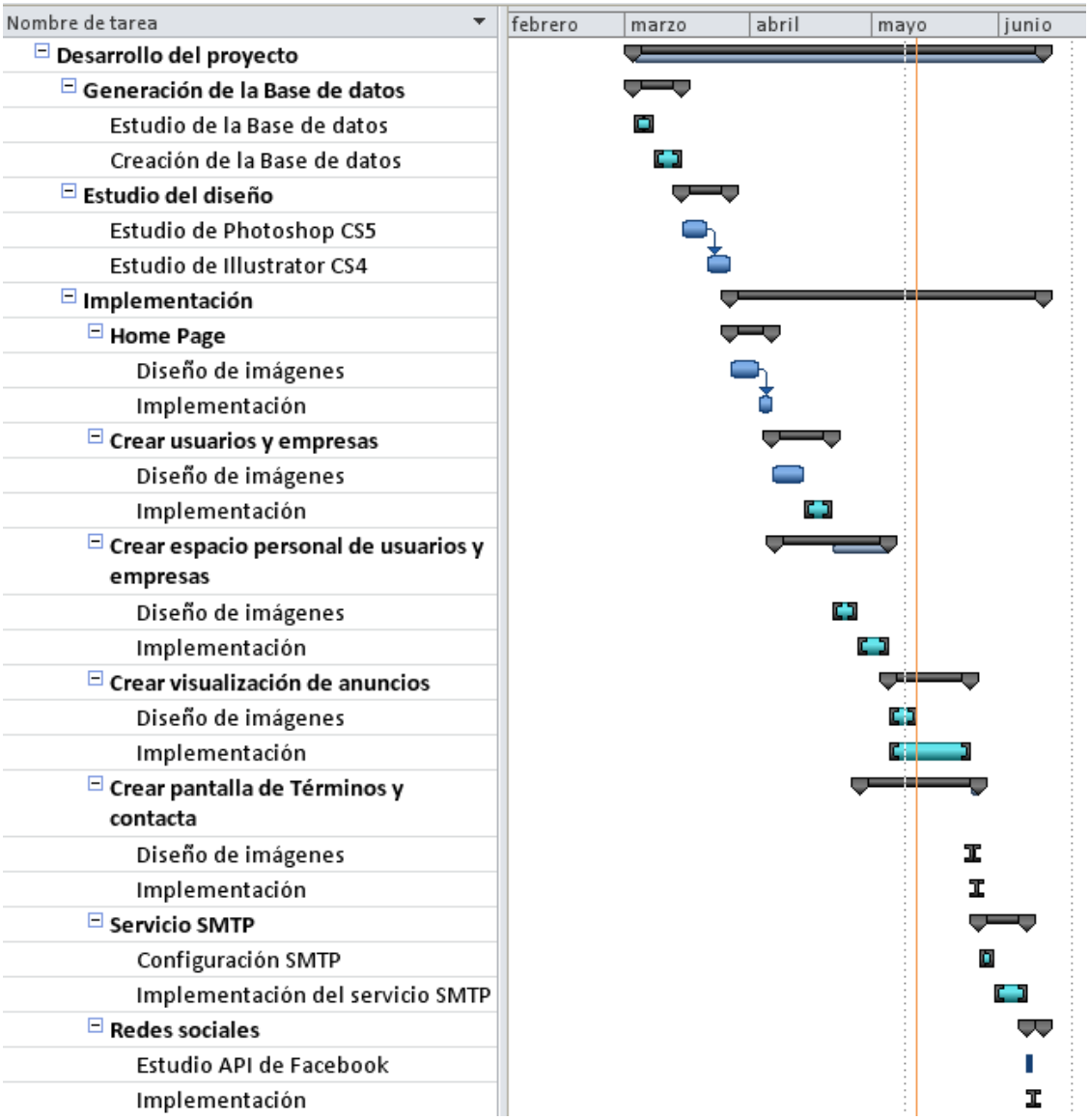

Figura 5-12: Etapa 2: Diagrama de Gantt

Etapa 3: Release del proyecto

Tareas

Nombre de tarea	Duración	Comienzo	Fin
Proyecto	**118 días**	**mar 10/01/12**	**lun 18/06/12**
Brainstorming de las funcionalidades	6 días	mar 10/01/12	mar 17/01/12
Investigación de tecnologías	**29 días?**	**mar 24/01/12**	**vie 02/03/12**
Desarrollo del proyecto	**75 días**	**sáb 03/03/12**	**lun 11/06/12**
Release Proyecto	**5 días**	**mar 12/06/12**	**lun 18/06/12**
Estudio de un Hosting	3 días	mar 12/06/12	jue 14/06/12
Configuración	3 días	jue 14/06/12	lun 18/06/12

Figura 5-13: Etapa 3: Tareas

Diagrama de Gantt

Figura 5-14: Etapa 3: Diagrama de Gantt

6 Propuestas de mejora

Los próximos objetivos está mejorar los servicios que ahora están implementados permitiendo la subida de más imágenes o la posibilidad de permitir ingresar más información en un anuncio.

Objetivos a largo plazo es el estudio de poder integrar un servicio para Guarderías y Residencias que les permitan tener la opción de ofrecerles a sus usuarios la posibilidad de ver a sus mascotas vía cámara web mediante esta aplicación. Para ellos se necesitara hacer un estudio de la situación de la residencia para saber el alcance de internet que tienen a su disposición y después crear en la Base de datos ahora existente las entidades apropiadas a cada usuario de dicha Residencia para que pueda tener acceso a esta cámara que estaría situado en el patío común donde las mascotas están en libertad.

Objetivos que se irán puliendo durante en un futuro será el diseño de la aplicación web donde puede variar el estilo o bien aplicar nuevas tecnologías que vayan apareciendo como HTML 5 o bien tecnologías de diseño como CSS 3.

7 Conclusiones

Al concluir el proyecto se ha conseguido llegar a los objetivos que se habían propuesto para desarrollar la aplicación y poner en marcha el servicio.

Se ha conseguido implementar un aplicación web que ofrece la posibilidad de anunciar a usuario y a empresas para que muestren sus servicios de cuidado, una aplicación que permite anunciar a personas particulares que tienen la intención de cuidar temporalmente tu mascota, aparte ofrecer la posibilidad de anunciar mascotas perdidas, la posibilidad de adoptar una mascota y de encontrar personas que quieran tener camadas cruzando mascotas.

Se ha desarrollado una aplicación que utiliza la arquitectura Modelo Vista Controlador (MVC). Para la implementación de esta arquitectura se a conseguido hacer funcionar correctamente las herramientas MyBatis junto con el lenguaje JSP para poder realizar peticiones a la base de datos y recoger información que se representa mediante HTML en la aplicación Web.

Se ha conseguido hacer funcionar correctamente la metodología de Struts que permite controlar las acciones del usuario. Se han controlado mediante clases Java los datos

introducidos por el usuario para verificar que esta información se haya cumplimentado de manera correcta.

De esta manera se ha obtenido una aplicación modular que permite en un dinamismo en el futuro que permita poder cambiar módulos por otros más eficientes o que introduzcan una nueva tecnología.

Como experiencia personal ha sido muy buena ya que he podido desarrollar cada una de las fases de las que se compone un proyecto hasta su puesta en marcha. He podido acercarme a tecnologías y herramientas que únicamente había utilizado en trabajos donde el proyecto ya estaba empezado y no pude hacerme con ellas. Por lo tanto este proyecto me ha servido para aprender estas nuevas tecnologías y alternativas a ellas y me ha ayudado a utilizar otras herramientas de gestión de proyectos como SVN y Microsoft Project 2010.

He podido ampliar los conocimientos que tenia sobre la gestión de un proyecto y la relación entre aplicaciones y redes sociales como Facebook. He ampliado mi experiencia en la publicación de aplicaciones mediante Shared Hostings e investigar sobre el funcionamiento y la utilización de las herramientas que te proporciona un servicio como este.

8 Bibliografía

[1] http://www.rccarpediem.com, Residencias caninas, 2012

[2] http://www.residenciacaninavoran.com, Residencia Canina Voran, 2012

[3] http://mascotas.facilisimo.com, Mascotas facilísimo, 2012

[4] http://struts.apache.org, Apache Software Fundation Struts, 2012

[5] http://www.hibernate.org, JBoss Community Hiberante, 2012

[6] http://www.mybatis.org, MyBatis, 2012

[7] www.netbeans.org, NetBeans, 2012

[8] www.eclipse.org, The Eclipse Fundation Open Source, 2012

[9] http://www.visionwebhosting.net, Java Hosting, 2012

RESUMEN

El proyecto consiste en la creación un software que permita dar servicio a familias con mascotas que necesiten encontrar de una manera rápida residencias cercanas, protectoras o bien veterinarios. En caso de no poder mantener a la mascota, tener la posibilidad de anunciarlo para que otra familia pueda hacerse cargo de nuestra mascota adoptándola. En esta etapa de crisis económica se quiere ofrecer un servicio que permite a familias con mascota ponerse en contacto con personas de su provincia que se responsabilicen de su mascota durante un periodo de tiempo llegando a un acuerdo. De esta manera los dueños de la mascota obtienen un cuidado temporal más económico que una guardería de verano.

RESUM

El projecte consisteix en la creació d'un software que permeti donar servei a famílies amb animals de companyia que necessitin trobar d'una manera ràpida residencies properes, protectores o bé veterinaris. En el cas de no poder mantenir al animal de companyia, tenir la possibilitat de anunciar-ho per a que un altre família pugui fer-se càrrec del nostre animal de companyia adoptant-lo. En aquesta etapa de crisis econòmica es disposa d'un servei que permeti a famílies posar-se en contacte amb persones de la mateixa província que es responsabilitzin del nostre animal de companyia durant un període arribant a un acord. D'aquesta manera els amos de l'animal obtenen un servei temporal més econòmic que una guarderia d'estiu.

SUMMARY

The project involves creating software that allows serving families with pets who need to find nearby residences, protective or veterinarians. In case you cannot keep your pet, you can advertise for another family to take care of your pet by adopting it. At this stage of economic crisis, provides a service that allows families with pet contact with people of his province to take responsibility for your pet during a period of time reaching an agreement. In this way the pet owners get a foster care cheaper than a summer residence.