Universidad Politécnica de Valencia

Facultad de Informática

Proyecto final de carrera:

Ingeniería Informática

PORTAL WEB:

GESTIÓN INMOBILIARIA

Autor: Roberto Tubilleja Calvo

Director: Félix Buendía García

Mayo 2011

Contenido

1- Introducción 7
1.1 - Motivación 7
1.2 - Objetivos y resumen del proyecto 7
1.3 - Contexto 7
1.4 - Estructura del documento 8
2- Especificación de requisitos 9
2.1- Introducción 9
2.1.1- Propósito 9
2.1.2- Ámbito 9
2.1.3- Definiciones, siglas y abreviaturas 9
2.1.4- Referencias 10
2.2- Descripción general 11
2.2.1- Perspectiva del producto 11
2.2.2- Funciones del producto 11
2.2.3- Características del usuario 13
2.2.4- Restricciones generales 13
2.2.5- Supuestos y dependencias 13
2.3- Requisitos específicos 14
2.3.1- Requerimientos de interfaces externos 14
2.3.2- Requisitos funcionales 15
2.3.3- Requisitos de eficiencia 22
2.3.4- Restricciones de diseño 22
2.3.5- Atributos 22
2.3.6- Otros requerimientos 23
3- Análisis 24
3.1- Casos de Uso 24
3.1.1- Caso de uso: Actor Anónimo 25
3.1.2- Caso de uso: Actor Inquilino 25
3.1.3- Caso de uso: Actor Propietario 26
3.1.4- Caso de uso: Actor administrador 27
3.2- Diagrama de clases 28
3.3- Diagrama de secuencias 30
3.3.1- Inicio de sesión 30
3.3.2- Añadir inmuebles 31

3.3.3- Crear contrato 32
3.3.4- Crear reunión 33
4- Diseño 34
4.1- Arquitectura de tres capas 34
4.1.1- Capa de presentación 35
4.1.2- Capa de lógica de negocio 37
4.1.3- Capa de persistencia 38
5- Implementación 39
5.1- Tecnologías 39
5.1.1- Nivel de presentación 39
HTML 39
CSS 40
JavaScript 41
5.1.2- Nivel de aplicación 42
Php 42
Ajax 44
5.1.3- Nivel de persistencia 45
MySql 45
5.2- Descripción de la implementación 46
5.2.1- Parte pública 47
index.php 47
login .php 48
registro.php 48
5.2.2- Parte privada 49
5.2.2.1- Gestión de inmuebles 51
inmuebles.php 51
anadirInmueble.php 52
5.2.2.2- Gestión de contratos 53
contratos.php 53
contratoNuevo.php 55
5.2.2.3- Gestión de ingresos 57
ingresos.php 57
5.2.2.4- Gestión de gastos mensuales 59
gastos.php 59
5.2.2.5- Gestión de reuniones 60

reuniones.php 60
5.2.2.6- Gestión de desgravaciones 60
desgravaciones.php 60
5.2.2.7- Gestión inquilinos 61
inquilino.php 61
5.2.3- Otras funciones 61
subirImagen.php 61
mostrarImagen.php 61
countbdd.php 62
popcalendar.js 62
6- Evaluación 63
Prueba de validación de CSS 63
Prueba de validación de enlaces 63
Prueba de exploradores 64
Prueba de seguridad de acceso 66
7- Conclusiones 67
7.1- Trabajo realizado 67
7.2- Valoración personal 67
7.3- Futuras mejoras 68
8- Bibliografía 69
Anexos 70
A. Anexo I (Herramientas usadas) 70
Notepad++ 70
Adobe photoshop 72
WAMP 72
Apache 72
Dia 74

Ilustraciones

Ilustración 1: Boceto interfaz 14
Ilustración 2: Actores del diagrama de casos de uso 24
Ilustración 3: Diagrama Casos de Uso: Actor anónimo 25
Ilustración 4: Diagrama Casos de Uso: Actor inquilino 25
Ilustración 5: Diagrama Casos de Uso: Actor propietario 26
Ilustración 6: Diagrama Casos de Uso: Actor Administrador 27
Ilustración 7: Diagrama de clases 28
Ilustración 8: Diagrama Secuencia: Inicio sesión 30
Ilustración 9: Diagrama Secuencia: Añadir inmueble 31
Ilustración 10: Diagrama Secuencia: Crear contrato 32
Ilustración 11: Diagrama Secuencia: Crear reunión 33
Ilustración 12: Arquitectura de tres capas 34
Ilustración 13: Capa de presentación; Página principal 35
Ilustración 14: Cabecera, sin login 35
Ilustración 15: Cabecera, con login 35
Ilustración 16: Menú sin login 35
Ilustración 17: Menú con login 35
Ilustración 18: Cuerpo de la página web 36
Ilustración 19: Pie de la página web 36
Ilustración 20: Módulos de la aplicación 37
Ilustración 21: Capa de persistencia 38
Ilustración 22: Estructura de la página web 46
Ilustración 23: Index.php sin loggeo 47
Ilustración 24: Index.php zona de contenido dinámico 47
Ilustración 25: Index.php pie web de la aplicación 47
Ilustración 26: Login.php 48
Ilustración 27: Registro.php 48
Ilustración 28: Valida_sesion.php 49
Ilustración 29: Comprobación de login 50
Ilustración 30: Página con login 50
Ilustración 31: Inmuebles.php 51
Ilustración 32: AnadirInmueble.php 52
Ilustración 33: Cambio visibilidad de los recuadros de servicios 52
Ilustración 34: Contratos.php 53
Ilustración 35: Selección de columna a ordenar 53
Ilustración 36: Pintar cabecera de columnas ordenadas 54
Ilustración 37: ContratoNuevo.php 55
Ilustración 38: Función Ajax 56
Ilustración 39: Función Ajax 56
Ilustración 40: Ingresos.php 57
Ilustración 41: Generar ingresos 58
Ilustración 42: Gastos.php 59
Ilustración 43: Reuniones.php 60
Ilustración 44: Desgravaciones.php 60

Ilustración 45: Inquilino.php 61
Ilustración 46: Popcalendar.js 62
Ilustración 47: Prueba CSS 63
Ilustración 48: Prueba links 63
Ilustración 49: Explorer 64
Ilustración 50: Chrome 64
Ilustración 51: Mozilla 65
Ilustración 52: Login 66
Ilustración 53: Error login 66
Ilustración 54: Error2 login 66
Ilustración 55: Notepad++ 71
Ilustración 56: Dia UML 74

1- Introducción

Este documento es la memoria descriptiva del proceso de desarrollo de un portal Web, como Proyecto Final de Carrera de Ingeniería Informática realizado en la Universidad Politécnica de Valencia.

El proyecto ha sido realizado por Roberto Tubilleja Calvo y dirigido por Félix Buendía García.

1.1 - Motivación

En nuestro alrededor existen personas que debido al boom inmobiliario han adquirido una gran cantidad de inmuebles y tienen dificultades para su gestión. Estas no disponen de un software sencillo y económico para gestionar sus inmuebles y acaban rindiéndose a inmobiliarias para facilitar sus trámites. Por ello la finalidad de este proyecto es la creación de un portal para facilitar la gestión inmobiliaria a sus usuarios.

1.2 - Objetivos y resumen del proyecto

El objetivo principal de este Proyecto Final de Carrera es desarrollar una aplicación Web que cubra las necesidades de gestión inmobiliaria a particulares y hacer publicidad de estos inmuebles, evitando la contratación de una inmobiliaria y su alto precio.

A nivel académico, el objetivo del proyecto es iniciarse en el desarrollo de aplicaciones de web. Para ello se deberán cubrir los siguientes objetivos secundarios:

- Programación con el lenguaje PHP.
- Manejo de la herramienta de administración de bases de datos phpMyAdmin.
- Contacto con bases de datos MySQL.
- Uso de servidor Apache.

Para el uso de la aplicación se requiere una conexión a internet y un navegador web. También será necesario el conocimiento básico a nivel de usuario para el manejo de la aplicación.

1.3 - Contexto

Para lograr los objetivos marcados para Inmofamily, se ha pensado en la necesaria existencia de diferentes tipos de usuarios con diversos privilegios, según el papel que tengan en la aplicación. Desde un control total sobre la gestión de los recursos de la aplicación para el administrador, pasando por la gestión de inmuebles para los clientes, hasta la visibilidad de inmuebles disponibles para usuarios anónimos.

Desde Inmofamily se ofrecerá una larga lista de servicios para tratar de cubrir los trámites de gestión de los inmuebles agregados a la aplicación. Entre estos servicios podemos destacar: Alta/baja/edición de inmuebles, ingresos y facturas con su estado de pagados o pendientes, alta/baja/edición de contratos, reuniones, desgravaciones. Además de una automática publicación de los inmuebles sin contrato de alquiler vigente.

1.4 - Estructura del documento

La presente memoria está ordenada de forma que se puede resumir cada una de las fases del ciclo de vida de la evolución del proyecto.

Se ha seguido como guía el estándar *IEEE std 830-1998*. En el siguiente apartado se especifican detalladamente los requisitos software de la aplicación.

Introducción: Capítulo de presentación del proyecto, en el se incluyen conceptos como motivación, objetivos, contexto y estructura.

Especificación de requisitos: Capítulo destinado a recoger los requisitos básicos que se demandan antes del inicio del proyecto. La especificación de requisitos es, un compendio de necesidades.

Análisis: En este apartado obtendremos un modelo conceptual de la aplicación mediante la ayuda de diagramas UML.

Diseño: Describirá los diseños realizados para el proyecto así como la metodología utilizada en el proceso.

Implementación: Veremos una descripción de las tecnologías y herramientas utilizadas durante la fase de implementación. Se describirá el funcionamiento de los componentes implementados en la solución.

Evaluación y pruebas: Exposición de las técnicas utilizadas en la evaluación de la aplicación y el resultado obtenido.

Conclusiones: Valoración personal sobre el desarrollo del proyecto y descripción del trabajo realizado.

Bibliografía: Documentación utilizada para el desarrollo.

2- Especificación de requisitos

2.1- Introducción

2.1.1- Propósito

El propósito de este apartado es reunir los requerimientos necesarios para el desarrollo correcto de la aplicación.

Esta especificación va dirigida a toda persona interesada en el funcionamiento de la aplicación. Se seguirán las directrices IEEE std 830-1998, uno de los estándares más referenciados para la especificación de requisitos en proyectos Web.

2.1.2- Ámbito

El producto software a describir es una aplicación web de gestión. Con todas las funcionalidades implementadas será posible llevar la gestión de inmuebles de propiedad privada a nivel particular y hacer publicidad de ellos, sin tener altos conocimientos ofimáticos. Para facilitar el tratamiento de datos se usará una base de datos relacional. Los usuarios podrán gestionar sus inmuebles y anunciar sus inmuebles o solicitar información respecto a los diferentes inmuebles ofertados.

2.1.3- Definiciones, siglas y abreviaturas

- **Apache**: tipo de servidor donde se almacena toda la información de la aplicación, y que permite la gestión de las peticiones sobre la web.
- **Artículo** (Ítem o Artículo de Contenido): Es la mínima pieza de información dentro de la web. Normalmente un texto enriquecido con imágenes. Pertenecerá a una sección y a una categoría.
- **Autenticación o autentificación**: proceso de intento de verificar la identidad digital del remitente de una comunicación como una petición para conectarse.
- **Banner**: es un formato publicitario en Internet. Esta forma de publicidad online consiste en incluir una pieza publicitaria dentro de una página web. Prácticamente en la totalidad de los casos, su objetivo es atraer tráfico hacia el sitio web del anunciante que paga por su inclusión.
- **Base de Datos o BBDD (Database en inglés)**: Una base de datos es una colección organizada de información. Ésta contiene una colección de registros que puede buscar, ordenar y analizar rápidamente.
- **EasyPHP**: Aplicación para instalar Apache, PHP y MySql de forma sencilla
- **Formulario** (HTML FORMS): Servicio que permite introducir información a enviar a la ubicación web remota para procesarla.
- **Hardware**: conjunto de elementos materiales que conforman una computadora.
- **HTML** (HyperText Markup Language): el lenguaje de autor usado para crear documentos en la World Wide web.

- **IEEE** (Institute of Electrical and Electronics Engineers): Asociación de profesionales norteamericanos que aporta criterios de estandarización de dispositivos eléctricos y electrónicos.
- **Interfaz** - Parte del programa informático que permite el flujo de información entre varias aplicaciones o entre el propio programa y el usuario.
- **MySql**: sistema para la creación y gestión SQL de la base de datos.
- **Navegador**: Programa "Navegador" usado para ver e interactuar con varios tipos de recursos de Internet disponibles en la World Wide web.
- **PHP**: lenguaje de programación para el acceso a la base de datos a través de las páginas web de la aplicación.
- **Registro**: Un "Registro" de la base de datos es una descripción de un artículo en particular que se almacena en la base de datos. En una base de datos relacional, cada fila de cada tabla es un registro.
- **Sistema Operativo (SO)**: Un sistema operativo es un software de sistema, es decir, un conjunto de programas de computadora destinado a permitir una administración eficaz de sus recursos. Comienza a trabajar cuando se enciende la computadora, y gestiona el hardware de la máquina desde los niveles más básicos, permitiendo también la interacción con el usuario. Como Windows, Linux, MacOS,…
- **Software**: conjunto de los componentes necesarios para hacer posible la realización de una tarea específica. Soporte lógico de la computadora.
- **WAMP:** Aplicación para instalar Apache, PHP y MySql de forma sencilla

2.1.4- Referencias

- IEEE STD 830- IEEE Guide to Software Requeriments Specifications.
- http://es.wikipedia.org/

2.2- Descripción general

2.2.1- Perspectiva del producto

Este producto software es independiente del sistema operativo utilizado, sólo se necesitará un navegador web e internet para poder acceder a usarlo.

Para ello será necesario un servidor web donde alojar la web además de un dominio para su direccionamiento.

2.2.2- Funciones del producto

Administrador:

- Gestión total de la aplicación, asume los diferentes roles.

Propietario:

- Acceder con sesión propietario personalizada
- Darse de baja usuario como propietario.
- Modificar sus datos como propietario.
- Dar de alta inmueble.
- Dar de baja inmueble.
- Modificar información inmueble.
- Generar ingresos.
- Generar gastos mensuales
- Actualizar ingreso.
- Actualizar gastos mensuales.
- Dar de alta contrato.
- Dar de baja contrato.
- Modificar contenido contrato.
- Dar de alta reunión
- Dar de baja reunión.
- Dar de alta desgravación.
- Dar de baja desgravación.
- Modificar datos reunión.
- Añadir imágenes a inmuebles.
- Borrar imágenes de inmuebles.
- Enviar y recibir mensajería interna.
- Listar de inmuebles
- Listar inquilinos.
- Listar contratos.

- Listar ingresos.
- Listar gastos mensuales.
- Buscar inmuebles.
- Buscar inquilinos.
- Buscar contratos.
- Buscar ingresos.
- Buscar gastos mensuales.
- Buscar reuniones.
- Buscar desgravaciones.

Inquilino:

- Modificar sus datos como inquilino.
- Dar de baja inquilino.
- Enviar y recibir mensajería interna.
- Visualizar pagos de ingresos pendientes
- Visualizar gastos mensuales pendientes
- Solicitar un nuevo contrato
- Renovar contrato

Anónimo:

- Visualizar de inmuebles libres.
- Darse de alta como propietario.
- Darse de alta como inquilino.

Sistema:

- Abrir sesión de usuario.
- Cerrar sesión de usuario.
- Avisar mediante alertas de información de interés “ingresos pendientes” a usuarios propietarios.
- Avisar mediante alertas de información de interés “gastos mensuales pendientes“a usuarios propietarios.
- Avisar mediante alertas de información de interés “reuniones próximas” a usuarios propietarios.

- Avisar mediante alertas de información de interés "finalización de contratos" a usuarios propietarios.
- Generar automáticamente gastos mensuales de cada inmueble dado de alta según sus servicios.
- Generar automáticamente ingresos mensuales por inmueble según lo pactado en el contrato.

2.2.3- Características del usuario

Existirán cuatro tipos de usuarios: administrador*, propietario, inquilino, anónimo*.

- El administrador tendrá un acceso total.
- Un usuario propietario tendrá un acceso a su espacio personal donde gestionar sus inmuebles, contratos, gastos mensuales generados, ingresos, desgravaciones, reuniones y sus propios inquilinos.
- El usuario inquilino tendrá acceso al sistema para comunicarse con los usuarios propietarios y solicitar información de los inmuebles ofertados.
- Un usuario anónimo sólo podrá ver los inmuebles libres que los usuarios propietarios tienen sin alquilar.

La aplicación ofrecerá unos servicios que por su complejidad podrán ser utilizados por cualquier persona, teniendo unos conocimientos mínimos de informática y de navegación por la Web para poder manejar al sitio Web.

2.2.4- Restricciones generales

Para acceder al portal web se deberá disponer del hardware necesario para conectarse a internet, siendo posible vía ordenador, PDA, móvil, Smartphone,…

Cualquier persona podrá visitar la página mediante un navegador web, pudiendo consultar información o acceder a las distintas funciones dependiendo del tipo de rol que tenga dicho usuario.

2.2.5- Supuestos y dependencias

El portal web requiere que los usuarios utilicen un navegador web para acceder a ella. No será necesario el uso de un navegador específico, pero estos navegadores deberán ser compatibles con el protocolo HTTP.

El servidor donde se hospede nuestro sistema deberá admitir:

- Ejecución de código PHP.
- Procesos transaccionales con bases de datos MySql.

2.3- Requisitos específicos

2.3.1- Requerimientos de interfaces externos

2.3.1.1- Interfaces de usuario

Se pretende que la interfaz del portal web sea lo más clara y amigable posible. Para ello se sigue el patrón de la mayoría de las páginas web. Es decir, un menú superior, otro lateral izquierdo y la zona central para mostrar la información. Todo ello acompañado con la interacción del ratón y el teclado por parte de los usuarios.

Las pantallas contendrán un estilo similar pero con diferente información dependiendo del rol que desempeña el usuario. A continuación (Ilustración 1) se muestra un boceto inicial de la interfaz de la aplicación:

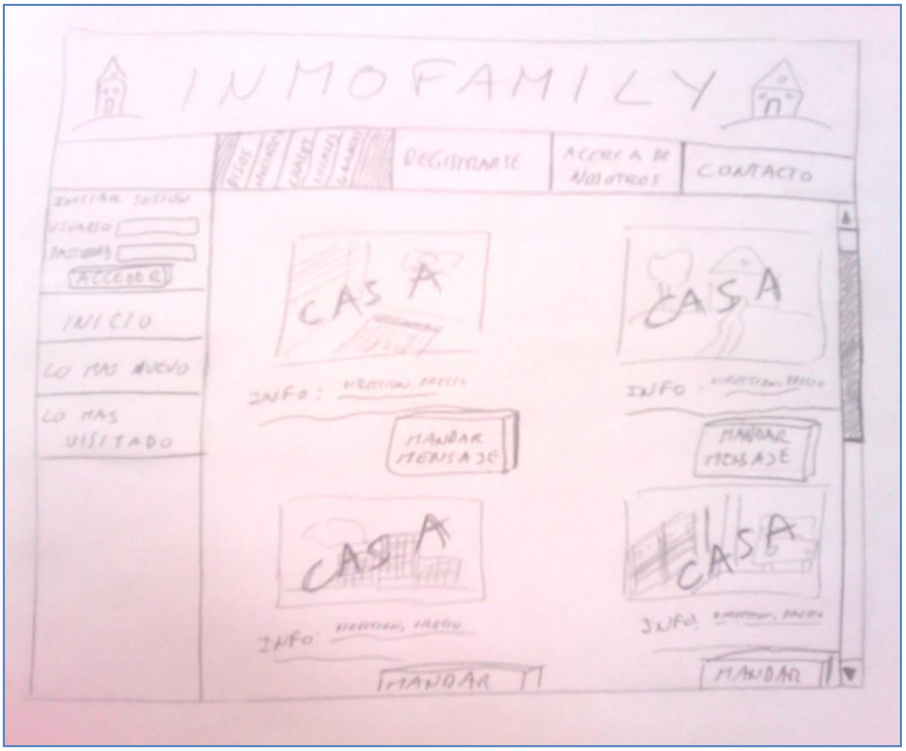

Ilustración 1: Boceto interfaz

2.3.1.2- Interfaces hardware

Los usuarios deberán disponer de acceso a la red para poder acceder a la aplicación, además de un equipo que permita la ejecución de un navegador web que permita resoluciones de al menos 800x600.

Para el alojamiento de datos y de la aplicación se contará con un servicio de almacenamiento externo.

2.3.1.3- Interfaces software

La aplicación ha sido desarrollada para ejecutarse en cualquier sistema operativo y navegador web estándar.

2.3.1.4- Interfaz de comunicaciones

Los interfaces de comunicación entre cliente y servidor son los estándares TCP/IP.

2.3.2- Requisitos funcionales

2.3.2.1- Propietario

Darse de baja como usuario propietario:

- *Introducción:* El usuario ha de ser capaz de darse de baja como usuario.
- Entrada: Identificador del usuario a eliminar.
- Proceso: Los datos del usuario son eliminados del Database.
- Salida: Mensaje de confirmación o de error, dependiendo del resultado de la operación. Error en caso de datos incompletos, incorrectos o inexistentes.

Modificar datos de usuario propietario:

- Introducción: El usuario ha de ser capaz de modificar sus propios datos.
- Entrada: Nuevos datos de usuario. Los datos marcados con * son obligatorios.
- Proceso: Los datos del usuario son actualizados en el Database.
- Salida: Mensaje de confirmación o de error, dependiendo del resultado de la operación. Error en caso de datos incompletos, incorrectos.

Dar de alta inmueble:

- Introducción: El propietario ha de ser capaz de dar de alta nuevos inmuebles.
- Entrada: Datos del nuevo inmueble en el formulario. Los datos marcados con * son obligatorios.
- Proceso: Los datos del nuevo inmueble son registrados en el Database.
- Salida: Mensaje de confirmación o de error, dependiendo del resultado de la operación. Error en caso de datos incompletos, incorrectos o repetidos.

Dar de baja inmueble:

- Introducción: El propietario ha de ser capaz de dar de baja sus inmuebles.
- Entrada: Identificador del inmueble a eliminar.
- Proceso: Los datos del inmueble son eliminados del Database.
- Salida: Mensaje de confirmación o de error, dependiendo del resultado de la operación. Error en caso de datos incompletos, incorrectos o inexistentes.

Modificar datos inmueble:

- Introducción: El propietario ha de ser capaz de modificar los datos de sus inmuebles.
- Entrada: Nuevos datos del inmueble. Los datos marcados con * son obligatorios.
- Proceso: Los datos del inmueble son actualizados en el Database.
- Salida: Mensaje de confirmación o de error, dependiendo del resultado de la operación. Error en caso de datos incompletos, incorrectos.

Actualizar ingreso:

- Introducción: El propietario debe ser capaz de actualizar los ingresos de cada inmueble.
- Entrada: Inmueble afectado, mensualidad a pagar y cantidad ingresada.
- Proceso: Los datos del ingreso son actualizados en el Database.
- Salida: Mensaje de confirmación o de error, dependiendo del resultado de la operación. Error en caso de datos incompletos, incorrectos.

Actualizar gastos mensuales:

- Introducción: El propietario debe ser capaz de actualizar los pagos de los gastos mensuales de cada inmueble.
- Entrada: Inmueble afectado, mes a pagar y cantidad ingresada.
- Proceso: Los datos de los gastos son actualizados en el Database.
- Salida: Mensaje de confirmación o de error, dependiendo del resultado de la operación. Error en caso de datos incompletos, incorrectos.

Dar de alta contrato:

- Introducción: El propietario ha de ser capaz de crear nuevos contratos.
- Entrada: Datos del nuevo contrato en el formulario como cantidad, fianza, ipc, fechas Inicio/revisión/fin, inquilinos e inmueble.
- Proceso: Los datos del contrato son registrados en el Database.
- Salida: Mensaje de confirmación o de error, dependiendo del resultado de la operación.

Dar de baja contrato:

- Introducción: El propietario ha de ser capaz de dar de baja contratos existentes.
- Entrada: identificador del contrato.
- Proceso: Los datos del contrato son eliminados en el Database.
- Salida: Mensaje de confirmación o de error, dependiendo del resultado de la operación.

Modificar datos contrato:

- Introducción: El propietario ha de ser capaz de crear modificar los contratos.
- Entrada: Nuevos datos del contrato en el formulario como cantidad, fianza, ipc, fechas Inicio/revisión/fin.
- Proceso: Los datos del contrato son actualizados en el Database.
- Salida: Mensaje de confirmación o de error, dependiendo del resultado de la operación.

Dar de alta reunión:

- Introducción: El propietario ha de ser capaz de dar de alta nuevas reuniones asociadas a sus inmuebles.
- Entrada: Datos de la nueva reunión en el formulario y el identificador del inmueble correspondiente. Los datos marcados con * son obligatorios.
- Proceso: Los datos de la nueva reunión son registrados en el Database.
- Salida: Mensaje de confirmación o de error, dependiendo del resultado de la operación. Error en caso de datos incompletos, incorrectos.

Dar de baja reunión:

- Introducción: El propietario ha de ser capaz de dar de baja las reuniones asociadas a sus inmuebles.
- Entrada: Identificador de la reunión a eliminar.
- Proceso: Los datos de la reunión son eliminados del Database.
- Salida: Mensaje de confirmación o de error, dependiendo del resultado de la operación. Error en caso de datos incompletos, incorrectos o inexistentes.

Modificar datos reunión:

- Introducción: El propietario ha de ser capaz de modificar las reuniones asociadas a sus inmuebles.
- Entrada: Identificador de la reunión a actualizar.
- Proceso: Los datos de la reunión son actualizados en el Database.
- Salida: Mensaje de confirmación o de error, dependiendo del resultado de la operación. Error en caso de datos incompletos, incorrectos o inexistentes.

Añadir imágenes a inmuebles:

- Introducción: El propietario ha de ser capaz de asociar imágenes a sus inmuebles.
- Entrada: Imagen y el identificador del inmueble correspondiente.
- Proceso: La imagen es almacenada en el Database.
- Salida: Mensaje de confirmación o de error, dependiendo del resultado de la operación. Error en caso de datos incorrectos.

Eliminar imágenes de inmuebles:

- Introducción: El propietario ha de ser capaz de eliminar las imágenes asociadas a sus inmuebles.
- Entrada: Identificador de la imagen a eliminar.
- Proceso: La imagen es eliminada del Database.
- Salida: Mensaje de confirmación o de error, dependiendo del resultado de la operación.

Envío y recepción de mensajería interna:

- Introducción: El usuario ha de ser capaz de modificar enviar mensajes a otros usuarios de la aplicación.
- Entrada: Mensaje, remitente, emisor, fecha de envío. Los datos marcados con * son obligatorios.
- Proceso: El mensaje es actualizado en el Database y mostrado al remitente.
- Salida: Mensaje de confirmación o de error, dependiendo del resultado de la operación.

2.3.2.2- Inquilino

Darse de baja como usuario inquilino:

- Introducción: El usuario ha de ser capaz de darse de baja como usuario.
- Entrada: Identificador del usuario a eliminar.
- Proceso: Los datos del usuario son eliminados del Database.
- Salida: Mensaje de confirmación o de error, dependiendo del resultado de la operación. Error en caso de datos incompletos, incorrectos o inexistentes.

Modificar datos de usuario inquilino:

- Introducción: El usuario ha de ser capaz de modificar sus propios datos.
- Entrada: Nuevos datos de usuario. Los datos marcados con * son obligatorios.
- Proceso: Los datos del usuario son actualizados en el Database.
- Salida: Mensaje de confirmación o de error, dependiendo del resultado de la operación. Error en caso de datos incompletos, incorrectos.

Envío y recepción de mensajería interna:

- Introducción: El usuario ha de ser capaz de modificar enviar mensajes a otros usuarios de la aplicación.
- Entrada: Mensaje, remitente, emisor, fecha de envío. Los datos marcados con * son obligatorios.
- Proceso: El mensaje es actualizado en el Database y mostrado al remitente.
- Salida: Mensaje de confirmación o de error, dependiendo del resultado de la operación.

2.3.2.3- Anónimo

Dar de alta usuario como propietario:

- Introducción: El usuario ha de ser capaz de darse de alta con el rol de propietario.
- Entrada: Datos de usuario del formulario. Los datos marcados con * son obligatorios.
- Proceso: Los datos del nuevo usuario son registrados en el Database.
- Salida: Mensaje de confirmación o de error, dependiendo del resultado de la operación. Error en caso de datos incompletos, incorrectos o repetidos.

Dar de alta usuario como inquilino:

- Introducción: El usuario ha de ser capaz de darse de alta con el rol de inquilino.
- Entrada: Datos de usuario del formulario. Los datos marcados con * son obligatorios.
- Proceso: Los datos del nuevo usuario son registrados en el Database.
- Salida: Mensaje de confirmación o de error, dependiendo del resultado de la operación. Error en caso de datos incompletos, incorrectos o repetidos.

Visibilidad de inmuebles libres:

- Introducción: Cualquier usuario será capaz de visualizar el tablón de anuncios con los inmuebles libres de los usuarios propietarios.
- Entrada: Muestra de la página principal.
- Proceso: Búsqueda en el Database.
- Salida: Muestra de los inmuebles libres actualmente. Error en caso de datos incompletos, incorrectos o fallo de conexión con la base de datos.

2.3.2.4- Sistema

Abrir sesión de usuario:

- Introducción: Esta función servirá como vía de acceso a la sección privada para usuarios de la aplicación.
- Entrada: Datos del formulario de identificación (email y contraseña).
- Proceso: Consulta en la base de dato. Se comprobará que exista la dirección de correo electrónico introducido por el usuario. En caso de existir la dirección de correo electrónico, se comprobará que la contraseña coincida con la almacenada en la base de datos y en caso afirmativo se generará una sesión que será válida hasta que sea cerrado el navegador o hasta que sea ejecutada la función específica para el cierre de sesión.
- Salida: Mensaje de confirmación o de error, dependiendo del resultado de la operación.

Cerrar sesión de usuario:

- Introducción: Usando esta función un cliente podrá cerrar su sesión.
- Entradas: Identificador de sesión.
- Proceso: Se procederá a cerrar la sesión eliminando el identificador de sesión.
- Salida: Mensaje de confirmación.

Búsqueda de contenido:

- Introducción: Dependiendo de los privilegios de usuario se podrán realizar ciertas búsquedas de información según las características indicadas.
- Entradas: Datos que se obtienen de los formularios. Los datos marcados con * son obligatorios.
- Proceso: Se comprobará que exista la información introducida en los formularios. En caso de existir, se mostrará. Si no existe se comunicará mediante un mensaje de aviso.
- Salida: Listado resultado.

Listado de contenido:

- Introducción: Dependiendo de los privilegios de usuario se podrá listar cierta información.
- Entradas: Datos elegidos para listar (inmuebles, inquilinos, contratos, ingresos,…).
- Proceso: Se buscará la información elegida en la tabla correspondiente.
- Salida: Listado resultado.

2.3.3- Requisitos de eficiencia

Ya que el sistema dependerá de la velocidad de acceso a la base de datos alojada en un servidor externo, este deberá proveer a la aplicación los servicios requeridos permitiendo un acceso fluido a todos los usuarios de la aplicación.

También añadir que el peso de la aplicación es lo más reducido posible, para que sea menor el tráfico de datos que tenga que enviarse para hacer más rápida la comunicación.

2.3.4- Restricciones de diseño

2.3.4.1- Estándares cumplidos

No existes restricciones en cuanto a estándares, aunque se ha procurado cumplir los propuestos por la w3c, por ser los más internacionales.

2.3.4.2- Limitaciones hardware

No existen limitaciones hardware.

2.3.5- Atributos

2.3.5.1- Seguridad

Sería interesante incluir sistemas de seguridad para cumplir con la Ley Orgánica de Protección de Datos. Este punto es fundamental si se pretende llevar al mundo real este proyecto.

Los datos confidenciales de la cuenta de un cliente registrado solo serán accesibles para el propietario de la cuenta y los administradores. Un usuario solo podrá acceder a sus datos tras realizar el proceso de autenticación.

Todas las comunicaciones que incluyan datos confidenciales serán protegidas mediante el uso de conexiones seguras SSL, suponiendo que el servidor donde esté alojada la aplicación disponga de cuenta SSL.

2.3.5.2- Mantenimiento

Para realizar un mantenimiento de la aplicación es recomendable seguir este informe para conocer al detalle su funcionamiento interno. Es recomendable hacer una copia de seguridad tanto de la aplicación web como de la base de datos antes de proceder a ningún cambio.

El mantenimiento básico de la aplicación se llevará a cabo por parte de la administradora, entendiendo como tal las tareas de mantener actualizado y al día los servicios que ofrece. Así como cualquier cambio que se deseara introducir y requiriese de la modificación de la base de datos.

2.3.5.3- Portabilidad

Como se ha comentado con anterioridad, esta aplicación web es compatible con cualquier sistema que siga los estándares convencionales de navegación web. Esto significa que nuestra aplicación podrá ser soportada por cualquier plataforma y sistema operativo.

2.3.6- Otros requerimientos

2.3.6.1- Bases de datos

La aplicación trabajará sobre una base de datos MySQL donde se almacenará toda la información necesaria para el funcionamiento de la tienda virtual.

Se han desarrollado funciones con las que añadir, modificar y eliminar información de la base de datos de una forma segura

Para la administración y gestión de la base de datos en momentos puntuales del sistema de administración, propongo el uso de phpMyAdmin.

Esta aplicación basa su funcionamiento en toda la información que almacena en su base de datos. El lenguaje utilizado para acceder será PHP.

2.3.6.2- Inicialización

Para poder acceder a la zona privada de administración es necesario crear un usuario administrador en la base de datos. Para ello se creará un usuario directamente en la base de datos mediante la aplicación de gestión de bases de datos PhpMyAdmin.

3- Análisis

En la fase de análisis del proyecto se estudian todas las funcionalidades de la aplicación a desarrollar. A lo largo de esta fase se describe la estructura y funcionalidad de la aplicación mediante el uso de diagramas UML.

3.1- Casos de Uso

La fase de análisis previa a la construcción de la aplicación describe su funcionalidad utilizando diagramas de casos de uso. Estos diagramas permiten una representación gráfica de las interacciones entre actores (usuarios o aplicaciones externas que podrán demandar la utilización de funciones ofrecidas por el sistema) y casos de uso (forma concreta de utilizar parte de la funcionalidad del sistema). Las interacciones entre actores y casos de uso describen el comportamiento del sistema desde el punto de vista de lo usuarios.

A continuación en la siguiente imagen (ilustración 2) se muestra la relación entre los usuarios de la aplicación.

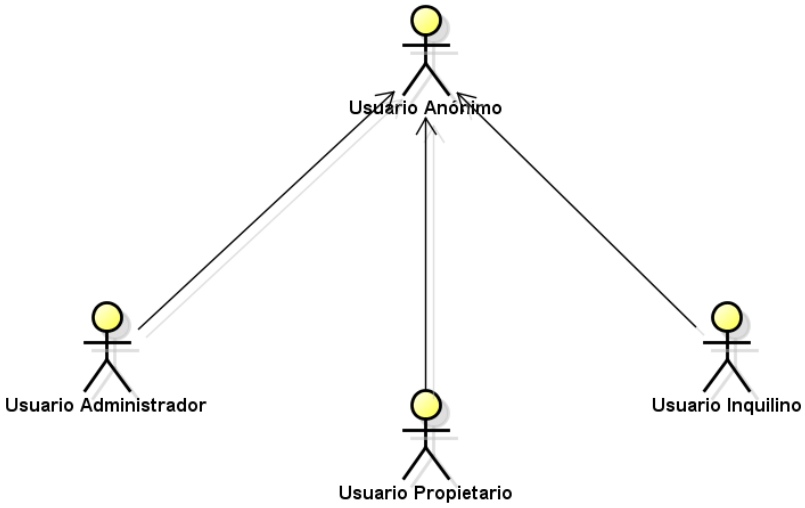

Ilustración 2: Actores del diagrama de casos de uso

Los usuarios podrán realizar diferentes tareas dependiendo del tipo al que pertenezcan, desde el más restringido (usuario anónimo) hasta el más completo (usuario administrador). Se muestra a continuación un diagrama de casos de uso para cada tipo de usuario, para así facilitar la comprensión.

3.1.1- Caso de uso: Actor Anónimo

En la ilustración 3 se pueden ver los casos de usos a los que podrá acceder el actor Anónimo. Dado que este tipo de actor no se ha identificado ante el sistema, las acciones que podrá realizar serán en consecuencia muy limitadas, centrándose en la obtención de información general del centro.

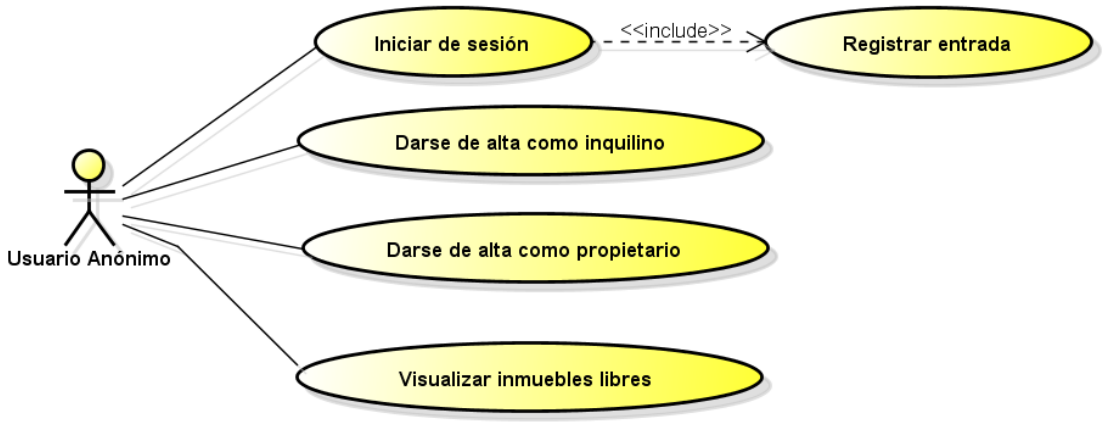

Ilustración 3: Diagrama Casos de Uso: Actor anónimo

3.1.2- Caso de uso: Actor Inquilino

En la ilustración 4 pueden verse todos los casos de usos a los que tendrá acceso el actor inquilino.

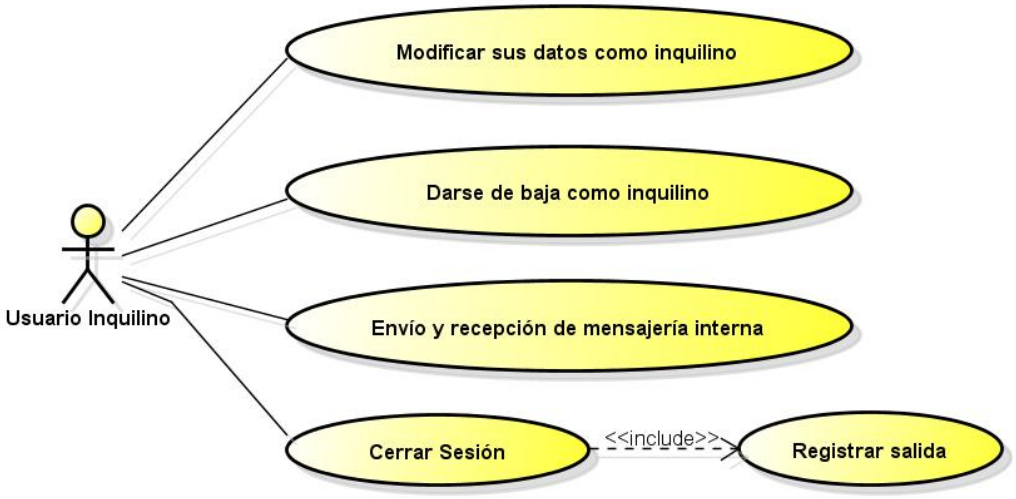

Ilustración 4: Diagrama Casos de Uso: Actor inquilino

3.1.3- Caso de uso: Actor Propietario

En la ilustración 5 se ven los casos de uso del actor propietario. Como puede verse, este usuario dispone de muchos casos de uso, ya que, es el usuario principal de la aplicación.

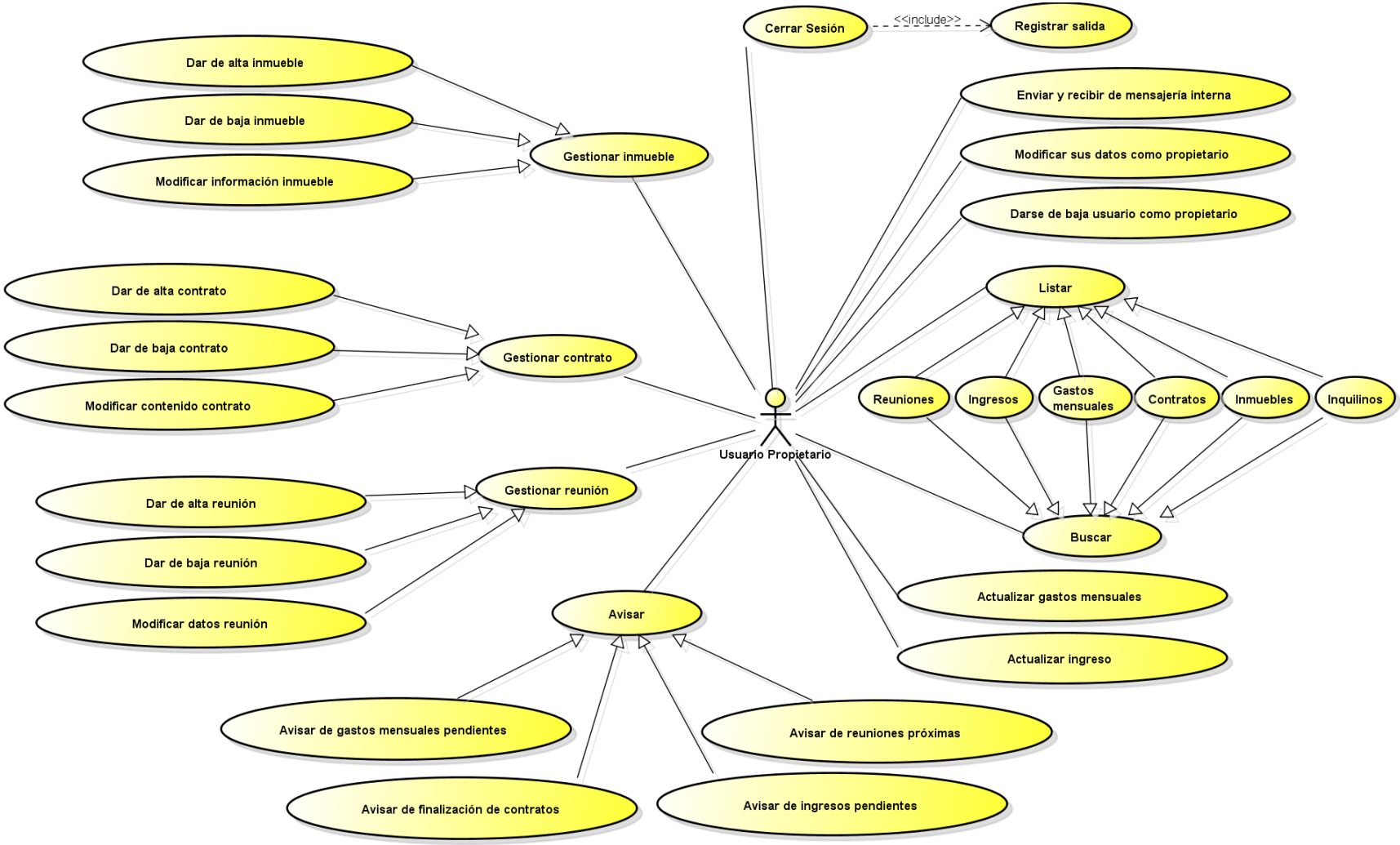

Ilustración 5: Diagrama Casos de Uso: Actor propietario

3.1.4- Caso de uso: Actor administrador

En la ilustración 6, puede verse el caso de uso del actor Administrador. Este actor será el que más funcionalidades posea, ya que, además de las funcionalidades que posee todo usuario registrado, como administrador tendrá acceso total al sistema.

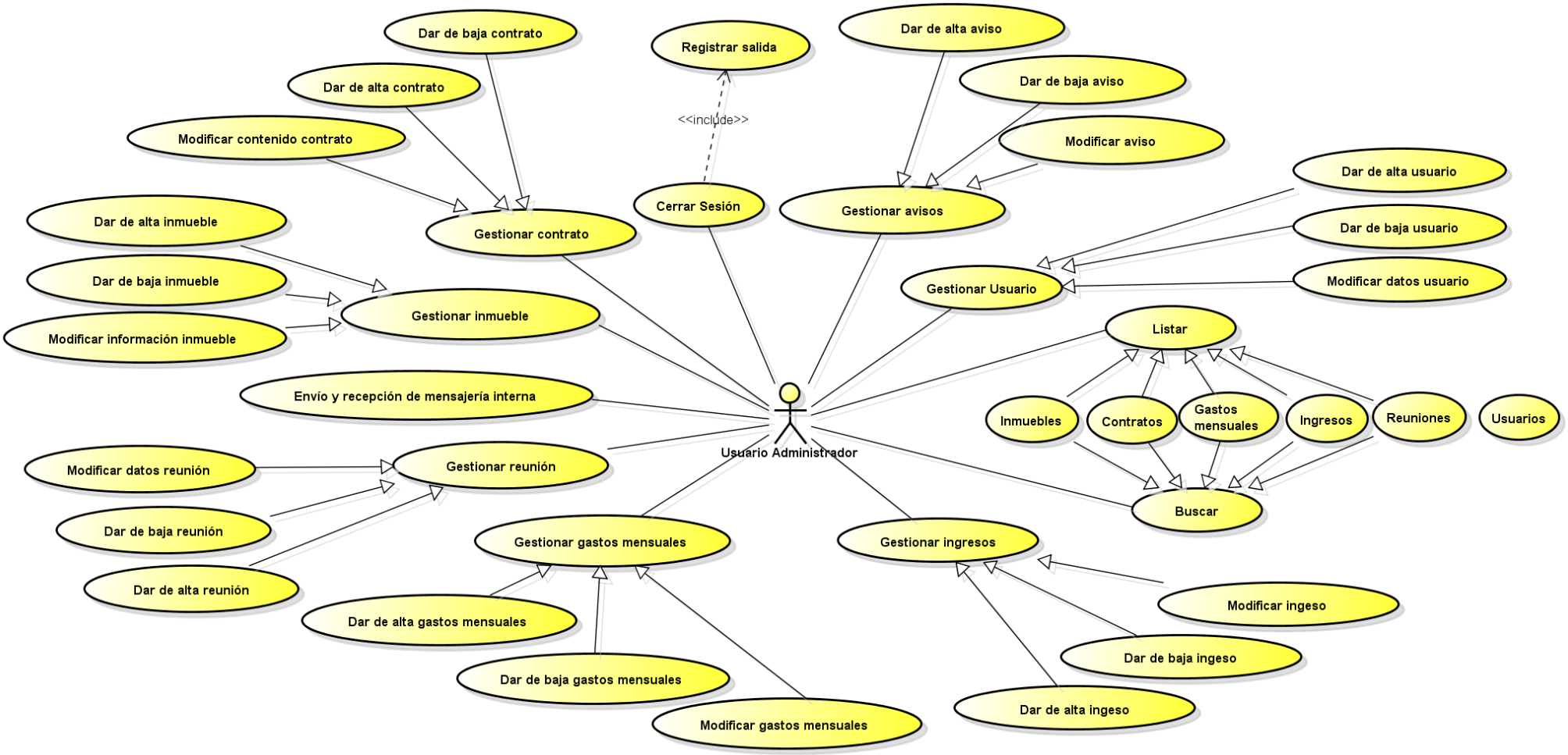

Ilustración 6: Diagrama Casos de Uso: Actor Administrador

3.2- Diagrama de clases

El diagrama de clases (Ilustración 7) se utiliza para describir la estructura de un sistema o aplicación. Utilizando la notación UML para diagramas de clases expondré la estructura para la aplicación Web.

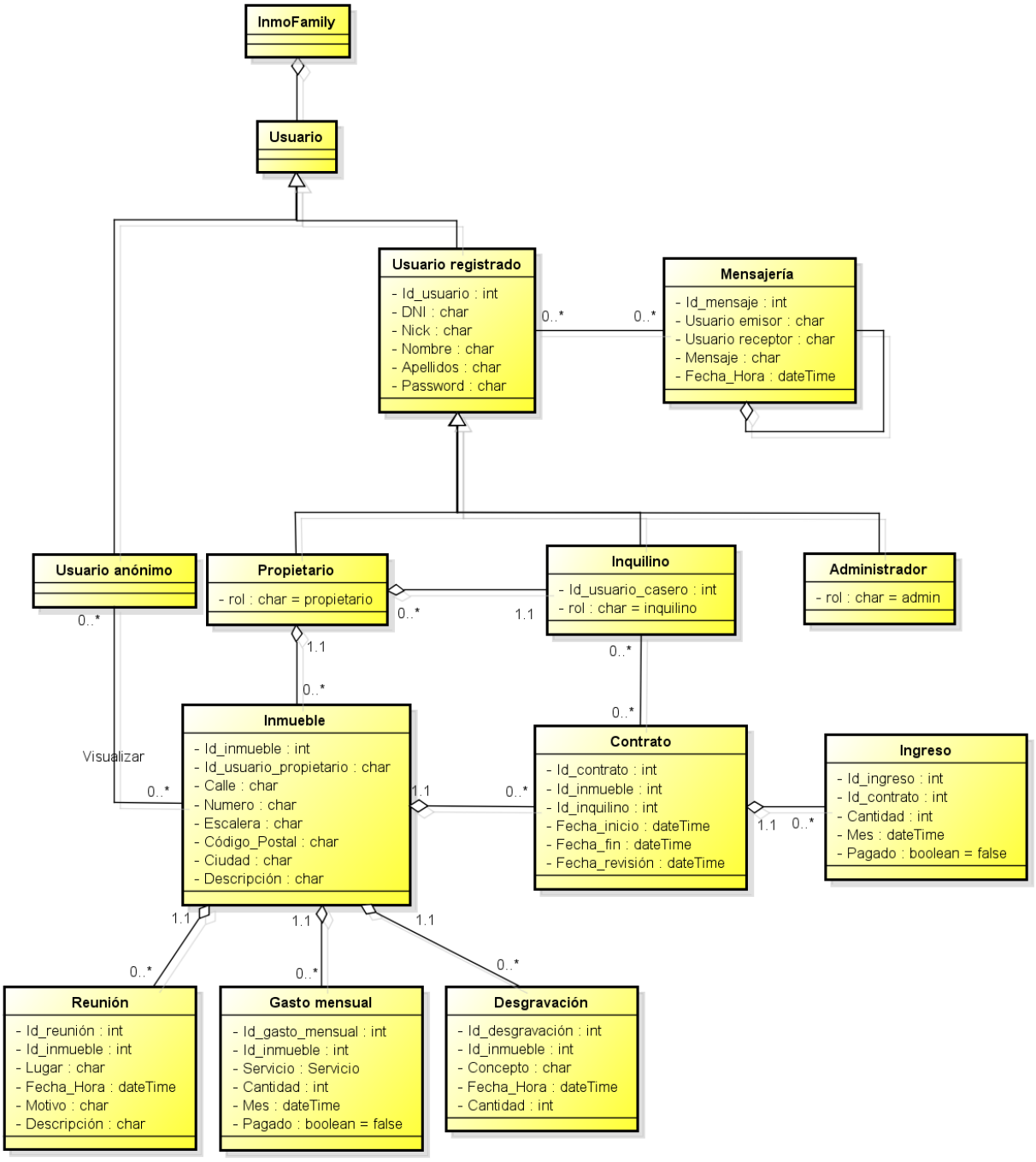

Ilustración 7: Diagrama de clases

Usuario: Será todo aquel usuario que interactúa con la aplicación, con más o menos privilegios.

Usuario registrado: Esta clase representa a aquellos usuarios que pueden acceder a la aplicación tras su correspondiente logeo y que podrán interactuar con el sistema, en la medida de los privilegios de su rol. Tiene una serie de atributos, siendo los más importantes, desde el punto de vista del sistema, el "Id_usuario" que deberá ser único dentro del sistema y el "Password" ya que serán necesarios a la hora de logearse. De esta clase se derivan las subclases: Propietario, Inquilino y Administrador.

Inmueble: Este objeto será uno de los fundamentales de la aplicación. Contendrá la información particular de cada inmueble: su identificador, su dirección, y atributos que describen las características como número de metros cuadrados, número de baños, habitaciones. Cada inmueble estará asignado a un único usuario propietario.

Contrato: Esta clase permite la creación de contratos entre inmuebles e inquilinos, configurando atributos como cantidad y fechas de origen y fin.

Ingreso: Con este objeto se refleja la cantidad que el inquilino debe abonar, tras el acuerdo del contrato. Cada uno de estos ingresos está asociado a un inmueble y una fecha.

Reunión: Esta clase permitirá identificar las reuniones asignadas a cada inmueble. De esta forma se podrá llevar un control de las reuniones más próximas por inmueble.

Gasto mensual: Cada objeto inmueble se relacionará con uno o varios de estos objetos "Gasto mensual" debido a las facturas generadas por los pagos de los servicios asociadas al inmueble.

Desgravación: Esta clase representa las inversiones realizadas en un inmueble que después pasarán a desgravar cuando se realice la declaración de la renta por el usuario propietario.

Mensajería: Los usuarios registrados tendrán la opción de enviarse mensajes a través de objetos de mensajería, los atributos principales son: emisor, remitente, mensaje, fecha y hora

3.3- Diagrama de secuencias

El diagrama de secuencia es un tipo de diagrama usado para modelar interacción entre objetos en un sistema según UML.

Son una forma de diagrama de interacción que muestra los objetos como líneas de vida a lo largo de la página y con sus interacciones en el tiempo representadas como mensajes dibujados como flechas desde la línea de vida origen hasta la línea de vida destino. Los diagramas de secuencia son buenos para mostrar qué objetos se comunican con qué otros objetos y qué mensajes disparan esas comunicaciones. Los diagramas de secuencia no están pensados para mostrar lógicas de procedimientos complejos.

3.3.1- Inicio de sesión

La Ilustración 8 muestra un diagrama de secuencias del acceso a la aplicación para usuarios registrados, en caso correcto se crea una sesión y en caso incorrecto se produce un error.

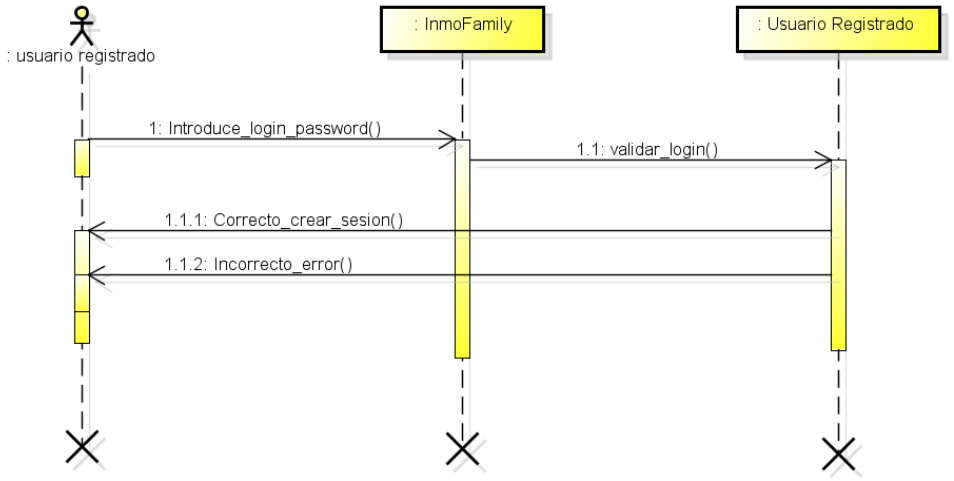

Ilustración 8: Diagrama Secuencia: Inicio sesión

3.3.2- Añadir inmuebles

En la Ilustración 9, se puede observar como los usuarios registrados pueden añadir inmuebles a su usuario, junto con los servicios que este dispone.

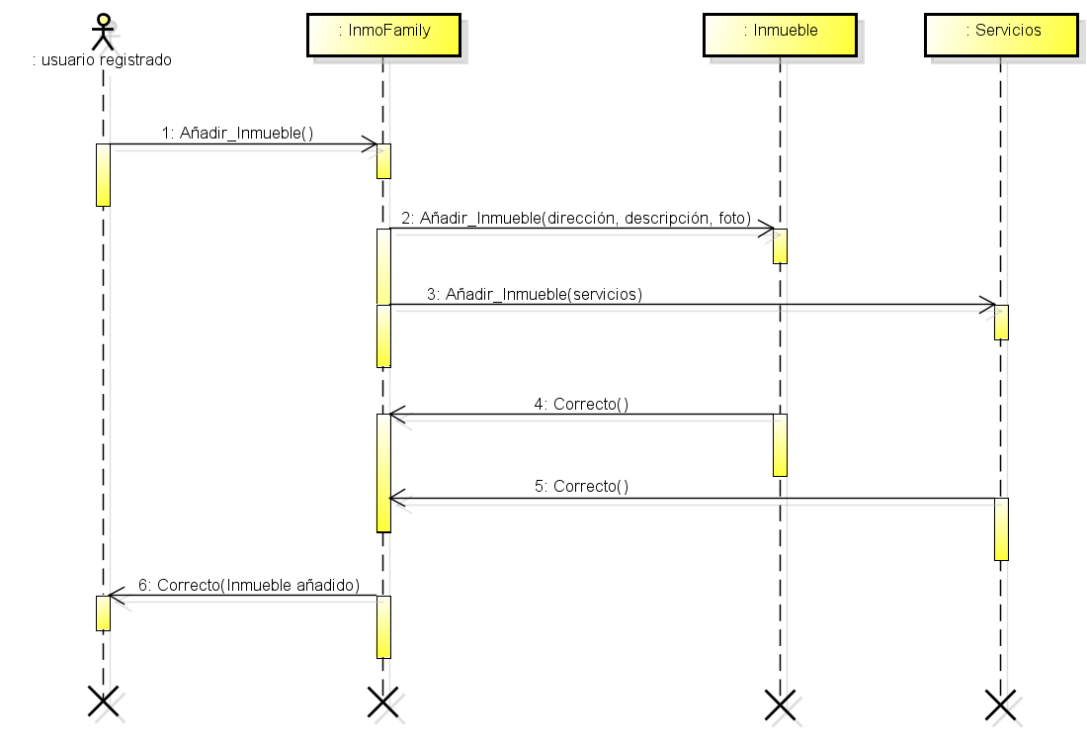

Ilustración 9: Diagrama Secuencia: Añadir inmueble

3.3.3- Crear contrato

Cuando un usuario es registrado y tiene inmuebles en su cuenta, podrá crear contratos para sus propiedades. El proceso es sencillo como se detalla a continuación con el diagrama (Ilustración 10). Se puede observar que entran en juego entidades como inmueble, servicios e inquilino.

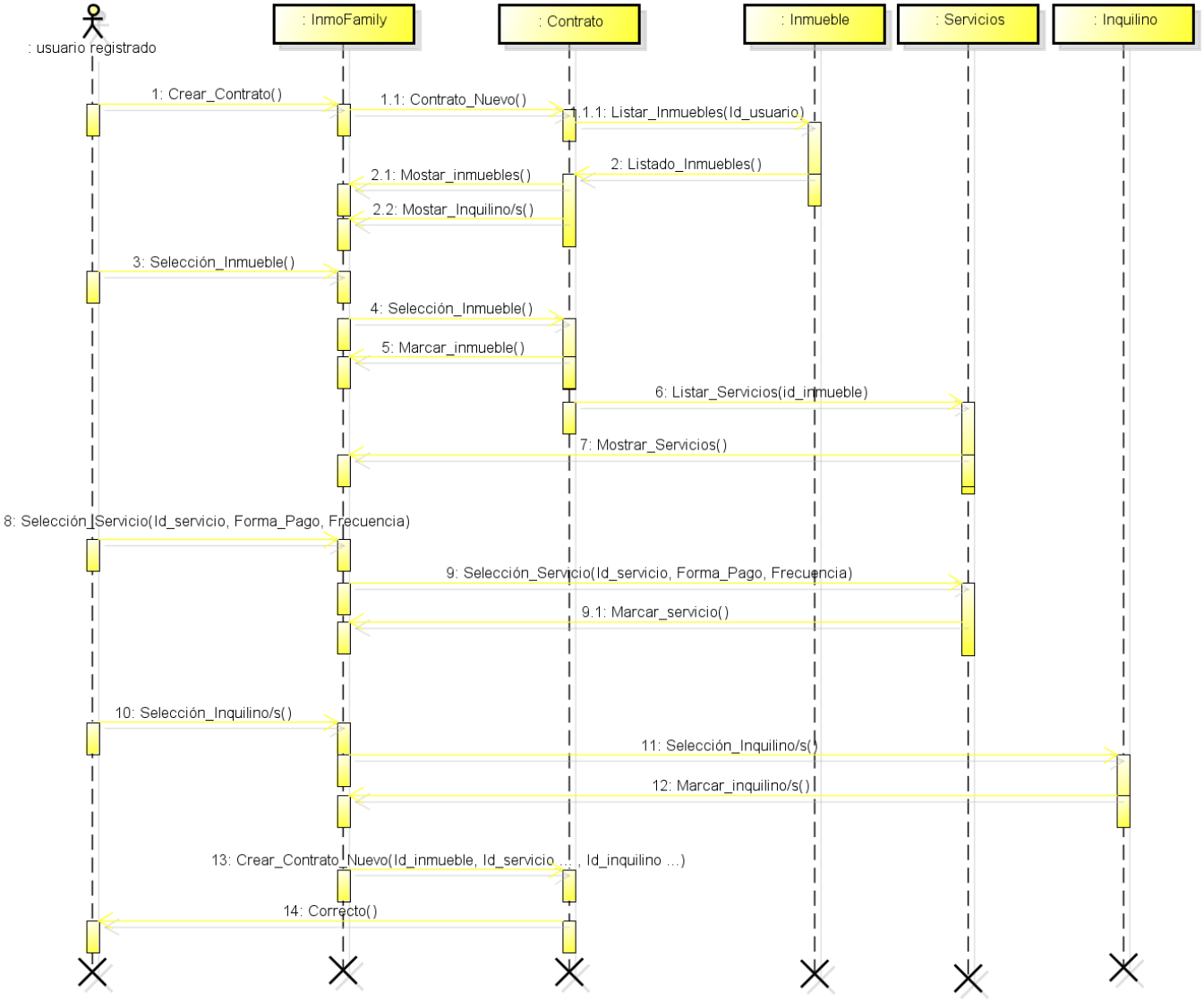

Ilustración 10: Diagrama Secuencia: Crear contrato

3.3.4- Crear reunión

Es posible la asignación de reuniones a los diferentes inmuebles que un usuario registrado tiene dados de alta en la base de datos como muestra la Ilustración 11.

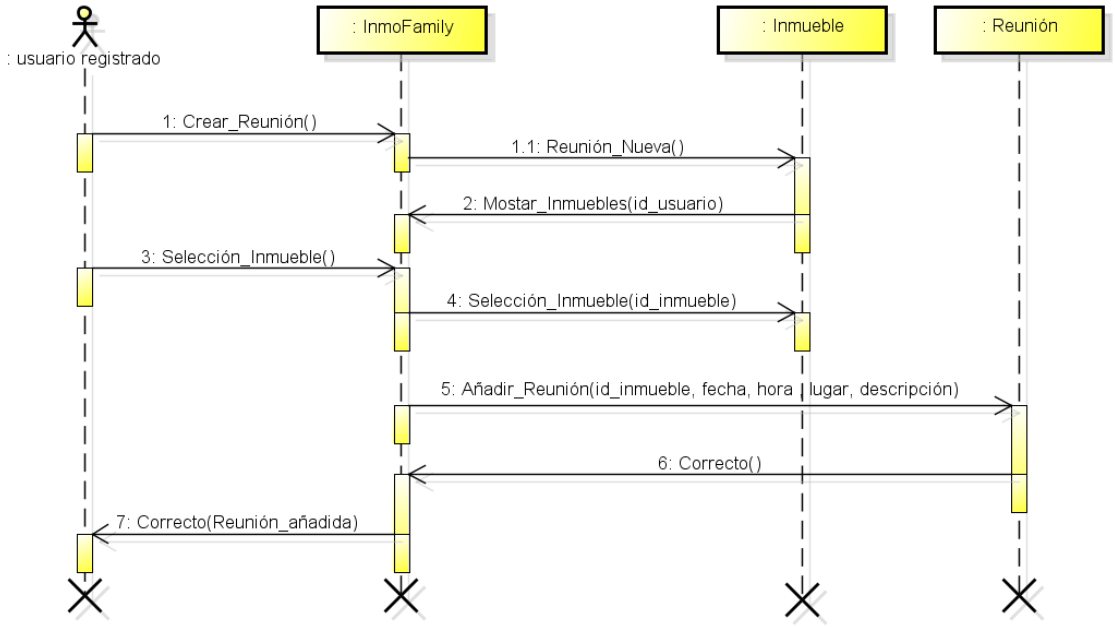

Ilustración 11: Diagrama Secuencia: Crear reunión

4- Diseño

4.1- Arquitectura de tres capas

El diseño sigue una arquitectura de tres capas con un reparto claro de funciones, una capa de presentación, una de lógica de negocio y una última de persistencia de datos.

Cada una de las capas es totalmente independiente al resto. Esto se hace para separar las tareas a realizar por cada una de ellas y dividir la solución final del proyecto, posibilitando avanzar en el proyecto en varios frentes a la vez.

Será necesaria una comunicación entre las capas con la información requerida por cada una. Con esta arquitectura se pretende conseguir una alta cohesión y un bajo acoplamiento para facilitar el desarrollo y mantenimiento de la aplicación.

En la imagen 12 podemos ver una representación gráfica de la arquitectura de tres capas.

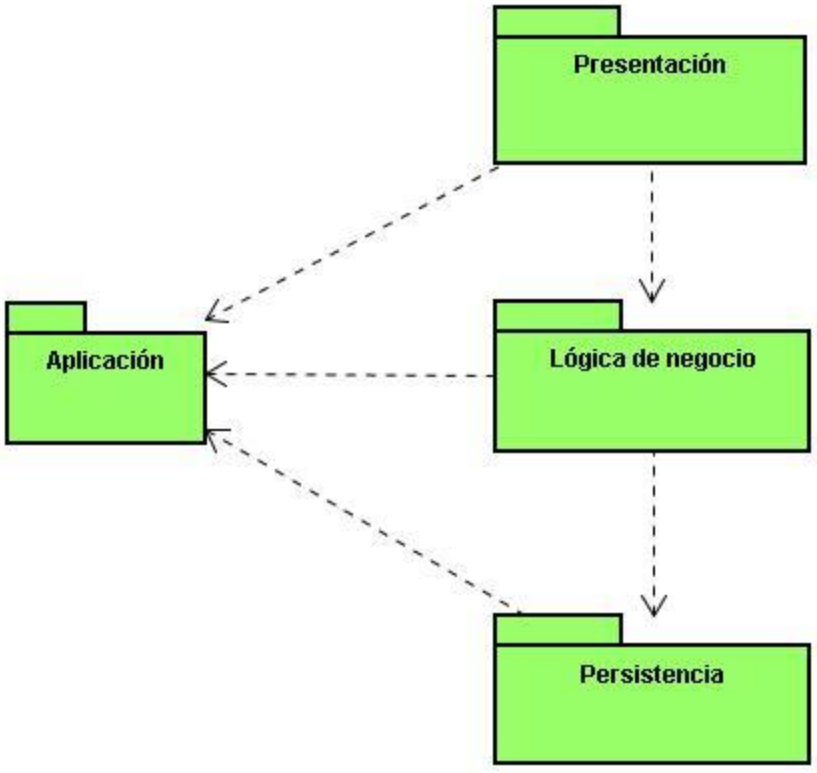

Ilustración 12: Arquitectura de tres capas

4.1.1- Capa de presentación

La capa de presentación es con la que interactúa el usuario. Contiene todas las funcionalidades descritas en los apartados anteriores, accediendo a la información a través de la capa de lógica de negocio. Tendrá un diseño sencillo para facilitar al usuario la usabilidad de la aplicación. El diseño de la capa de presentación dispondrá de un menú superior y la zona central para mostrar la información.

Ilustración 13: Capa de presentación; Página principal

La pantalla principal (Ilustración 13) del sitio web la podemos dividir en cuatro partes:

- La cabecera, consta de varios elementos. Logo de la web y saludo/invitación al registro. A continuación las imágenes muestran ejemplos de conexión sin registro y con él (Ilustraciones 14 y 15 respectivamente).

Ilustración 14: Cabecera, sin login

Ilustración 15: Cabecera, con login

- El menú superior, que dependiendo del rol del usuario tendrá más o menos opciones, para el manejo de las funcionalidades de la aplicación ((Ilustraciones 16 y 17 respectivamente)):

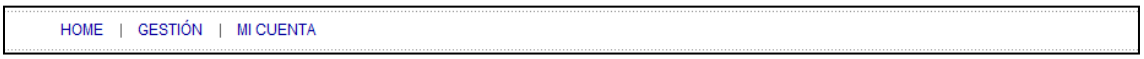

Ilustración 16: Menú sin login

Ilustración 17: Menú con login

- En la parte central de la página (Ilustración 18) se haya la zona de información, donde se publicará el contenido de 5 inmuebles disponibles aleatoriamente:

Ilustración 18: Cuerpo de la página web

- Por último, en la zona de pie de página (Ilustración 19), se muestran diferentes opciones de ayuda al usuario (estos enlaces no son funcionales en esta versión):

Ilustración 19: Pie de la página web

4.1.2- Capa de lógica de negocio

La capa de lógica de negocio es la que contiene los componentes de software que implementan el comportamiento especificado en la fase anterior. Esta capa debe realizar comunicación con otras capas, con la de presentación para dar respuesta a las solicitudes de información y a la de persistencia para hacer consultas de información.

Toma información de la base de datos según las necesidades del usuario y del sistema. También recogerá información suministrada por el usuario mediante la cada de presentación y la almacenará en la base de datos.

Los componentes de software que implementan el comportamiento de la solución vienen definidos por el conjunto de archivos que forman la aplicación (Ilustración 20). A través de código en lenguaje PHP contenido en estos archivos es posible definir el funcionamiento concreto de cada una de las partes de la aplicación.

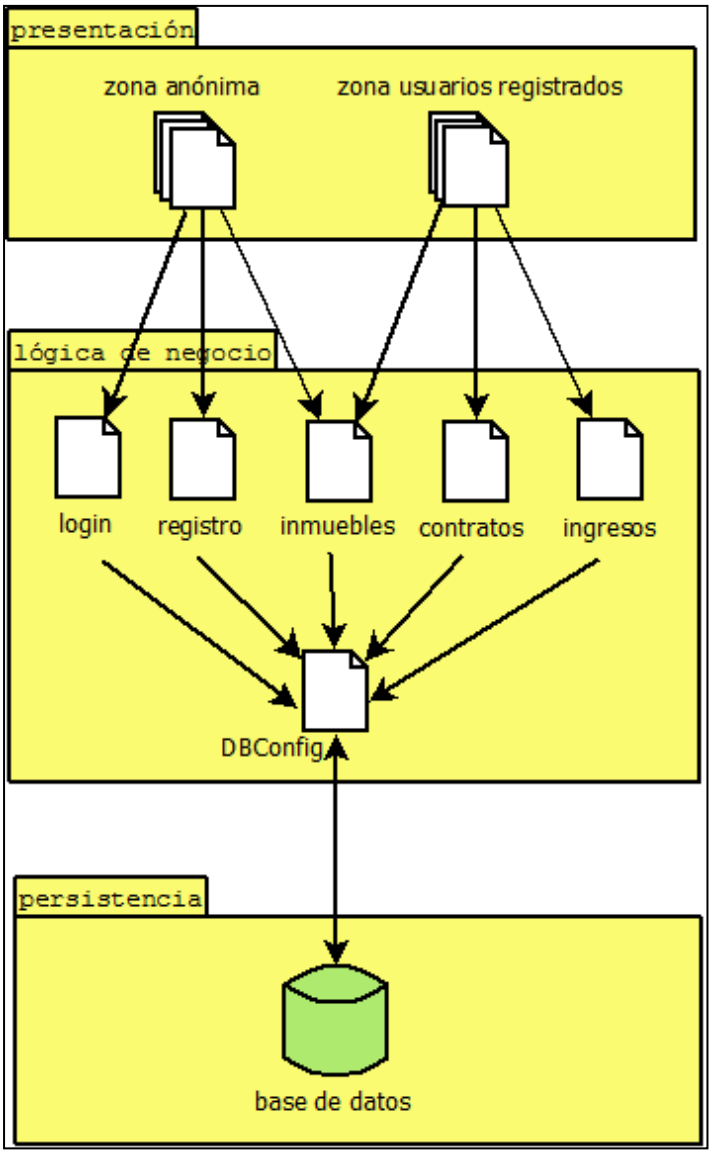

Ilustración 20: Módulos de la aplicación

El módulo de lógica de negocio está organizado en varios componentes (Ilustración 20). El analizador de *login* será el encargado de comprobar si el usuario está registrado correctamente y permitir la carga de información de la base de datos; para luego redirigir al usuario de la zona anónima a su zona privada como usuario registrado. El componente *registro*, permitirá crear un nuevo usuario para acceder al área privada.

El módulo *inmueble* se comportará de forma diferente según el usuario que acceda. Mostrará los inmuebles libres para su alquiler para un usuario anónimo, mientras que mostrará los inmuebles propios para un usuario registrado. El resto de componentes como contratos e ingresos sólo son accesibles desde un usuario registrado.

Cuando alguno de estos ficheros necesite acceder a la base de datos, lo hará a través de *DBConfig*. Este componente surge de aplicar el patrón fachada. Este patrón consiste en utilizar la clase *DBConfig* para todas las operaciones que interactúen con la base de datos. El uso de este patrón es muy útil para actualizar cambios en la base de datos como pueden ser el tipo de base de datos (MySQL, SQL Server, etc.) o el nombre de la base de datos por ejemplo.

4.1.3- Capa de persistencia

La capa base de la aplicación será la formada por los datos. La base de datos está compuesta por las tablas del diagrama Entidad-Relación de la Ilustración 21 mostrada a continuación.

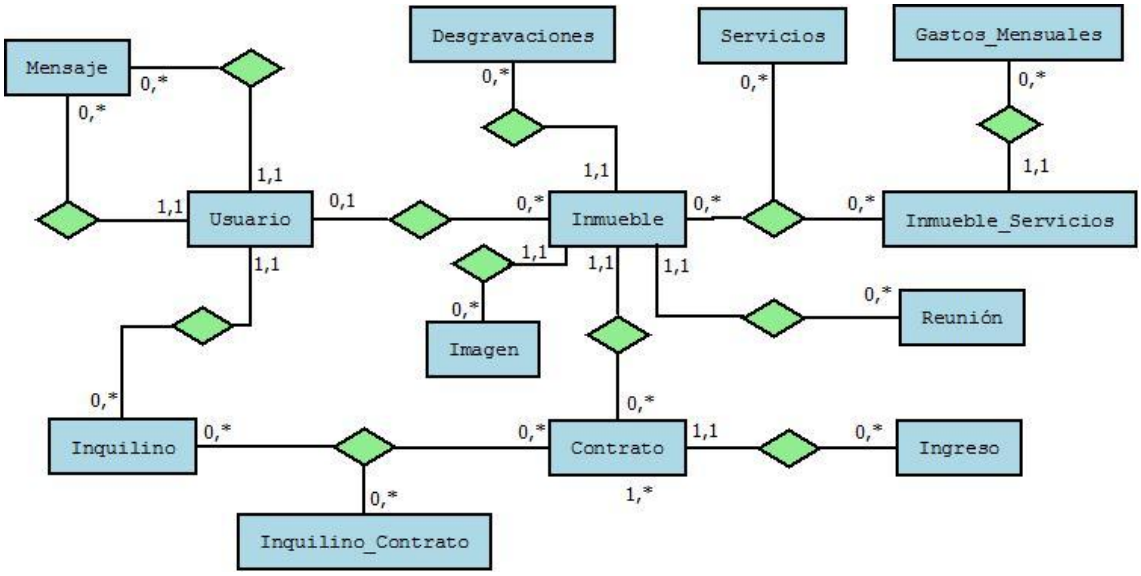

Ilustración 21: Capa de persistencia

5- Implementación

Para la elaboración de esta aplicación deseamos crear contenidos dinámicos para la total configuración. Para ello la base de la aplicación será código HTML. La aplicación utilizará lenguaje PHP acompañado de una base de datos MySql. Estas tecnologías serán maquetadas a través de lenguaje Css y JavaScript. Para el aumento del dinamismo y usabilidad de la aplicación se ha elegido también la tecnología Ajax.

A continuación paso a enumerar y describir las diferentes tecnologías utilizadas en el desarrollo de la aplicación.

5.1- Tecnologías

5.1.1- Nivel de presentación

HTML

El HTML, Hyper Text Markup Language, o en español lenguaje de marcación de hipertexto, es el lenguaje de marcas de texto utilizado normalmente en World Wide Web (WWW). Fue creado por el físico nuclear Tim Berners-Lee en 1986, a partir de dos herramientas preexistentes: el concepto de hipertexto y el SGML. El concepto de hipertexto, también conocido como link o ancla, permite conectar dos elementos entre sí. Por otro lado SGML, acrónimo de Standard Generalized Markup Language, es un lenguaje estándar de marcación general que sirve para colocar etiquetas o marcas en un texto que indique como debe mostrarse.

HTML es una sintaxis que se utiliza para definir la estructura y ubicación de los elementos que se quieren mostrar en una página Web. Esta sintaxis también es capaz de definir relaciones entre componentes de un sitio Web gracias al uso de hipervínculos. La interpretación de los documentos HTML suelen realizarla los navegadores Web, como por ejemplo: Mozilla Firefox, o Microsoft Internet Explorer.

El lenguaje HTML puede ser creado y editado con cualquier editor de textos básico, como puede ser el Bloc de Notas de Windows (o Notepad), o cualquier otro editor que admita texto sin formato como GNU Emacs, Microsoft Wordpad, Notepad++... Existen además, otras herramientas más complejas para la realización de sitios Web o edición de código HTML, como por ejemplo Microsoft FrontPage, Macromedia Dreamweaver, Komodo...

CSS

El termino CSS, acrónimo de Cascading Style Sheets, en español hojas de estilo en cascada, hace referencia al lenguaje formal usado para definir la presentación de documentos estructurados escritos en HTML, XML y XHTML por extensión. El World Wide Web Consortium (W3C) es el organismo encargado de formular la especificación de las hojas de estilo que servirán como estándar para los navegadores. La idea que se reside tras el desarrollo de CSS es separar la estructura de un documento de su presentación.

Ventajas derivadas de la utilización de CSS

• *Aumento de la legibilidad del documento HTML.* El documento es más fácil de leer y editar puesto que en él no se mezclan estructura y descripciones de estilo. Además, si no se utiliza la modalidad de CSS en línea, consistente en introducir la definición de estilo dentro del mismo documento de estructura, se reduce considerablemente el tamaño del documento.

• *Disminución del tiempo necesario para cambiar el aspecto global del sitio Web.* El control centralizado de la presentación de un sitio Web permite agilizar de forma considerable la actualización del mismo.

• *Aumento de la accesibilidad.* Los Navegadores permiten a los usuarios especificar y aplicar al sitio Web su propia hoja de estilo local, con lo que se aumenta considerablemente la accesibilidad. Por ejemplo, personas con deficiencias visuales pueden configurar su propia hoja de estilo para aumentar el tamaño del texto o remarcar más los enlaces.

• *Aumento de disponibilidad.* Una página puede disponer de diferentes hojas de estilo según el dispositivo que la muestre o incluso a elección del usuario. Por ejemplo, para ser impresa, mostrada en un dispositivo móvil o ser "leída" por un sintetizador de voz.

JavaScript

JavaScript es un lenguaje interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java.

Al contrario que Java, JavaScript no es un lenguaje orientado a objetos propiamente dicho, ya que no dispone de Herencia, es más bien un lenguaje basado en prototipos, a que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad. Todos los navegadores interpretan el código JavaScript integrado dentro de las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del DOM.

El lenguaje fue inventado por Brendan Eich en la empresa Netscape Communications, que es la que fabricó los primeros navegadores web comerciales.

Tradicionalmente, se venía utilizando en páginas web HTML, para realizar tareas y operaciones en el marco de la aplicación únicamente cliente, sin acceso a funciones del servidor. JavaScript se ejecuta en el agente de usuario al mismo tiempo que las sentencias van descargándose junto con el código HTML.

Los autores inicialmente lo llamaron Mocha y más tarde LiveScript pero fue rebautizado como JavaScript en un anuncio conjunto entre Sun Microsystems y Netscape, el 4 de diciembre de 1995. En 1997 los autores propusieron JavaScript para que fuera adoptado como estándar de la (European Computer Manufacturers' Association) ECMA, que a pesar de su nombre no es europeo sino internacional, con sede en Ginebra. En junio de 1997 fue adoptado como un estándar ECMA, con el nombre de ECMAScript. Poco después también lo fue como un estándar ISO.

5.1.2- Nivel de aplicación

Php

PHP es un lenguaje de programación usado generalmente para la creación de contenido para sitios web. PHP es un acrónimo recurrente que significa "PHP Hypertext Pre-processor" (inicialmente PHP Tools, o, Personal Home Page Tools), y se trata de un lenguaje interpretado usado para la creación de aplicaciones para servidores, o creación de contenido dinámico para sitios web. Últimamente también para la creación de otro tipo de programas incluyendo aplicaciones con interfaz gráfica.

El fácil uso y la similitud con los lenguajes más comunes de programación estructurada, como C y Perl, permiten a la mayoría de los programadores experimentados crear aplicaciones complejas con una curva de aprendizaje muy suave.

Su interpretación y ejecución se da en el servidor, en el cual se encuentra almacenado el script, y el cliente sólo recibe el resultado de la ejecución. Cuando el cliente hace una petición al servidor para que le envíe una página web, generada por un script PHP, el servidor ejecuta el intérprete de PHP, el cual procesa el script solicitado que generará el contenido de manera dinámica, pudiendo modificar el contenido a enviar, y regresa el resultado al servidor, el cual se encarga de regresárselo al cliente.

Además es posible utilizar PHP para generar archivos PDF, Flash, así como imágenes en diferentes formatos, o enviar emails, entre otras cosas.

Permite la conexión a diferentes tipos de servidores de bases de datos tales como MySQL, Postgres, Oracle, ODBC, DB2, Microsoft SQL Server, Firebird y SQLite; lo cual permite la creación de Aplicaciones web muy robustas.

PHP también tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos tales como UNIX (y de ese tipo, como Linux), Windows y Mac OS X, y puede interactuar con los servidores de web más populares ya que existe en versión CGI, módulo para Apache, e ISAPI.

Aunque su creación y desarrollo se da en el ámbito de los sistemas libres, bajo la licencia GNU, existe además un IDE comercial llamado Zend Optimizer.

Los principales usos del PHP son los siguientes:

- Programación de páginas web dinámicas, habitualmente en combinación con el motor de base datos MySQL, aunque cuenta con soporte nativo para otros motores, incluyendo el estándar ODBC, lo que amplía en gran medida sus posibilidades de conexión.
- Programación en consola, al estilo de Perl o Shell scripting.
- Creación de aplicaciones gráficas independientes del navegador, por medio de la combinación de PHP y GTK (GIMP Tool Kit), lo que permite desarrollar aplicaciones de escritorio en los sistemas operativos en los que está soportado.

Entre las ventajas de PHP podemos encontrar:

- Es un lenguaje multiplataforma.
- Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL.

- Leer y manipular datos desde diversas fuentes, incluyendo datos que pueden ingresar los usuarios desde formularios HTML.
- Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados ext's o extensiones).
- Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite las técnicas de Programación Orientada a Objetos.
- Permite crear los formularios para la web.
- Biblioteca nativa de funciones sumamente amplia e incluida.
- No requiere definición de tipos de variables ni manejo detallado del bajo nivel.

Ajax

Acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

Ajax es una tecnología asíncrona, en el sentido de que los datos adicionales se requieren al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. JavaScript es el lenguaje interpretado (scripting language) en el que normalmente se efectúan las funciones de llamada de Ajax mientras que el acceso a los datos se realiza mediante XMLHttpRequest, objeto disponible en los navegadores actuales. En cualquier caso, no es necesario que el contenido asíncrono esté formateado en XML.

Ajax es una técnica válida para múltiples plataformas y utilizable en muchos sistemas operativos y navegadores, dado que está basado en estándares abiertos como JavaScript y Document Object Model (DOM).

Ajax no constituye una tecnología en sí, sino que es un término que engloba a un grupo de éstas que trabajan conjuntamente.

Problemas e Inconvenientes

- Las páginas con AJAX son más difíciles de desarrollar que las páginas estáticas.
- Las páginas creadas dinámicamente mediante peticiones sucesivas AJAX, no son registradas de forma automática en el historial del navegador, así que haciendo clic en el botón de "volver" del navegador, el usuario no será devuelto a un estado anterior de la página, en cambio puede volver a la última página que visitó. Soluciones incluyen el uso de IFrames invisible para desencadenar cambios en el historial del navegador y el cambio de la porción de anclaje de la dirección (después de un #).
- Los motores de búsquedas no entienden JavaScript. La información en la página dinámica no se almacena en los registros del buscador.
- Hay problemas usando Ajax entre nombres de dominios. Eso es una función de seguridad.
- El sitio con Ajax usa más recursos en el servidor. Recomendación: sólo usar las peticiones necesarias en Ajax, no desarrollar todo el sitio en AJAX. Con esto garantizamos menos recursos del servidor.
- Es posible que páginas con Ajax no puedan funcionar en teléfonos móviles, PDA u otros aparatos. Ajax no es compatible con todo el software para ciegos u otras discapacidades.

5.1.3- Nivel de persistencia

MySql

MySQL es un sistema de gestión de base de datos, multihilo y multiusuario. MySQL AB desarrolla MySQL como software libre en un esquema de licenciamiento dual. Por un lado lo ofrece bajo la GNU GPL, pero, empresas que quieran incorporarlo en productos privativos pueden comprar a la empresa una licencia que les permita ese uso. Está desarrollado en su mayor parte en ANSI C.

Al contrario de proyectos como el Apache, donde el software es desarrollado por una comunidad pública, y el copyright del código está en poder del autor individual, MySQL está poseído y patrocinado por una empresa privada, que posee el copyright de la mayor parte del código. Esto es lo que posibilita el esquema de licenciamiento anteriormente mencionado.

Además de la venta de licencias privativas, la compañía ofrece soporte y servicios. Para sus operaciones contratan trabajadores alrededor del mundo que colaboran vía Internet. MySQL AB fue fundado por David Axmark, Allan Larsson, y Michael Widenius.

SQL, acrónimo de Lenguaje de Consulta Estructurado, fue comercializado por primera vez en 1981 por IBM, el cual fue presentado a ANSI y desde ese entonces ha sido considerado como un estándar para las bases de datos relacionales. Desde 1986, el estándar SQL ha aparecido en diferentes versiones como por ejemplo: SQL: 92, SQL: 99, SQL: 2003.

MySQL es una idea originaria de la empresa opensource MySQL AB establecida inicialmente en Suecia en 1995 y cuyos fundadores son David Axmark, Allan Larsson, y Michael "Monty" Widenius. El objetivo que persigue esta empresa consiste en que MySQL cumpla el estándar SQL, pero sin sacrificar velocidad, fiabilidad o usabilidad.

Michael Widenius en la década de los 90 trató de usar mSQL para conectar las tablas usando rutinas de bajo nivel ISAM, sin embargo, mSQL no era rápido y flexible para sus necesidades. Esto lo conllevó a crear una API SQL denominada MySQL para bases de datos muy similar a la de mSQL pero más portable.

Su popularidad como aplicación web está muy ligada a PHP, que a menudo aparece en combinación con MySQL. MySQL es una base de datos muy rápida en la lectura cuando utiliza el motor no transaccional MyISAM, pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación. En aplicaciones web hay baja concurrencia en la modificación de datos y en cambio el entorno es intensivo en lectura de datos, lo que hace a MySQL ideal para este tipo de aplicaciones.

5.2- Descripción de la implementación

Desde un punto de vista del diseño, queremos que todas las páginas, o la mayoría, de nuestro portal web tengan una cierta estructura de contenido y maquetación común.

En un primer lugar, lo que se podría pensar es que tendremos diferentes partes a repetir en cada página. Para ello dividiremos esta información en ficheros, para poder después incluirlos en cada una de las páginas en que sea necesario.

Con esta estructura, facilitaremos la modificación y mantenimiento del portal, de esta forma que modificando solo un fichero podríamos hacer los cambios que afectan a todas las páginas que lo incluyen.

Todo no es tan bueno como parece, ya que si procedemos a cambiar el nombre de alguno de estos ficheros raíz, tendremos que realizar el cambio en todo el proyecto. Para facilitarnos esta acción existen aplicaciones de refactorización para remodelar el código de una forma sencilla y eficaz, evitando tener que realizarlo a mano.

Todos los ficheros de la aplicación incluyen ciertos otros ficheros contenidos en los directorios page y php. Estos son los siguientes (Ilustración 22):

- DBConfig.php: para la disposición de las funciones de acceso y configuración de MySql.
- Functions.php: contiene funciones varias como validar usuario, cambiar formatos fecha, comprobar datos…
- headAll.php: contenido inicial de código HTML.
- head.php: cabecera del fichero HTML.
- Header.php: En ella se dibuja el menú de la aplicación.
- Countdbb.php: contador de visitas.
- Footer.php: es el pie de la página.

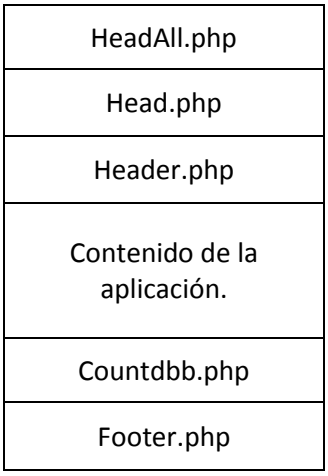

Ilustración 22: Estructura de la página web

A pesar de esta estructuración principal, para aumentar la modularidad de la programación se han dividido a su vez, cada página en diferentes módulos (también capas) reutilizables. Así cada componente puede ser reutilizado en más partes del proyecto y no sólo en una página web de toda la aplicación. Por ello he tratado de hacer una estructuración modular para evitar ficheros interminables e ilegibles, facilitando así la reutilización de código.

5.2.1- Parte pública

La parte pública de la aplicación está formada por ficheros como index.php, registro.php y login.php. El resto de ficheros tienes un control de validación para no permitir acceder a usuarios sin privilegios.

index.php

Paso a describir index.php con las siguientes imágenes (Ilustración 23). En la primera no se ha abierto sesión por ningún usuario y por lo tanto aparece un menú reducido con y un enlace a acceso de propietarios o para el registro en la aplicación.

Ilustración 23: Index.php sin loggeo

A continuación (Ilustración 24) se muestra el contenido principal de la aplicación. El contenido genera una búsqueda aleatoria de cinco inmuebles disponibles y los muestra junto con una breve descripción. Es posible acceder a información más detallada de cada piso haciendo clic sobre el inmueble.

Ilustración 24: Index.php zona de contenido dinámico

Finalmente se muestra el contenido del pie de página (Ilustración 25) que no variará en ningún momento durante el transcurso de trabajo de la aplicación.

Ilustración 25: Index.php pie web de la aplicación

login .php

Es la página encargada de loggear a los usuarios, internamente abre una sesión por usuario, es necesario introducir nick y contraseña, también existe un botón para que los usuarios puedan registrarse (Ilustración 26):

Ilustración 26: Login.php

registro.php

Pantalla para el registro de nuevos usuarios (Ilustración 27), estos deberán aceptar las políticas de protección de datos ofrecidas por la aplicación, e insertar los datos marcados con * asterisco rojo.

Ilustración 27: Registro.php

5.2.2- Parte privada

A continuación paso a mostrar el código que inicia a un usuario en la parte privada de la aplicación. Es necesario recordar que es imprescindible estar registrado en la aplicación con antelación para poder acceder a esta zona.

El fichero valida_sesion.php (Ilustración 28) es el encargado de crear la sesión que durará el tiempo que el usuario permanezca en conectado, su código se muestra a continuación y podremos observar los mensajes, también de error que devuelve la aplicación tras el intento de logeo:

```php
<?php
require_once("php/require.php");

//datos para establecer la conexion con la base de mysql.
$db = new DBConfig();
$db->config();
$db->conn();

if( $_POST['nick'] != '' && $_POST['password'] != ''){
  $nick = $_POST['nick'];
  $password = $_POST['password'];
  $password = md5($password);
  $result = mysql_query('SELECT id_usuario,nick, rol FROM usuario
                         WHERE nick=\''.$nick.'\'
                         AND password = \''.$password.'\'');
  if($row = mysql_fetch_array($result)){
    if($row["nick"] == $nick){
      $_SESSION['id_usuario'] = $row['id_usuario'];
      $_SESSION['logeo'] = 'true';
      $_SESSION['logeosms'] = '';
      //Ingreso exitoso, ahora sera dirigido a la pagina principal.
      header("Location: index.php");
    }else{
      $_SESSION['logeo'] = 'false';
      $_SESSION['logeosms'] = 'Password incorrecto';
      header("Location: login.php");
    }
  }else{
    $_SESSION['logeo'] = 'false';
    $_SESSION['logeosms'] = 'Usuario no existente en la base de datos o Password incorrecto';
    header("Location: login.php");
  }
  mysql_free_result($result);
}else{
    $_SESSION['logeo'] = 'false';
    $_SESSION['logeosms'] = 'Debe especificar un usuario y password';
    header("Location: login.php");
}
$db->close();
?>
```

Ilustración 28: Valida_sesion.php

En cada una de las páginas se ejecutará esta parte de código. Este código se encargará de comprobar que la sesión está abierta y el usuario existe. En caso positivo le muestra un saludo, y en caso negativo le invita a registrarse o acceder. Como muestra el código siguiente (Ilustración 29):

```
<?php
if (isLoggedIn()) {

    $conexion=mysql_connect("XXXXXXX","XXXX","XXXXXXXX") or
      die("Problemas en la conexion");

    mysql_select_db("tubi",$conexion) or die("Problemas en la selección de la base de datos");

    $registro=mysql_query("select id_usuario, nick from usuario
                           where id_usuario='$_SESSION[id_usuario]'",
                    $conexion) or die("Error:".mysql_error());

    if ($reg=mysql_fetch_array($registro))
    {
      $nick=$reg['nick'];
      $_SESSION['id_usuario']=$reg['id_usuario'];
      echo "<li class=\"first\">Hola <a href=\"#\">".$nick."</a> </li>";
      echo "<li class=\"desconectar\"><a href=\"cierra_sesion.php\">Desconectar</a></li>";
    }
    else
    {
      echo "<li class=\"propietarios\"><a href=\"login.php\">Acceso propietarios</a></li>";
      echo "<li><a class=\"propietarios\" href=\"registro.php\">Regístrate</a></li>";
    }

}else
{
  echo "<li class=\"propietarios\"><a href=\"login.php\">Acceso propietarios</a></li>";
  echo "<li><a class=\"propietarios\" href=\"registro.php\">Regístrate</a></li>";
}
?>
```

Ilustración 29: Comprobación de login

Por ejemplo, en esta otra imagen (Ilustración 30) se puede apreciar como el usuario Roberto se ha loggeado y por lo tanto aparece su nombre “Hola Roberto”, un link para la desconectar al usuario y un submenú desdoblado con más opciones que al inicio.

Ilustración 30: Página con login

A partir de aquí paso a explicar las diferentes partes de la sección privada de la aplicación:

5.2.2.1- Gestión de inmuebles

inmuebles.php

Principal parte de gestión. Aquí podremos agregar todos los inmuebles que el usuario tenga en alquiler o desee alquilar. El diseño es muy simple y sencillo de entender, se muestra una lista con los inmuebles existentes y un enlace para añadir nuevos inmuebles. La intuición nos dice que podremos modificar los datos de nuestros inmuebles clicando sobre ellos, y así es. También se facilita una opción para eliminar, que podemos observar en la Ilustración 31.

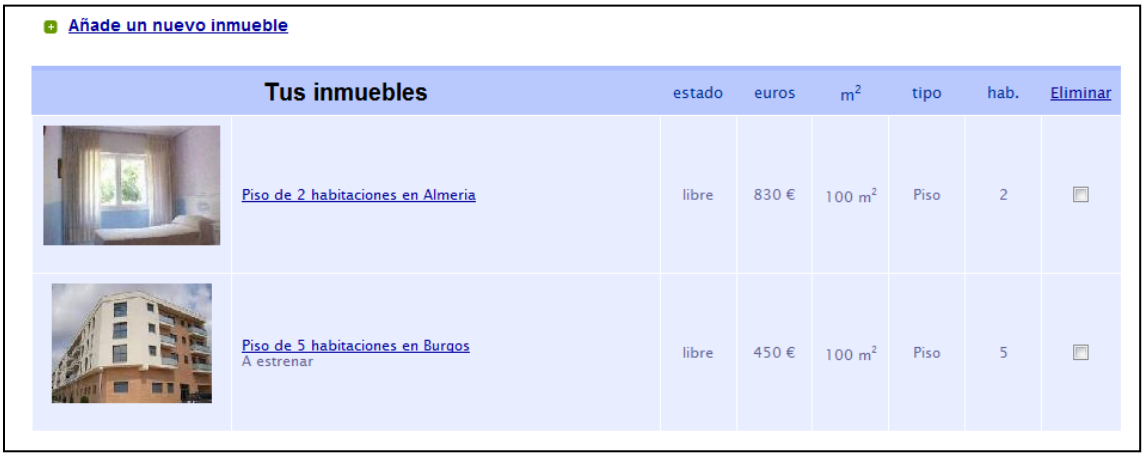

Ilustración 31: Inmuebles.php

anadirInmueble.php

Ventana para dar de alta un nuevo inmueble en la aplicación (Ilustración 32).

En esta página existen 4 módulos (capas), todos ellos independientes y programados en archivos php independientes para aumentar su reusabilidad.

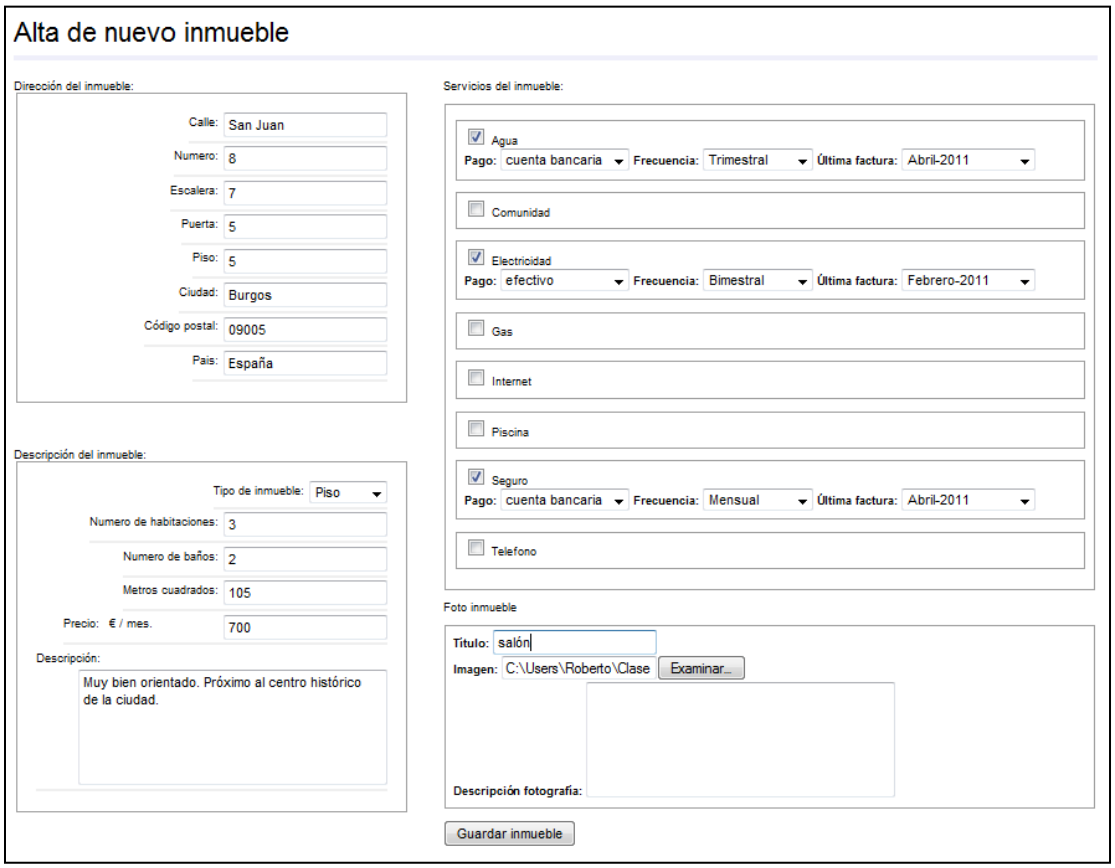

Ilustración 32: AnadirInmueble.php

Cabe destacar el uso de javascript para una visión más cómoda y dinámica de la interfaz en la capa de Servicios del inmueble, debido a que existen muchos datos a introducir y liberar da datos la pantalla. Con este código (Ilustración 33) se recogen y se despliegan los recuadros de servicios según el estado del checkbox, como se ha dicho anteriormente esto es posible gracias a que la web se ha desarrollado usando capas ( ) y javaScript.

```
<script type="text/javascript">
  function cambiarVisibilidad(clase, checkmuestra){
    var div = document.getElementById(clase);
    if(div.style.display == 'block'){
      div.style.display = 'none';
    }else if(div.style.display == 'none'){
      div.style.display = 'block';
    }
  }
</script>
```

Ilustración 33: Cambio visibilidad de los recuadros de servicios

5.2.2.2- Gestión de contratos

contratos.php

Para la gestión de inmuebles es necesario poder gestionar sus contratos, para ello es esta parte de la aplicación. Aquí (Ilustración 34) podremos visualizar crear, modificar y eliminar todos nuestros contratos.

Para aumentar el dinamismo en esta sección de la aplicación se ha utilizado la tecnología Ajax, que permite cargar asíncronamente partes de la web. Gracias a ello podemos ordenar los contratos clicando en la cabecera de cada columna sin ser necesario recargar la web entera, ahorrando con ello tiempos de espera y trabajo al servidor.

Ilustración 34: Contratos.php

Con este código (Ilustración 35) recogemos la columna seleccionada para ordenar:

```
if(isset($_GET['campo']) and isset($_GET['orden']) and isset($_GET['idusuario'])){
  $campo=$_GET['campo'];
  $orden=$_GET['orden'];
  $idusuario=$_GET['idusuario'];
}else{
  //por defecto
  $campo='calle';
  $orden='ASC';
  $idusuario=$_SESSION['id_usuario'];
}
```

Ilustración 35: Selección de columna a ordenar

Dibujamos las columnas dinámicamente con el código siguiente (Ilustración 36):

```
<th scope="col" class="finalizar" >
  <a href="#" onclick = "document.ckfinalizar.submit(); return false" title="Finalizar contratos"> Finalizar</a>
</th> <?php echo "\n"; ?>
<?php
$i=0;
while($i<=$nroItemsArray-1){
  if($campos[$i]==$campo){
    if($orden=="DESC"){
      $orden="ASC";
      $flecha="objetos/arrow_down.gif";
    }else{
      $orden="DESC";
      $flecha="objetos/arrow_up.gif";
    }
    ?>
    <th scope="col" class="encabezado_selec">
      <a onclick = "Ordenar('<?php echo $campos[$i]; ?>','<?php echo $orden; ?>','<?php echo $idusuario; ?>', 'listado',
      'contratos/listadoContratosVigentes.php')"><img src="<?php echo $flecha; ?>" /> <?php echo $cabecera[$i]; ?>
      </a>
    </th> <?php echo "\n"; ?>
    <?php
  }else{
    ?>
    <th scope="col">
      <a onclick = "Ordenar('<?php echo $campos[$i]; ?>','DESC','<?php echo $idusuario; ?>' , 'listado',
      'contratos/listadoContratosVigentes.php')"> <?php echo $cabecera[$i]; ?>
      </a>
    </th> <?php echo "\n"; ?>
    <?php
  }
  $i++;
}
  ?>
  <th scope="col" class="inquilino" >
    Inquilino
  </th> <?php echo "\n"; ?>
  <th scope="col" class="telefonos" >
    Teléfonos
  </th> <?php echo "\n"; ?>
  <th scope="col" class="modificar" >
    Modificar
  </th> <?php echo "\n"; ?>
</tr>
```

Ilustración 36: Pintar cabecera de columnas ordenadas

Y finalmente mediante un simple bucle recorreremos una consulta lanzada a la base de datos insertando una nueva fila a la tabla por cada registro encontrado.

contratoNuevo.php

Esta es una de las partes más complicadas debido al uso de tecnología Ajax, cada uno de los recuadros que se ven en la imagen que sigue, pertenece a un fichero que recargaremos asíncronamente mediante Ajax. Añadir también el uso de javaScript para la inserción de un calendario que facilita y ayuda en la inserción de fechas.

Según el inmueble seleccionado obtendremos una cantidad previa para el contrato, que podremos modificar. Iremos describiendo el contrato añadiendo los datos que nos piden los formularios de la aplicación.

Podremos asignar uno o varios inquilinos, que deberemos añadir mediante el recuadro inferior derecho y seleccionando en su correspondiente checkbox en el recuadro superior derecho.

Anotaremos también como se realizaran los pagos, tanto de la mensualidad del inmueble como de los servicios. Estos podrán ser efectivo, cuenta bancaria o pagados por el inquilino. Podemos visualizar un ejemplo en la Ilustración 37.

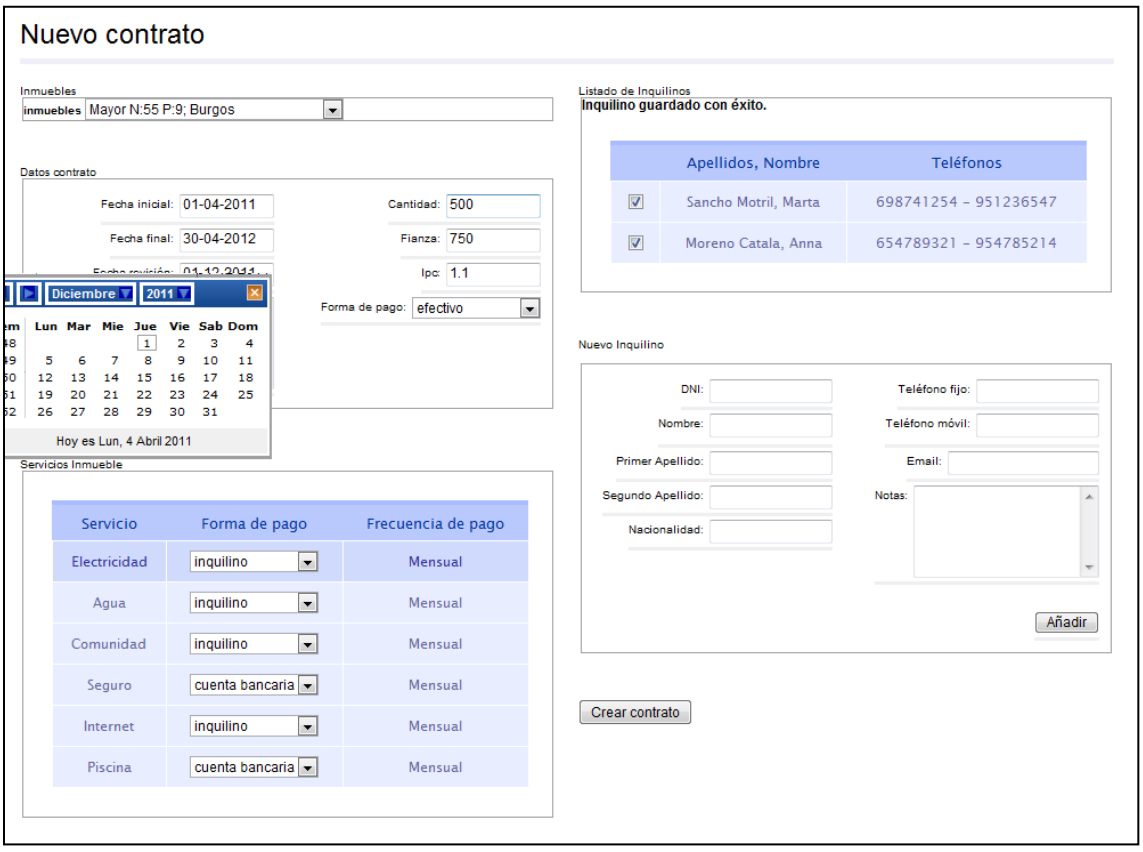

Ilustración 37: ContratoNuevo.php

Funcionamiento del objeto Ajax

En la siguiente imagen (Ilustración 38) podemos ver la función que se encarga de rellenar la lista de inquilinos que podremos añadir a un contrato, tras añadir un nuevo inquilino.

```
function enviarDatoInquilino(){
  //donde se mostrar lo resultados
  divResultado = document.getElementById('listaInquilinos');
  //valores de los inputs que se recogen del formulario
  var idusuario = document.formInquilinoNuevo.id_usuario.value;
  var nom = document.formInquilinoNuevo.nombre.value;
  var ape1 = document.formInquilinoNuevo.apellido1.value;
  var ape2 = document.formInquilinoNuevo.apellido2.value;
  var dni = document.formInquilinoNuevo.dni.value;
  var nac = document.formInquilinoNuevo.nacionalidad.value;
  var telfi = document.formInquilinoNuevo.telefono_fijo.value;
  var telmo = document.formInquilinoNuevo.telefono_movil.value;
  var emai = document.formInquilinoNuevo.email.value;
  var nota = document.formInquilinoNuevo.notasInq.value;
  var anadirinquilino = 0;
  //instanciamos el objetoAjax
  var ajax=Ajax();
  ajax.open("POST", "inquilinos/listaInquilinos.php", true);
  ajax.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded; charset=utf-8');
  ajax.onreadystatechange=function() {
    if (ajax.readyState==4) {
      //mostrar resultados en esta capa
      divResultado.innerHTML = ajax.responseText;
      limpiarFormularioGenerico("formInquilinoNuevo");
    }
  }
  //enviando los valores
  ajax.send("anadirinquilino="+anadirinquilino+"&idusuario="+idusuario+"&nom="+nom+"&ape1="+ape1+
  "&ape2="+ape2+"&dni="+dni+"&nac="+nac+"&telfi="+telfi+"&telmo="+telmo+"&emai="+emai+"&nota="+
  nota);
}
```

Ilustración 38: Función Ajax

Es necesaria la existencia de esta función (Ilustración 39) que crea el objeto Ajax.

```
function Ajax(){
  var xmlhttp=false;
  try {
    xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
  } catch (e) {
    try {
      xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
    } catch (E) {
      xmlhttp = false;
    }
  }
  if (!xmlhttp && typeof XMLHttpRequest!='undefined') {
    xmlhttp = new XMLHttpRequest();
  }
  return xmlhttp;
}
```

Ilustración 39: Función Ajax

5.2.2.3- Gestión de ingresos

ingresos.php

En ingresos (Ilustración 40) podremos observar dos listados diferenciados, los ingresos pendientes y los ingresos pagados. En las dos tablas se verán reflejados todos los ingresos junto con una breve descripción, con datos como dirección del inmueble, mensualidad a pagar, forma de pago, cantidad del ingreso.

Para el desarrollo de esta parte y aumentar la usabilidad se ha incluido la funcionalidad de ordenar las tablas según el contenido de sus columnas a través del uso de Ajax, se puede apreciar como por defecto se ordena por Dirección de inmueble ya que aparece una flecha roja indicándolo.

Ilustración 40: Ingresos.php

Los ingresos se pueden pagar y eliminar, permitiéndose hacer en varios a la vez (no solo de uno en uno), para aumentar la usabilidad, velocidad y comodidad de uso de la aplicación.

Para la generación de ingresos se ha creado un botón de generación de ingresos para los ingresos todavía no generados, lo ideal sería la existencia de un proceso Cron. Los procesos cron son servicios que incluyen algunos servidores web y permiten la autoejecución de acciones, es decir, crear tareas programadas. Gracias a este servicio no sería necesaria la existencia de este botón. Además este botón obliga al usuario (propietario) a crear sus propios ingresos, cuando debería hacerse automáticamente mediante unos procesos cron.

Muestro el código que se ejecuta tras pulsar el botón de generar ingresos con la imagen siguiente (Ilustración 41):

```
if (isset($_POST['generaringresosname'])){
  $db = new DBConfig();
  $db->config();
  $db->conn();
  $consulta=("SELECT id contrato. fecha inicio. fecha_fin,
              mes_ultimo_ingreso, cantidad, cobro
              FROM contrato
              WHERE estado='Vigente'");
  $consulta = mysql_query($consulta);
  while ($rows = mysql_fetch_array($consulta)){
    if(compararMesAnnio($rows['fecha_inicio'], date("Y-m-d")) <= '1'){
      $fultimafactura=$rows['mes_ultimo_ingreso'];
      if($rows['mes_ultimo_ingreso']=='0000-00-00'){
        $sql = "INSERT INTO ingreso (id_ingreso, id_contrato, mes_a_pagar,
        fecha_ingreso, valor_pagado, valor_factura, pagado, forma_pago, notas)
        VALUES (NULL, '$rows[id_contrato]', '$rows[fecha_inicio]',
        NULL , '0', '$rows[cantidad]', '0', '$rows[cobro]', NULL)";
        mysql_query($sql);
        $sql = "UPDATE contrato SET mes_ultimo_ingreso = '$rows[fecha_inicio]'
                WHERE id_contrato='$rows[id_contrato]'";
        mysql_query($sql);
        $fultimafactura=$rows['fecha_inicio'];
      }

      $fultimafactura=sumarmeses($fultimafactura.' 00:00:00', '1');
      list($anio,$mes,$dia)=explode("-",$fultimafactura);
      $fultimafactura = $anio.'-'.$mes.'-01';

      while(compararMesAnnio($fultimafactura, date("Y-m-d")) < '1' &&
            compararMesAnnio($fultimafactura, $rows['fecha_fin']) < '1'){
        $cantidad=$rows['cantidad'];
        $sql = "INSERT INTO ingreso (id_ingreso, id_contrato, mes_a_pagar,
          fecha_ingreso, valor_pagado, valor_factura, pagado, forma_pago, notas)
          VALUES (NULL, '$rows[id_contrato]', '$fultimafactura',
          NULL , '0', '$rows[cantidad]', '0', '$rows[cobro]', NULL)";
          mysql_query($sql);
          $fultimafactura=sumarmeses($fultimafactura.' 00:00:00', '1');
          list($anio,$mes,$dia)=explode("-",$fultimafactura);
          $fultimafactura = $anio.'-'.$mes.'-01';
      }
      $fultimafactura=restarmeses($fultimafactura.' 00:00:00', '1');
      $sql = "UPDATE contrato SET mes_ultimo_ingreso = '$fultimafactura'
              WHERE id_contrato='$rows[id_contrato]'";
      mysql_query($sql);
    }
  }
  $db->close();
}
```

Ilustración 41: Generar ingresos

5.2.2.4- Gestión de gastos mensuales

gastos.php

El funcionamiento de gastos mensuales es similar a ingresos, comentado en el apartado anterior. Por lo tanto muestro una única imagen (Ilustración 42) que demuestra la similitud externa y añado que el funcionamiento interno es muy parecido y sin importantes diferencias que destacar.

Ilustración 42: Gastos.php

5.2.2.5- Gestión de reuniones

reuniones.php

Cada uno de los inmuebles incluidos para su gestión tendrá la posibilidad de guardar reuniones con una breve descripción; fecha, hora, lugar, descripción. (Ejemplo, véase Ilustración 43)

Ilustración 43: Reuniones.php

5.2.2.6- Gestión de desgravaciones

desgravaciones.php

Como ocurre en el apartado anterior, cada uno de los inmuebles incluidos para su gestión tendrá la posibilidad de guardar las desgravaciones con una breve descripción, (véase Ilustración 44). De esta forma se facilitará contabilizar los desembolsos hechos por inmueble y se facilitará la contabilidad de los gastos.

Ilustración 44: Desgravaciones.php

5.2.2.7- Gestión inquilinos

inquilino.php

Todos los propietarios tienen un acceso fácil y sencillo a un listado de inquilinos, en el aparecerán todos los inquilinos que han dispuesto de sus servicios en algún inmueble. De esta forma será muy sencillo acceder a los datos personales de estos para ponerse en contacto con cada uno de ellos.

En esta página (Ilustración 45) he utilizado tecnología Ajax para mejorar el dinamismo y la comodidad de utilización de la aplicación. Esta parte está formada por tres capas principales que se regeneraran asíncronamente cuando sea necesario. Estas tres capas son la del listado, la generación de un inquilino nuevo y la de modificación de un inquilino existente.

Si creamos un nuevo inquilino se verán afectadas dos capas, la de inserción de datos y la de listado de datos. De esta forma cuando hacemos clic en añadir inquilino, se cogen los datos de la capa, se reinicia la capa y finalmente se redibuja la capa con la tabla con el nuevo inquilino. Para modificar un inquilino, haremos clic en modificar en la capa de listado y se recargará el formulario de la capa de modificación, cuando demos clic en actualizar, se recargará la capa listado con los cambios actualizados.

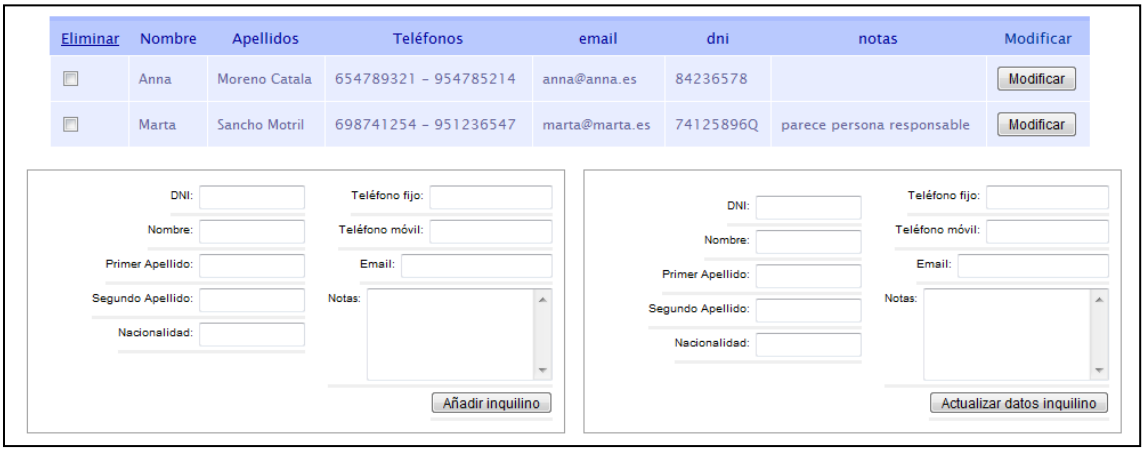

Ilustración 45: Inquilino.php

5.2.3- Otras funciones

subirImagen.php

Este archivo se encarga de subir de forma correcta los datos a la base de datos de la aplicación. Se realiza un doble almacenado por cada una de las imágenes. Una en tamaño grande mientras que el otro es una miniatura.

mostrarImagen.php

Este archivo será el encargado de mostrar las imágenes guardadas por subirImagen.php, es necesario que llamemos a este fichero mediante GET y con dos parámetros, el nombre del inmueble del que deseamos la imagen y el tamaño de imagen.

countbdd.php

Con este archivo llevamos un recuento de las visitas que tiene la web y la duración de estas. Esta información es almacenada en la base de datos.

popcalendar.js

Su labor es facilitar al usuario la forma de insertar fechas en la aplicación. Siempre que pulsemos sobre un campo en el cual sea necesario insertar una fecha, automáticamente aparecerá y nos dará un formato de fecha correcto para la aplicación. Visualmente tendrá un aspecto similar al de la Ilustración 46:

Ilustración 46: Popcalendar.js

6- Evaluación

En esta fase de evaluación concluye el ciclo de vida del proyecto y lo deja preparado para su uso. La fase de evaluación del proyecto medirá el nivel de calidad que ofrece la solución construida, donde se comprobará la corrección del código implementado. Se ha sometido a pruebas de validación del CSS y se han realizado verificaciones de todos los enlaces de la Web..

Prueba de validación de CSS

Puesto que se ha utilizado una hoja de estilo para la interfaz del portal web, se ha realizado una prueba de validación a través de la página web http://jigsaw.w3.org/css-validator/ y el resultado ha sido válido (Ilustración 47).

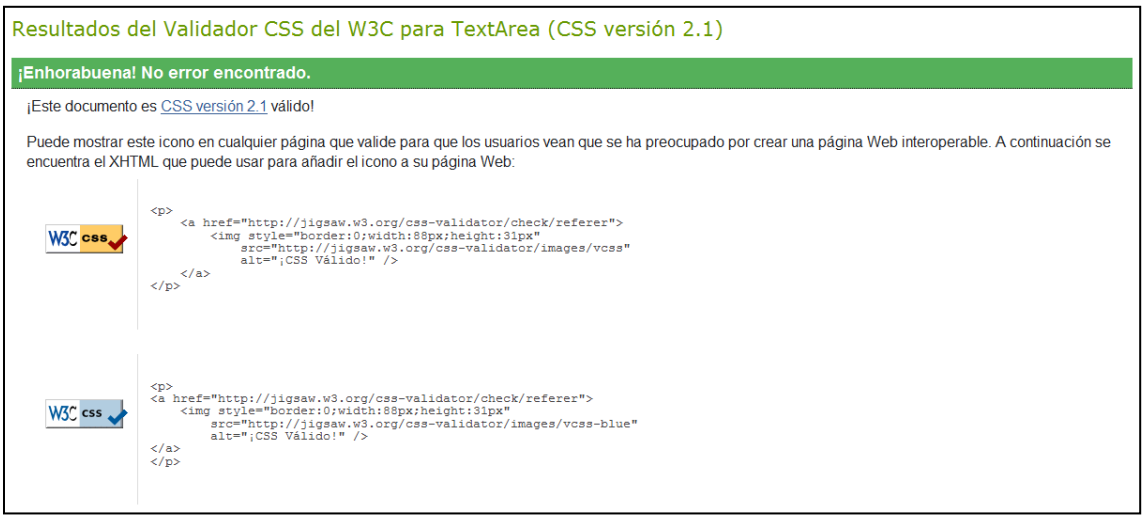

Ilustración 47: Prueba CSS

Prueba de validación de enlaces

Esta prueba para la validación de enlaces se ha realizado para comprobar que todos los enlaces de la aplicación están bien formados y que todos enlazasen con otra página, por lo que no existe ningún enlace roto en la misma.

Para ello se ha usado la siguiente página web http://validator.w3.org/checklink que busca errores en los links. El resultado ha sido válido (Ilustración 48).

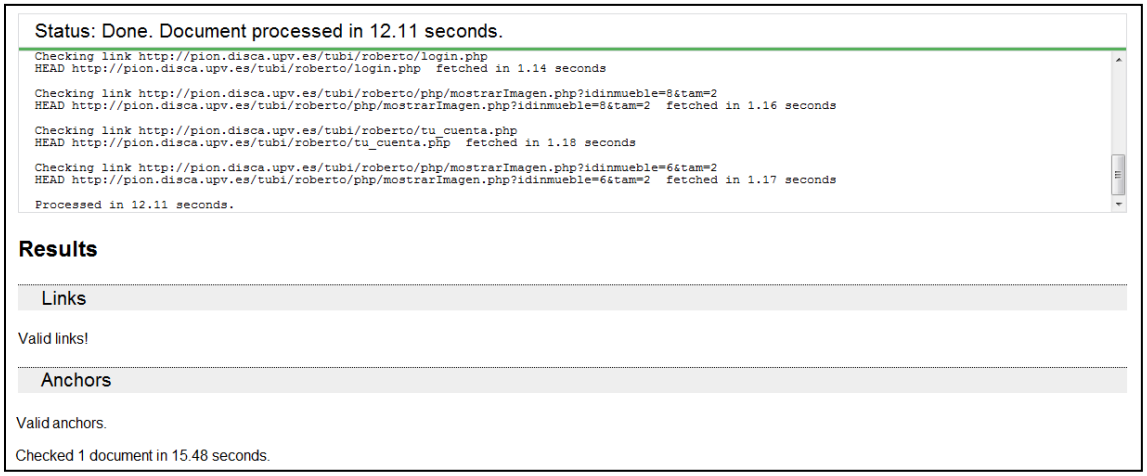

Ilustración 48: Prueba links

Prueba de exploradores

Se ha comprobado el correcto funcionamiento de la aplicación tanto en Microsoft Explorer, Mozilla Firefox y Google Chrome (Ilustraciones 49, 50 y 51 respectivamente). El mayor problema existió durante la situación de las capas en sus lugares correspondientes y demás número de elementos y estilos.

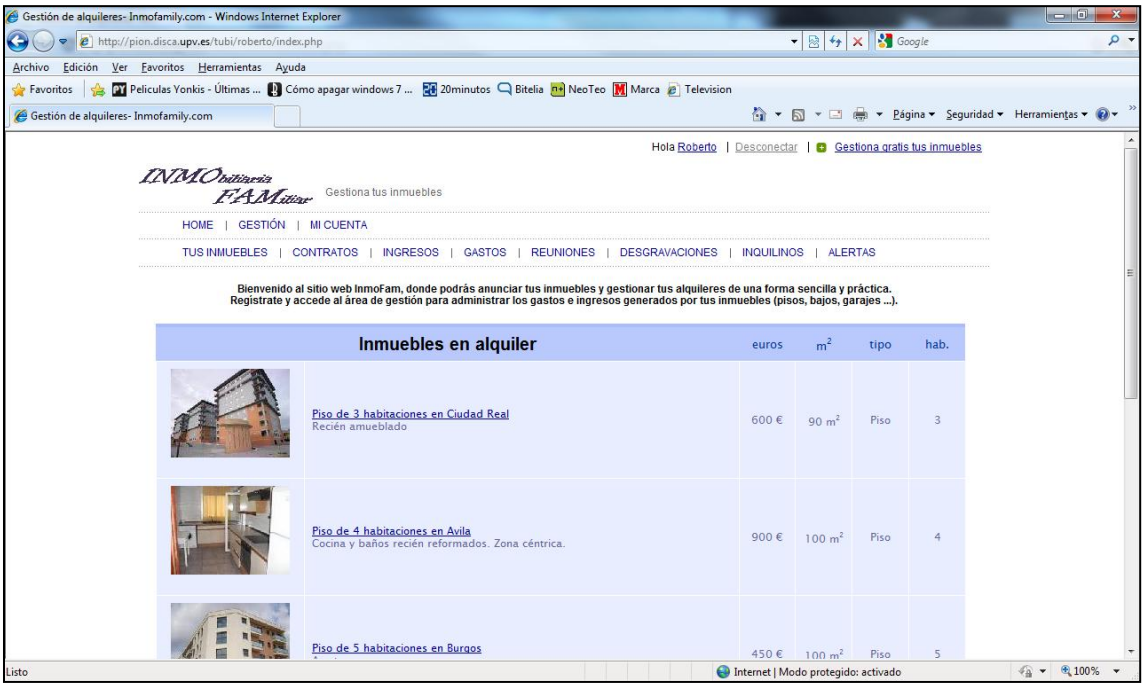

Ilustración 49: Explorer

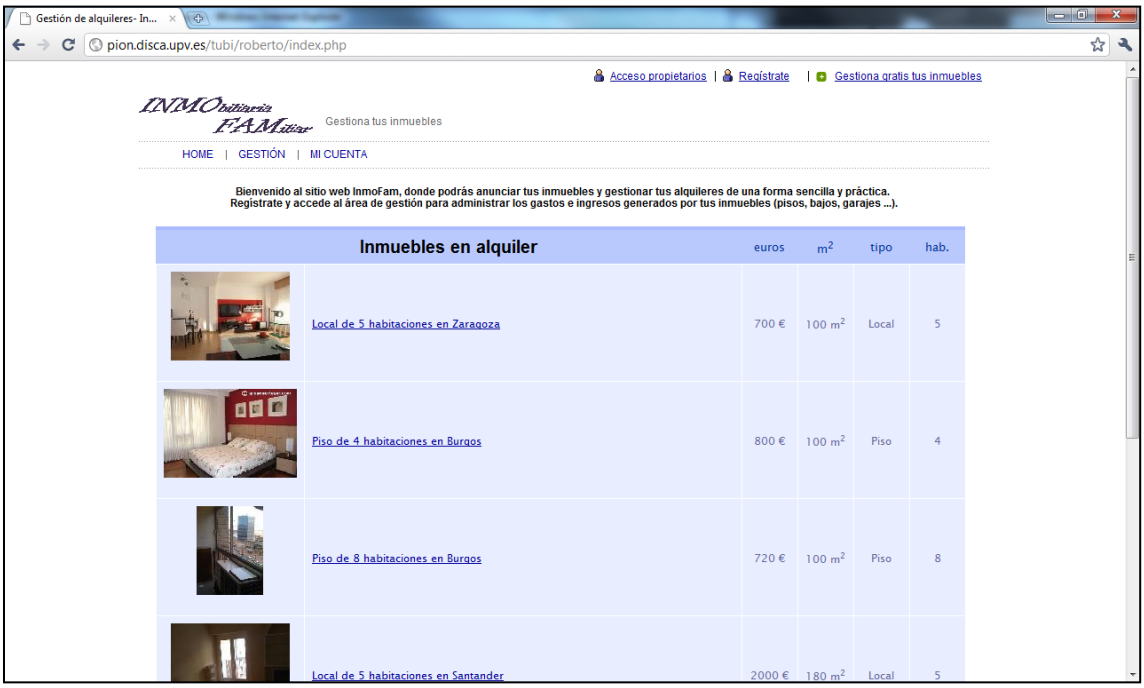

Ilustración 50: Chrome

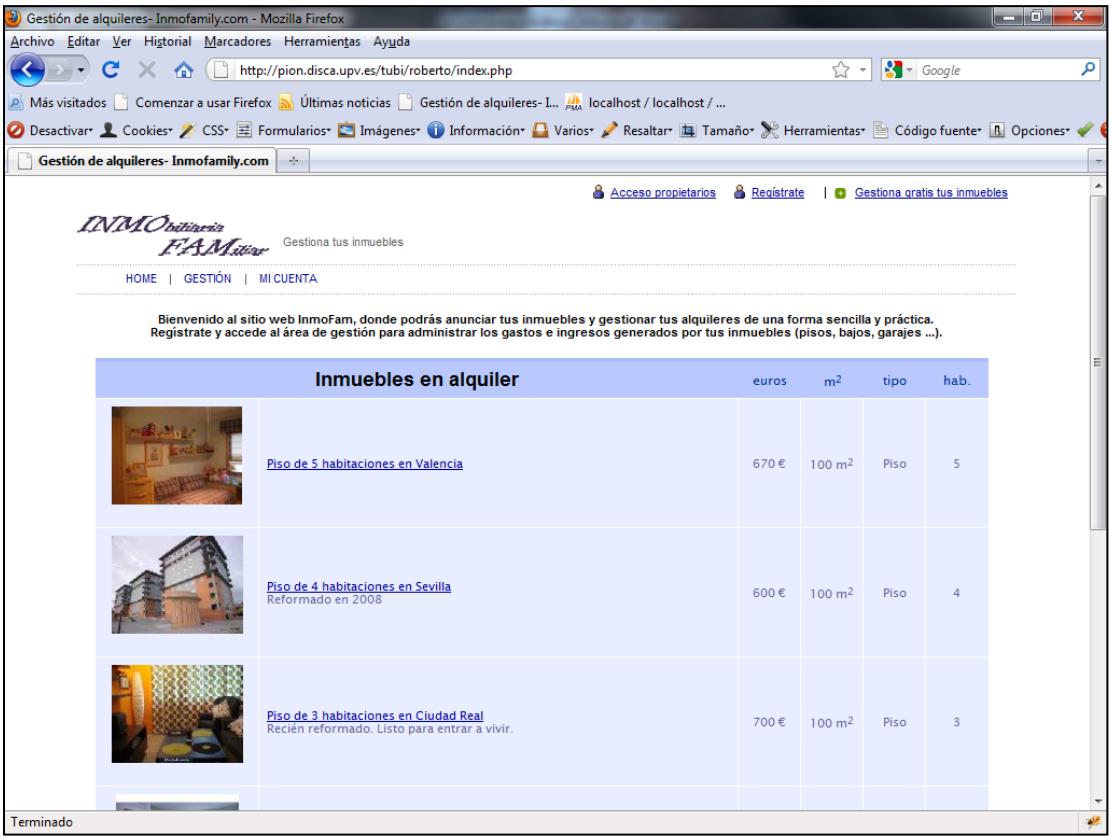

Ilustración 51: Mozilla

Prueba de seguridad de acceso

Para acceder a la aplicación como usuario registrado es necesario insertar los datos correctamente en una pantalla similar a la que sigue, es necesario haber hecho un registro previo del usuario. En caso de ser correcto el registro, nos reenviará a la página inicial de la aplicación "index.php", con una nueva sesión abierta para el usuario registrado. La forma de comprobación se hace en la base de datos usando un cifrado MD5 para ganar en seguridad (Ilustración 52):

Ilustración 52: Login

En caso de no insertar datos correctos, se retornará al usuario a la página de login "login.php", donde aparecerá el mensaje "*Error en los datos de acceso: Usuario no existente en la base de datos o Password incorrecto*", como se muestra en la imagen de abajo (Ilustración 53).

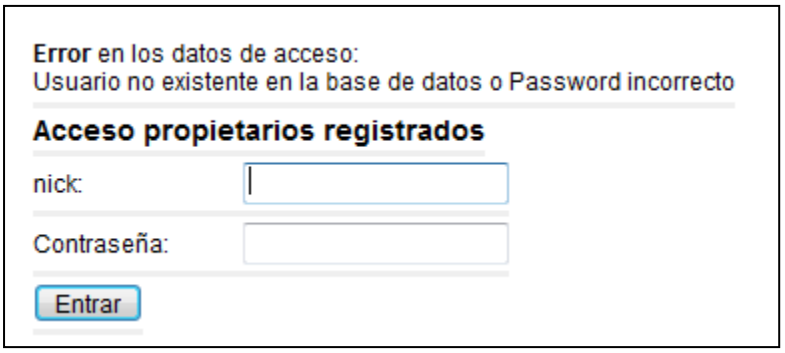

Ilustración 53: Error login

Existe otra posibilidad de error, y es que el usuario no inserte su nick o contraseña, en ese caso, la página se recargará con el siguiente mensaje de error: "*Debe especificar un usuario y password*", como vemos 3en la imagen de a continuación (Ilustración 54)

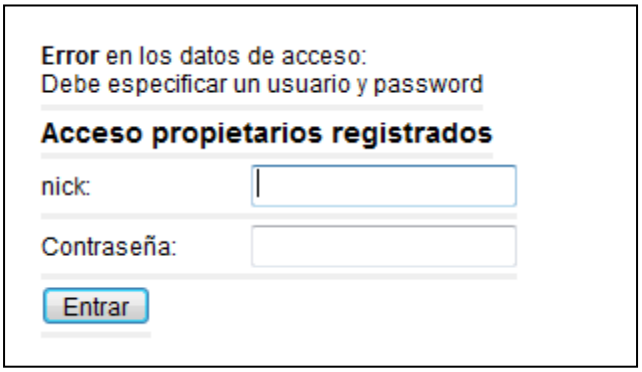

Ilustración 54: Error2 login

7- Conclusiones

7.1- Trabajo realizado

La aplicación InmoFamily responde a las expectativas y requerimientos recogidos en la especificación de requisitos. El trabajo realizado ofrece un alto nivel de cumplimiento de objetivo y es un punto de partida para que propietarios que pagan por la gestión de sus inmuebles, puedan desarrollar esta actividad vía internet con una aplicación gratuita.

InmoFamily ofrece al usuario una interfaz intuitiva de fácil manejo y que tiene aspecto cuidado, atractivo y sencillo. La sencillez del proceso de gestión y la rapidez del proceso gestión hacen de InmoFamily un gestor inmobiliario ideal para el usuario inexperto.

La incorporación de un módulo para gestionar las desgravaciones de un inmueble supone un punto más a favor para esta aplicación. Gracias a ello los usuarios propietarios podrán facilitar la visualización y gestión de sus desgravaciones en inmuebles para incorporarlo a la declaración de la renta, y beneficiarse de una manera más cómoda.

La aplicación pretende atraer usuarios gracias a la sencillez y usabilidad del sistema, ya que no existe nada similar en el mercado online de manera gratuita. Añadir también que no precisa de ninguna instalación y está disponible vía web.

7.2- Valoración personal

Durante todo el tiempo de creación de este proyecto fin de carrera he podido ver desde cero, cuál es el proceso para la creación de una aplicación web. Empezando por recabar la especificación de requisitos, planificación y estudio del proyecto, elección de tecnologías a utilizar, fase de implementación y hasta la fase de pruebas para que el usuario no tenga problema en su uso.

Al realizar el proyecto fin de carrera he puesto a prueba todos los conocimientos que he ido adquiriendo a lo largo de mis estudios. “Análisis e Ingeniería del Software”, “Diseño y Mantenimiento del Software I” y “Diseño y Mantenimiento del Software II”, me han ayudado en la planificación a la hora de esquematizar y simplificar (divide y vencerás) el proyecto en partes más pequeñas, así como en el desarrollo de diagramas UML. “Fundamentos de Programación” y “Metodología de la programación” me han ayudado en temas de programación estructurada, en mayor medida la primera debido a la similitud del lenguaje *C* al lenguaje *php*. Finalmente también añadir asignaturas relacionadas con la persistencia de información como “Sistemas de Gestión de Bases de Datos”, “Administración de Bases de Datos”, “Bases de Datos Avanzadas” y “Diseño de Bases de Datos”.

Ha sido un trabajo costoso, y que me ha llevado mucho tiempo. No he tenido tiempo de terminarlo como hubiese querido, puesto que considero que se pueden hacer mejoras y ampliaciones en los servicios que la web puede aportar a los usuarios. Aun así considero que se han alcanzado los requisitos y cumplido los objetivos con éxito.

Gracias a este proyecto final de carrera he obtenido un mayor conocimiento en las tecnologías utilizadas, tanto en html, sql, php, javascript, css como en Ajax, muy de moda en las páginas web actuales.

Para concluir decir que estoy satisfecho del trabajo realizado y de todo lo aprendido en el desarrollo de la aplicación.

7.3- Futuras mejoras

La curva de aprendizaje derivada de la formación en las nuevas tecnologías que he necesitado para realizar el proyecto me ha obligado a concentrarme en la funcionalidad básica que debía incorporar la aplicación dejando de lado posibles mejoras en algunas partes de la solución.

A continuación se expone un listado de futuras ampliaciones que podrían hacerse en:

- Mejorar los formularios de edición de la aplicación.
- Inserción de una funcionalidad de Foro de debate.
- Revisión completa interfaz gráfica.
- Crear un sistema de banners publicitarios.
- Crear una versión de la aplicación para resoluciones menores a 1024 x 768 pixeles.
- Incluir un modulo para el control del estado de pagos e ingresos vía SMS o MMS.
- Diseño del módulo de alertas para situar en un mismo punto de vista toda la información crítica para los propietarios.
- Generador de archivos pdf para impresión de contratos.
- Crear un gestor de contenidos para guardar copias de los contratos creados.
- Facilitar la comunicación entre usuarios inquilinos y usuarios propietarios con un gestor de mensajes.
- Crear un perfil útil para usuarios inquilinos (actualmente existe pero no tiene incorporada ningún servicio), para ver la evolución de su contrato, como facturas pagadas y facturas pendientes.
- Agregar más funcionalidades al módulo de desgravaciones para facilitar la visualización de datos.
- Incorporar un sistema de filtrado de datos para la visualización de todos los datos.

8- Bibliografía

- Desarrollo Web con PHP y MySQL. Glass, Michael - Madrid: Anaya Multimedia, D.L. 2004.
- Una guía para la realización de supervisión de proyectos final de carrera (PFC) en el ámbito de la web (Félix Buendía García)
- Guía del IEEE para la Especificación de Requerimientos Software. Departamento de Sistemas Informáticos y Computación Universidad Politécnica de Valencia.
- IEEE Std 830 - IEEE Guide to Software Requeriments Specifications. IEEE Standards Board. 345 Eas 47 th Street. New York, NY 10017, USA. 1984.
- Página oficial Apache. http://www.apache.org/
- Página oficial MySQL. http://www.mysql.com/
- Página oficial PHP. http://www.php.net/
- Página oficial Wamp. http://www.wampserver.com/en/index.php
- Wikipedia. http://es.wikipedia.org/

Anexos

A. Anexo I (Herramientas usadas)

Notepad++

Es un editor de texto y de código fuente libre con soporte para varios lenguajes de programación para Microsoft Windows.

Gracias a su velocidad, puede convertirse en una alternativa al bloc de notas. Con la implementación de navegación por pestañas, moverse entre los archivos de texto abiertos es más cómodo.Aunque Scintilla no permite la búsqueda y reemplazo de expresiones regulares múltiples, Notepad++ permite el uso de complementos que ayudan a mitigar este hecho.

Se distribuye bajo los términos de la Licencia Pública General de GNU.

Características:

- Coloreado y envoltura de sintaxis.
- Autocompletado.
- Multidocumento (pestañas).
- Multivista.
- Soporte para buscar/reemplazar. Permite el uso de expresiones regulares.
- Soporte completo para "arrastrar y colocar".
- Posición dinámica de las vistas.
- Detención automática del estado del documento.
- Herramienta de zoom.
- Funcionamiento bajo entornos plurilingües.
- Puntos de marca.
- Resaltado de paréntesis e indentación.
- Grabación y reproducción de macros.
- Soporte de extensiones (incluye algunas por defecto).
- Multilenguaje

Lenguajes soportados

Ada, ASP, ASM Ensamblador, , AutoIt, Batch, C, C#, C++, Caml, CMake, COBOL, CSS, D, Diff, Flash ActionScript, Fortran, Gui4Cli, Haskell, HTML, INNO, Java, JavaScript, JSP, KiXtart, Lisp, Lua, Makefile, MATLAB, MS INI (archivo), NSIS, Objective-C, Pascal, Perl, PHP, PostScript, PowerShell, Properties, Python, R, RC (fichero de recurso), Ruby, Shell, Scheme, Smalltalk, SQL, Tcl, TeX, VB, VHDL, Verilog, XML, YAML

Además, permite al usuario definir su propio lenguaje: no sólo las palabras clave para la sintaxis coloreada, sino también las palabras clave para la envoltura de sintaxis, los comentarios clave y los operadores. A pesar de la infinita lista de lenguajes soportados la lista sigue creciendo.

Desarrollo

Está basado en el componente de edición Scintilla y está escrito en C++ utilizando directamente la API de Windows y STL, lo que asegura una velocidad mayor de ejecución y un tamaño más reducido del programa final.

Uso en GNU/Linux

Puede utilizarse en GNU/Linux mediante Wine, y recientemente se ha añadido drag and drop (arrastrar y soltar) de archivos para este sistema operativo libre (Ilustración 55).

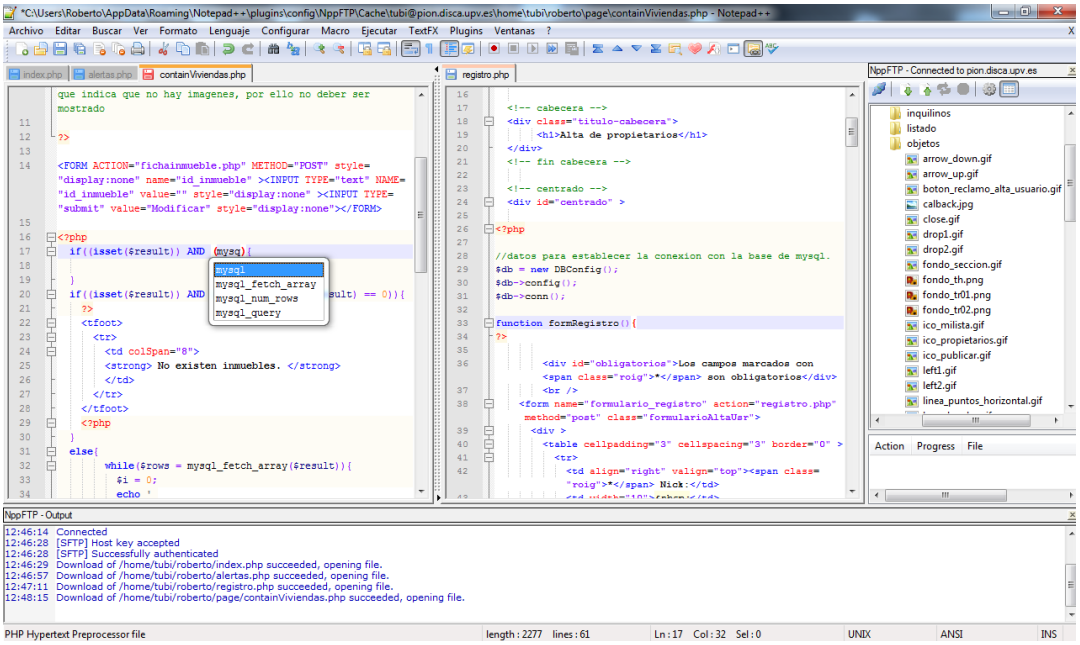

Ilustración 55: Notepad++

Adobe photoshop

En lo referente a la edición de imágenes y creación de logos del centro de estética he usado el programa Adobe Photoshop CS, que es uno de los paquetes software más potentes y conocidos para la edición y retoque de imágenes. Con este programa hemos creado la gran mayoría de imágenes que aparecen en la aplicación.

WAMP

WAMP es el acrónimo usado para describir un sistema de infraestructura de internet que usa las siguientes herramientas:

- Windows, como sistema operativo.
- Apache, como servidor web.
- MySQL, como gestor de bases de datos.
- PHP (generalmente), Perl, o Python, como lenguajes de programación.

El uso de un WAMP permite servir páginas HTML a internet, además de poder gestionar datos en ellas, al mismo tiempo un WAMP, proporciona lenguajes de programación para desarrollar aplicaciones web.

LAMP es el sistema análogo que corre bajo ambiente Linux.
WAMP es el sistema análogo que corre bajo ambiente Windows.
MAMP es el sistema análogo que corre bajo ambiente Macintosh.

Apache

El servidor HTTP Apache es un servidor web HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 [1] y la noción de sitio virtual. Cuando comenzó su desarrollo en 1995 se basó inicialmente en código del popular NCSA HTTPd 1.3, pero más tarde fue reescrito por completo. Su nombre se debe a que Behelendorf quería que tuviese la connotación de algo que es firme y enérgico pero no agresivo, y la tribu Apache fue la última en rendirse al que pronto se convertiría en gobierno de EEUU, y en esos momentos la preocupación de su grupo era que llegasen las empresas y "civilizasen" el paisaje que habían creado los primeros ingenieros de internet. Además Apache consistía solamente en un conjunto de parches a aplicar al servidor de NCSA. Era, en inglés, a patchy server (un servidor "parcheado").

El servidor Apache se desarrolla dentro del proyecto HTTP Server (httpd) de la Apache Software Foundation.

Apache presenta entre otras características altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración.

Apache tiene amplia aceptación en la red: desde 1996, Apache, es el servidor HTTP más usado. Alcanzó su máxima cuota de mercado en 2005 siendo el servidor empleado en el 70% de los sitios web en el mundo, sin embargo ha sufrido un descenso en su cuota de mercado en los últimos años. (Estadísticas históricas y de uso diario proporcionadas por Netcraft [2]).

La mayoría de las vulnerabilidades de la seguridad descubiertas y resueltas tan sólo pueden ser aprovechadas por usuarios locales y no remotamente. Sin embargo, algunas se pueden accionar remotamente en ciertas situaciones, o explotar por los usuarios locales malévolos en las disposiciones de recibimiento compartidas que utilizan PHP como módulo de Apache.

Ventajas

- Modular
- Código abierto
- Multi-plataforma
- Extensible
- Popular (fácil conseguir ayuda/soporte)

Apache es usado principalmente para enviar páginas web estáticas y dinámicas en la World Wide Web. Muchas aplicaciones web están diseñadas asumiendo como ambiente de implantación a Apache, o que utilizarán características propias de este servidor web.

Apache es usado para muchas otras tareas donde el contenido necesita ser puesto a disposición en una forma segura y confiable. Un ejemplo es al momento de compartir archivos desde una computadora personal hacia Internet. Un usuario que tiene Apache instalado en su escritorio puede colocar arbitrariamente archivos en la raíz de documentos de Apache, desde donde pueden ser compartidos.

Los programadores de aplicaciones web a veces utilizan una versión local de Apache con el fin de pre visualizar y probar código mientras éste es desarrollado.

Dia

Dia es una aplicación informática de propósito general para la creación de diagramas, desarrollada como parte del proyecto GNOME. Está concebido de forma modular, con diferentes paquetes de formas para diferentes necesidades. Tiene una interfaz muy atractiva como se ve en la Ilustración 56.

Dia está diseñado como un sustituto de la aplicación comercial Visio de Microsoft. Se puede utilizar para dibujar diferentes tipos de diagramas. Actualmente se incluyen diagramas entidad-relación, diagramas UML, diagramas de flujo, diagramas de redes, diagramas de circuitos eléctricos, etc. Nuevas formas pueden ser fácilmente agregadas, dibujándolas con un subconjunto de SVG e incluyéndolas en un archivo XML.

El formato para leer y almacenar gráficos es XML (comprimido con gzip, para ahorrar espacio). Puede producir salida en los formatos EPS, SVG y PNG.

También conviene recordar que Dia, gracias al paquete dia2code, puede generar el esqueleto del código a escribir, si utilizáramos con tal fin un UML.

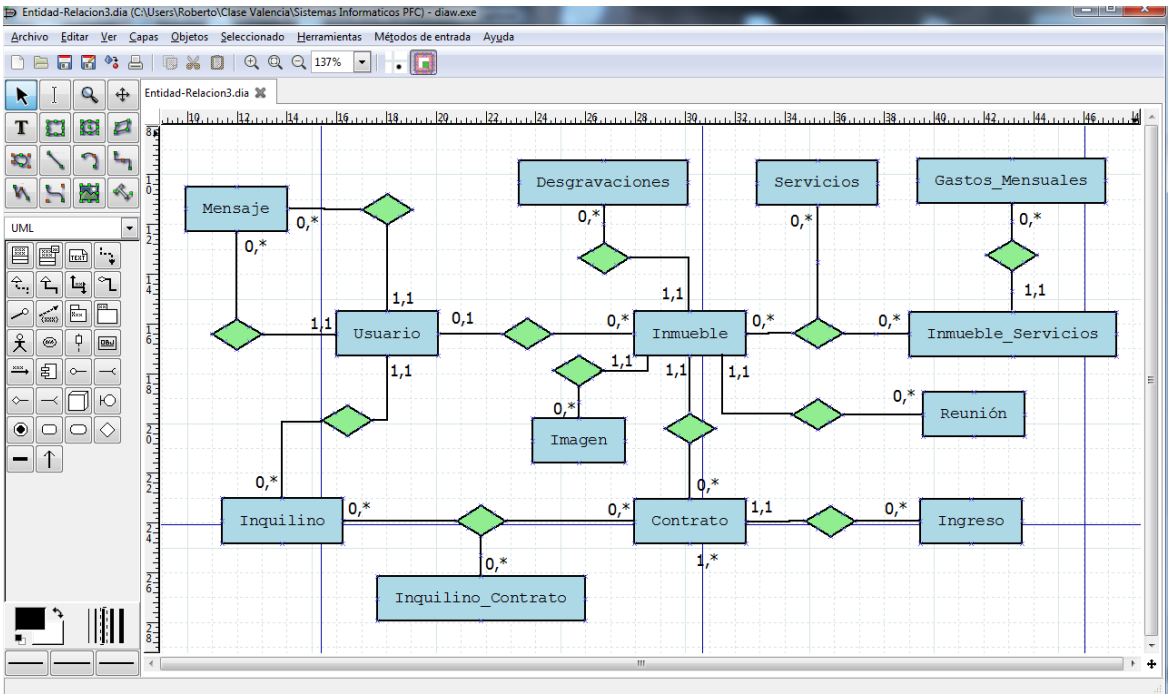

Ilustración 56: Dia UML