PROYECTO: GESTIÓN DE INCIDENCIAS EN UN EDIFICIO DE APARTAMENTOS

Mario Francisco Tojar Trujillo
Consultor: Oscar Escudero Sánchez
Ingeniería Técnica en Informática de Sistemas

INDICE

INDICE 2
TABLA DE IMÁGENES 4
1.- INTRODUCCION 5
1.1.- JUSTIFICACION DEL PROYECTO 5
1.2.- OBJETIVOS DEL TRABAJO 5
1.3.- ENFOQUE METODOLÓGICO 5
1.4.- PLANIFICACION DEL PROYECTO 6
2.- ESPECIFICACIÓN DE REQUISITOS 7
2.1.- INFORMACIÓN INICIAL 7
2.3.- CASOS DE USO 8
Caso 1: “Entrada al Sistema” 8
Caso 2: “Salida del Sistema” 9
Caso 3: “Añadir Apartamento” 9
Caso 4: “Modificar Apartamento” 10
Caso 5: “Añadir Profesional” 10
Caso 6: “Modificar Profesional” 11
Caso 7: “Borra Profesional” 11
Caso 8: “Añadir Tipo Desperfecto” 12
Caso 9: “Enlazar Profesional con Tipo de Desperfecto” 12
Caso 10: “Borrar enlace de Profesional con Tipo de Desperfecto” 13
Caso 11: “Añadir Cliente” 13
Caso 12: “Modificar Cliente” 14
Caso 13: “Administrador enlaza el cliente con el apartamento” 14
Caso 14: “Administrador borra el enlace del cliente con el apartamento” 15
Caso 15: “Recorrer partes sin visitar” 15
Caso 16: “Cliente crea un incidencia” 16
Caso 19: “Profesional rellena parte de incidencia” 16
Caso 20: “Sistema crea el parte de incidencias” 17
Caso 21: “Sistema envía los emails” 17
2.4.- Requisitos de la Interfaz de Usuario 18
3.- ANÁLISIS 19
3.1.- DIAGRAMA DE ESTADOS 19
3.2.- DIAGRAMAS DE CLASES 19
3.2.- DIAGRAMAS DE SECUENCIA 21
3.2.1.- CREACIÓN DE PARTE 21
3.2.2.- ACTUALIZACIÓN DE PARTE 22
3.2.3.- ASOCIACIÓN CLIENTE-APARTAMENTO 22
3.3.- MODELO ENTIDAD RELACIÓN 23
3.3.1.- DISEÑO DE TABLAS 23
4.- DISEÑO TÉCNICO 25
4.1.- PATRONES UTILIZADOS 25
4.1.1.- PATRON MVC (Modelo Vista-Controlador) 25
4.1.1.1- MARCO DE TRABAJO STRUTS (CONTROLADOR) 26
4.1.1.2.- PATRÓN COMPOSITE VIEW (VISTA) 27
4.1.2.- PATRON BUSSINES DELEGATE 27
4.1.3.- CONTROL DE SESSION 28
4.1.4.- PATRON VALUE OBJECT 28
4.1.5.- PATRÓN DAO (Data Access Object) 29
5.- IMPLEMENTACIÓN 31

5.1.- FUNCIONAMIENTO 31
5.1.1.- CAPA WEB 32
5.1.2.- CAPA NEGOCIO 34
5.1.3.- MANEJO DE EXCEPCIONES 34
5.2.- HERRAMIENTAS DE DESARROLLO 34
5.3.- PROCESO DE INSTALACION 35
5.3.2.- CONFIGURACIÓN 36
SERVIDOR DE BASE DE DATOS 36
APLICACIÓN 36
5.4.- JUEGOS DE PRUEBA 37
6.- CONCLUSIONES 40
Anexo I 42
CÓMO INSTALAR Y CONFIGURAR MYSQL SERVER EN WINDOWS 42
Anexo II 48
CÓMO INSTALAR Y CONFIGURAR jdk-1_5_0_15 EN WINDOWS 48
Anexo III 50
CÓMO INSTALAR Y CONFIGURAR TOMCAT 6.0.16 EN WINDOWS 50

TABLA DE IMÁGENES

Ilustración 1 *Diagrama De Gantt* 6
Ilustración 2 *Esquema de páginas de GIEDA* 18
Ilustración 3 *Diagrama De Estados* 19
Ilustración 4 *Diagrama de Clases* 20
Ilustración 5 *Creación de Parte* 21
Ilustración 6 *Rellena Parte* 22
Ilustración 7 *Asociación Cliente-Apartamento* 22
Ilustración 8 *Modelo Entidad-Relación* 23
Ilustración 9 *Patrón MVC* 25
Ilustración 10 *Funcionamiento De Struts* 26
Ilustración 11 *Patrón Bussines Delegate* 27
Ilustración 12 *Patrón Value Object* 28
Ilustración 13 *Patrón DAO (Data Access Object)* 30
Ilustración 14 *Implementación* 32
Ilustración 15 *Funcionamiento del Usuario Administrador visto desde Struts* .. 33
Ilustración 16 *Estructura de directorios de Tomcat* 36

1.- INTRODUCCIÓN

1.1.- JUSTIFICACIÓN DEL PROYECTO

La idea surgió en unas vacaciones. El apartamento que alquilamos sufrió algunos desperfectos, así como algunas de las instalaciones del edificio donde estábamos hospedados.

El administrador del edificio solo pasaba cada semana y la única manera de ponernos en contacto con él, era a través de un teléfono móvil que siempre comunicaba.

Nos pareció un buen tema para el TFC la manera de poder gestionar este escenario.

1.2.- OBJETIVOS DEL TRABAJO

El objetivo del TFC es automatizar la gestión de los desperfectos, para ello se observan varias fases o apartados:

- La creación de la incidencia por el cliente, que se verá reflejado en un "parte de incidencias".
- La comunicación entre los distintos actores que se realizará mediante el correo electrónico:
  - El administrador del edificio
  - El cliente que genera el desperfecto
  - El profesional que debe reparar el desperfecto
- La gestión del parte de incidencias por el profesional adecuado.

Para conseguir el objetivo anterior, debemos tener conocimiento de los clientes, los profesionales disponibles y que desperfecto repara cada uno, la ocupación de los apartamentos por los clientes y los que hay vacíos.

1.3.- ENFOQUE METODOLÓGICO

El TFC es sobre J2EE, por lo tanto se usará esta tecnología para realizar el desarrollo de la aplicación.

El desarrollo del proyecto se basa en el modelo MVC (Vista-Controlador) implementado por medio de Struts, junto con los patrones Business Delegate, Value Object y DAO para separar las diferentes capas de la aplicación (Web, Negocio, acceso al SGBD). Toda esta metodología y arquitectura se emplea para garantizar la independencia y modularidad de las diferentes partes de la aplicación.

Como servidor de aplicaciones usaremos Tomcat y MySql 5.0 como Sistema de Gestión de Base de Datos.

1.4.- PLANIFICACIÓN DEL PROYECTO

Especificación y Análisis

8 de marzo al 22 de marzo

- Descripción textual de la aplicación a desarrollar
- Análisis del software según las necesidades del cliente
- Requisitos no funcionales relacionados con el software y hardware
- Casos de uso

Diseño

23 de marzo al 14 de abril

- Diseño de la arquitectura de la aplicación según los requisitos de la aplicación
- Diseño de las estructuras de datos
- Prototipo de la interfaz gráfica

Implementación

15 de abril al 19 de mayo

- Implementación
- Pruebas

Redactado de la Memoria Final

19 de mayo al 25 de junio

- Finalización de la implementación
- Redactado final de la memoria
- Preparación de la presentación virtual

Diagrama De Gantt

	Nombre de tarea	Comienzo	Fin
1	**PLAN DE TRABAJO**	**28/02/08**	**09/03/08**
2	Localización del software	28/02/08	29/02/08
3	Desarrollo del plan de trabajo	01/03/08	06/03/08
4	Creacion Diagramade Gantt	07/03/08	09/03/08
5	Entrega PEC 1	14/03/08	14/03/08
6	**ESPECIFICACION Y ANALISIS**	**10/03/08**	**17/03/08**
7	Descripción textual de la	10/03/08	11/03/08
8	Análisis del Softw	12/03/08	14/03/08
9	Requisitos no funcionales	15/03/08	17/03/08
10	Casos de Uso	10/03/08	17/03/08
11	**DISEÑO**	**18/03/08**	**13/04/08**
12	Diseño de la arquitectura	18/03/08	13/04/08
13	Diseño de la Estructura de	26/03/08	07/04/08
14	Diseño Interfaz Grafica	09/04/08	13/04/08
15	Entrega PEC2	14/04/08	14/04/08
16	**IMPLEMENTACION**	**16/04/08**	**18/05/08**
17	Implementacion	16/04/08	18/05/08
18	Pruebas	21/04/08	18/05/08
19	Entrega PEC3	19/05/08	19/05/08
20	**MEMORIA FINAL**	**20/05/08**	**24/06/08**
21	Implementación II	20/05/08	01/06/08
22	Redactado Memoria Final	02/06/08	11/06/08
23	Presentación Virtu	17/06/08	24/06/08
24	Entrega Memoria Final	25/06/08	25/06/08

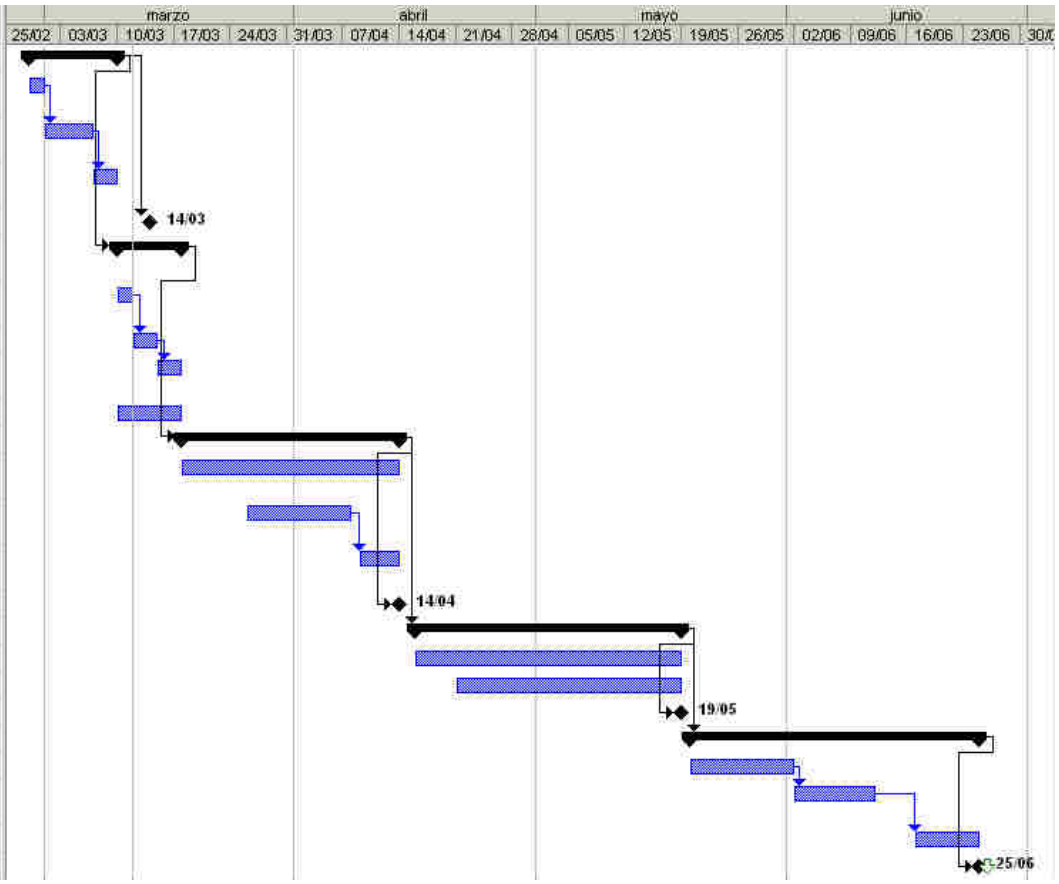


Ilustración 1 ***Diagrama De Gantt***

2.- ESPECIFICACIÓN DE REQUISITOS

2.1.- INFORMACIÓN INICIAL

La idea principal de la aplicación es la de mantener un sistema de comunicación de desperfectos lo más automatizado posible. La descripción de la funcionalidad es la siguiente:

El administrador de la empresa es el encargado de mantener y crear los apartamentos, profesionales, desperfectos, clientes, así como de las relaciones entre ellos; emparejar los profesionales con los desperfectos que reparen y los clientes con los apartamentos que ocupen.

Otra facultad del administrador es la de comprobar que los partes son reparados, pudiendo mostrar un listado con los partes que no han sido revisados con más de 3 días de antigüedad.

Los desperfectos que existen inicialmente son (entre paréntesis están sus códigos):

- TV (tv)
- Electrodomésticos (em)
- Fontanería (ft)
- Electricidad (ed)
- Cerrajería (cj)
- Mobiliario (mb)
- Ascensores (ac)
- Cristalería (ct)
- Carpintería (cp)
- Tapicería (tp)
- Otros (ot)

Los clientes al hacer uso de los apartamentos y zonas comunes, son los encargados de crear los partes, cuando noten que existe algún desperfecto.

Al crear el parte, el sistema pondrá en marcha el envío de los 3 emails; uno al profesional, otro al administrador y otro al cliente, como confirmación de que se ha dado aviso.

El profesional cuando realice la visita o visitas y arregle el desperfecto, terminará de confeccionar el parte con las particularidades del desperfecto, el arreglo y cualquier comentario que considere oportuno.

Cada parte solo puede ser visitado tres veces, si a la tercera vez no se ha podido efectuar su reparación, el sistema enviará un email al administrador para que realice las operaciones que considere oportuno.

Cuando el cliente se marche el administrador eliminará la asociación del cliente con el apartamento, dejando esté último libre para que pueda venir otro cliente.

2.2.- GLOSARIO

Administrador: Persona encargada del edificio. Se encarga de mantener los clientes, los apartamentos, los profesionales y la asociación cliente-apartamento, así como de comunicarse con los profesionales en el caso de que se necesite alguna cosa.

Profesional: Persona encargada de solucionar el desperfecto. Puede variar en función del tipo de desperfecto. Se encarga de rellenar los partes de trabajo.

Cliente: Persona que alquila el apartamento. Creará el desperfecto en caso de que los haya.

Apartamento: Objeto sobre el cual se producen desperfectos.

Parte de incidencias: Elemento clave que relaciona el desperfecto, el profesional y la ubicación del desperfecto, ya sea un apartamento o una zona común.

Incidencia: Cualquier tipo de avería o anomalía en el funcionamiento normal ocurrida en un apartamento o en una zona común (escaleras, ascensor) que sea susceptible de reparar por el dueño de los apartamentos.

Desperfecto: Tipo específico de la incidencia.

MVC: Patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

Business Delegate: Patrón usado para separar la capa de presentación de los servicios de negocio, como detalles del SGBD.

Value Object: Son simples contenedores de datos estructurales, encapsulamos los datos de negocio para independizarlos del SGBD usado.

2.3.- CASOS DE USO

Caso 1: “Entrada al Sistema”

Caso de Uso: EntrarAlSistema
ID: T 1
Breve Descripción: El usuario introduce el login y contraseña, el sistema valida los datos y le deja entrar o no
Actores principales: Administrador, Cliente, Profesional
Actores Secundarios: Ninguno
Precondiciones: 1. Ninguna
Flujo principal: 1. El sistema muestra la pantalla de entrada al sistema 2. El usuario introduce su login y el password 3. Si el usuario existe. 3.1. Se direcciona a su página de entrada (menú personalizado) 3.2. Se carga en la sesión los datos del usuario, el tipo de usuario que es, su email y su nif. 4. Si el cliente no existe 4.1. Vuelve a mostrar la página de entrada. 4.2. Se escribe una entrada en el log de suceso
Postcondiciones: 1. El usuario se introduce en el sistema
Flujos Alternativos:

Caso 2: “Salida del Sistema”

Caso de Uso: SalirDelSistema
ID: T 2
Breve Descripción: El usuario selecciona esta opción
Actores principales: Cliente, Administrador, Profesional
Actores Secundarios: Ninguno
Precondiciones: 1. El usuario elija esta opción
Flujo principal: 1. El usuario selecciona esta opción. 2. Se vacía la sesión y se cierran todas las conexiones que tuviese abiertas. 3. Se muestra el mensaje de despedida
Postcondiciones: 2. El usuario queda fuera del sistema
Flujos Alternativos: 1. Ninguno.

Caso 3: “Añadir Apartamento”

Caso de Uso : AñadirApartamento
ID: A 1
Breve Descripción: Añade un apartamento al sistema
Actores principales: Administrador
Actores Secundarios: Ninguno
Precondiciones: 1. No exista el apartamento
Flujo principal: 1. El Administrador selecciona esta opción 2. El sistema muestra una página con los datos a rellenar del apartamento 3. El Administrador rellena los datos. 4. El Administrador acepta la operación 5. Si los datos son correctos 5.1. Si el apartamento no existe 5.1.1. Se crea el registro en la tabla APARTAMENTOS 5.2. Si el apartamento existe 5.2.1. Se muestra mensaje de error 6. Si los datos no son correctos 6.1. El sistema muestra un mensaje de error
Postcondiciones: 1. El apartamento queda insertado en tabla
Flujos Alternativos: 1. Error al realizar la operación en la base de datos.

Caso 4: “Modificar Apartamento”

Caso de Uso : ModificaApartamento
ID: A 2
Breve Descripción: Modifica un apartamento al sistema
Actores principales: Administrador
Actores Secundarios: Ninguno
Precondiciones: 1. Exista el apartamento
Flujo principal: 1. El Administrador selecciona esta opción. 2. El sistema muestra una página con los apartamentos disponibles. 3. El Administrador selecciona un apartamento. 4. El sistema muestra los datos del apartamento. 5. El administrador modifica los datos que necesite 6. El administrador acepta los datos 7. El sistema actualiza los datos del apartamento
Postcondiciones: 2. El apartamento queda modificado en la tabla
Flujos Alternativos: 2. Error al realizar la operación en la base de datos.

Caso 5: “Añadir Profesional”

Caso de Uso : AñadirProfesional
ID: A 3
Breve Descripción: Añade un profesional al sistema.
Actores principales: Administrador
Actores Secundarios: Ninguno
Precondiciones: 1. No exista el profesional
Flujo principal: 1. El Administrador selecciona esta opción 2. El sistema muestra una página con los datos a rellenar del profesional 3. El Administrador rellena los datos. 4. El Administrador acepta la operación 5. Si los datos son correctos 5.1. Si el profesional no existe 5.1.1. Se crea el registro en la base de datos 5.2. Si el profesional existe 5.2.1. Se muestra mensaje de error 6. Si los datos no son correctos 6.1. El sistema muestra un mensaje de error
Postcondiciones: 3. El profesional queda insertado en tabla
Flujos Alternativos: 3. Error al realizar la operación en la base de datos.

Caso 6: “Modificar Profesional”

Caso de Uso : ModificaProfesional
ID: A 4
Breve Descripción: Modifica un profesional al sistema.
Actores principales: Administrador
Actores Secundarios: Ninguno
Precondiciones: 1. Exista el profesional
Flujo principal: 1. El Administrador selecciona esta opción. 2. El sistema muestra una página con los profesionales disponibles. 3. El Administrador selecciona un profesional. 4. El sistema muestra los datos del profesional. 5. El Administrador modifica los datos necesarios. 6. El Administrador acepta la operación 7. El sistema guarda los datos modificados
Postcondiciones: 4. El profesional queda modificado en tabla
Flujos Alternativos: 4. Error al realizar la operación en la base de datos.

Caso 7: “Borra Profesional”

Caso de Uso : BorraProfesional
ID: A 5
Breve Descripción: Borra un profesional al sistema.
Actores principales: Administrador
Actores Secundarios: Ninguno
Precondiciones: 1. Exista el profesional. 2. El profesional no esté asociado a ningún desperfecto
Flujo principal: 1. El Administrador selecciona esta opción. 2. El sistema muestra una página con los profesionales disponibles. 3. El Administrador selecciona un profesional. 4. El Administrador acepta la operación. 5. Si el profesional no tiene partes por arreglar 5.1. Si el profesional no tiene asignados desperfectos 5.1.1. El sistema borra el profesional 5.2. Si no 5.2.1. El sistema muestra un mensaje de error 6. Si no 6.1. El sistema muestra un mensaje de error
Postcondiciones: 5. El profesional queda borrado de la tabla
Flujos Alternativos: 5. Error al realizar la operación en la base de datos.

Caso 8: "Añadir Tipo Desperfecto"

Caso de Uso : AñadirDesperfecto
ID: A 6
Breve Descripción: Añade un desperfecto al sistema.
Actores principales: Administrador
Actores Secundarios: Ninguno
Precondiciones: 1. No exista el desperfecto
Flujo principal: 1. El Administrador selecciona esta opción 2. El sistema muestra una página con los datos del desperfecto para rellenar 3. El Administrador rellena los datos. 4. El Administrador acepta la operación 5. Si los datos son correctos 5.1. Si el desperfecto no existe 5.1.1. Se crea el registro en la base de datos 5.2. Si el desperfecto existe 5.2.1. Se muestra mensaje de error 6. Si los datos no son correctos 6.1. Se muestra mensaje de error
Postcondiciones: 6. El desperfecto queda insertado en tabla
Flujos Alternativos: 6. Error al realizar la operación en la base de datos.

Caso 9: "Enlazar Profesional con Tipo de Desperfecto"

Caso de Uso : AsociaProfesionalDesperfecto
ID: A 7
Breve Descripción: Asigna a cada profesional los desperfectos que arregle
Actores principales: Administrador
Actores Secundarios: Ninguno
Precondiciones: 1. Exista el desperfecto 2. Exista el profesional
Flujo principal: 1. El Administrador selecciona esta opción 2. El sistema muestra en una página los dni y el nombre de los profesionales y los nombres de los desperfectos que no están asociados aun. 3. El Administrador selecciona un profesional y un desperfecto. 4. El Administrador acepta la operación. 5. El sistema añade un registro en la tabla adecuada
Postcondiciones: 1. La asignación queda almacenada en la tabla
Flujos Alternativos: 1. Error al insertar el registro en la base de datos.

Caso 10: “Borrar enlace de Profesional con Tipo de Desperfecto”

Caso de Uso : BorrarProfesionalDesperfecto
ID: A 8
Breve Descripción: Borra una asociación de un profesional con un tipo de desperfecto.
Actores principales: Administrador
Actores Secundarios: Ninguno
Precondiciones: 1. Exista el enlace 2. No haya partes por arreglar con esta asociación.
Flujo principal: 1. El Administrador selecciona esta opción 2. El sistema muestra en una página con los enlaces entre profesional y tipo de desperfecto existentes. 3. El Administrador selecciona uno de ellos. 4. El Administrador acepta la operación. 5. El sistema borra el registro de la tabla adecuada
Postcondiciones: 1. La asignación queda borrada de la tabla
Flujos Alternativos: 1. Error al insertar el registro en la base de datos.

Caso 11: “Añadir Cliente”

Caso de Uso : AñadirCliente
ID: A 9
Breve Descripción: Añade un cliente en la base de datos
Actores principales: Administrador
Actores Secundarios: Ninguno
Precondiciones: 1. No exista el cliente
Flujo principal: 1. El Administrador selecciona esta opción 2. El sistema muestra una página con los datos del cliente 3. El Administrador rellena los datos del cliente. 4. El Administrador acepta la operación. 5. Si el cliente no existe 5.1. Se añade el cliente a la base de datos 6. Si el cliente existe 6.1. Se muestra mensaje de error
Postcondiciones: 1. El cliente queda almacenado en la base de datos
Flujos Alternativos: 1. Error al añadir el cliente a la base de datos.

Caso 12: “Modificar Cliente”

Caso de Uso : ModificarCliente
ID: A 10
Breve Descripción: Modifica un cliente en la base de datos
Actores principales: Administrador
Actores Secundarios: Ninguno
Precondiciones: 1. Exista el cliente
Flujo principal: 1. El Administrador selecciona esta opción 2. El sistema muestra una página con el NIF y el nombre de los clientes que hay. 3. El Administrador selecciona el cliente. 4. El sistema muestra el nif y el nombre y los apellidos de todos los clientes. 5. El Administrador selecciona un cliente. 6. El sistema muestra los datos del cliente 7. El Administrador modifica los datos del cliente. 8. El Administrador acepta la operación. 9. El sistema guarda los datos modificados.
Postcondiciones: 2. El cliente modificado queda almacenado en la base de datos
Flujos Alternativos: 2. Error al añadir el cliente a la base de datos.

Caso 13: “Administrador enlaza el cliente con el apartamento”

Caso de Uso : AsociaClienteApartamento
ID: A 11
Breve Descripción: Relaciona un cliente con el apartamento que ocupa
Actores principales: Administrador
Actores Secundarios: Ninguno
Precondiciones: 1. Exista el cliente 2. Exista el apartamento
Flujo principal: 1. El Administrador selecciona esta opción 2. El sistema muestra el nombre de los clientes y los números de los apartamentos. 3. El Administrador selecciona un cliente y un apartamento. 4. El Administrador acepta la operación. 5. Si el apartamento está libre en esa fecha 5.1. El sistema añade un registro en la tabla relaciones. 6. Si el apartamento no está libre 6.1. Se muestra mensaje de error
Postcondiciones: 3. La relación queda almacenada en la base de datos
Flujos Alternativos: 3. Error al añadir la relación a la base de datos.

Caso 14: "Administrador borra el enlace del cliente con el apartamento"

Caso de Uso : BorraAsociacionClienteApartamento
ID: A 12
Breve Descripción: Elimina la relación entre apartamento y el cliente, cuando éste se marcha
Actores principales: Administrador
Actores Secundarios: Ninguno
Precondiciones: 1. Exista el cliente 2. Exista el apartamento
Flujo principal: 1. El Administrador selecciona esta opción 2. El sistema muestra en una página los dni de los clientes que están en el complejo, el número del apartamento que ocupa y la fecha de entrada. 3. El Administrador selecciona un cliente y su apartamento. 4. El Administrador acepta la operación. 5. Se elimina el registro de la tabla de relaciones 6. Se modifica el Apartamento colocándolo como disponible
Postcondiciones: 1. La relación queda eliminada de la base de datos
Flujos Alternativos: 1. Error al eliminar la relación a la base de datos.

Caso 15: "Recorrer partes sin visitar"

Caso de Uso : RecorrerPartesSinVisitar
ID: A 13
Breve Descripción: El sistema recorre los partes, comprobando los que llevan más de tres días sin ser revisados
Actores principales: Administrador
Actores Secundarios: Ninguno
Precondiciones: 1. Ninguna
Flujo principal: 1. El Administrador selecciona esta opción. 2. Para cada parte en la tabla. 2.1. Si el parte está arreglado 2.1.1. Se mueve el parte al histórico 2.2. Si el parte no está arreglado 2.2.1. Si lleva más de 3 días sin ser revisado 2.2.1.1. Envío de email al administrador, incluye(EnviarEmail) 2.2.1.2. Envió de email al profesional, incluye(EnviarEmail) 2.2.1.3. Añade al listado
Postcondiciones: 1. Los partes arreglados están en la tabla histórico 2. Se han enviado emails de aquellos partes no arreglados y que llevan más de una semana sin ser revisados 3. Se muestra un listado con los partes

Flujos Alternativos: 1. Error al enviar algún correo. 2. Error al borrar de la tabla 3. Error al insertar en la tabla

Caso 16: "Cliente crea un incidencia"

Caso de Uso : CrearIncidencia
ID: C.1
Breve Descripción: Cliente crea un tipo de incidencia.
Actores principales: Cliente
Actores Secundarios:
Precondiciones: 1. Cliente observe la incidencia.
Flujo principal: 1. Cliente entra en el sistema 2. Cliente selecciona esta opción 3. Cliente rellena los datos del parte 4. Cliente acepta la operación 5. incluye (CrearParte)
Postcondiciones: 1. El parte se ha creado en la tabla adecuada
Flujos Alternativos: 1. Error al añadir el parte en la tabla

Caso 17: "Profesional rellena parte de incidencia"

Caso de Uso : RellenarParte
ID: P.1
Breve Descripción: Profesional modifica el parte si lo arregla y con cada visita.
Actores principales: Profesional
Actores Secundarios: Sistema
Precondiciones: 1. Exista el parte. 2. Exista el profesional.
Flujo principal: 1. Profesional entra en el sistema. 2. Sistema muestra todos los partes pendientes del profesional. 3. Profesional selecciona el parte que ha venido a reparar 4. Profesional modifica el parte (arreglado si o no) 5. Profesional acepta la operación 6. Sistema modifica la fecha de visita, según sea la primera, segunda o tercera visita. 7. Si es la tercera visita y no está arreglado 7.1. Sistema envía email al Administrador 8. Sistema modifica el registro
Postcondiciones: 1. El parte ha sido modificado y en su caso arreglado
Flujos Alternativos: 2. Error al modificar el parte en la tabla

Caso 18: “Sistema crea el parte de incidencias”

Caso de Uso : CrearParte
ID: S 1
Breve Descripción: Creación de un parte de incidencias, por medio de un cliente
Actores principales: Cliente
Actores Secundarios:
Precondiciones: 1. Funcionalmente ninguna, excepto que el cliente encuentre un desperfecto
Flujo principal: 1. El caso de uso empieza cuando un cliente crea una incidencia 2. El sistema recoge los datos y crea el parte añadiendo el registro en la tabla. 3. Si existe profesional asignado a la incidencia 3.1. incluye¨(EnviarEmail) 4. Envíar email al administrador, incluye (EnviarEmail) 5. Envíar email al cliente, incluye (EnviarEmail)
Postcondiciones: 2. El parte ha sido creado e insertado en la tabla
Flujos Alternativos: 2. Error al añadir el parte en la base de datos.

Caso 19: “Sistema envía los emails”

Caso de Uso : EnviarEmail
ID: S 2
Breve Descripción: Sistema crea el email y lo envía al destinatario
Actores principales: Cliente
Actores Secundarios:
Precondiciones: 1. Emails enviados
Flujo principal: 1. Sistema coge la dirección electrónica con el nif que le han enviado. 2. Sistema recoge los datos necesarios para el envío del email 3. Sistema crea el email y lo envía.
Postcondiciones: 1. Los emails han sido enviados
Flujos Alternativos: 1. Error al enviar el correo.

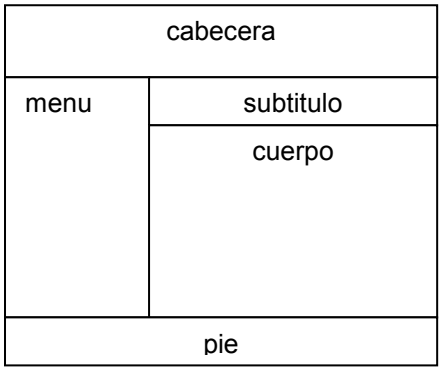
2.4.- Requisitos de la Interfaz de Usuario

La aplicación web permitirá al usuario conectarse al sistema, previa autenticación (con control usuario/clave).

El interfaz de usuario tiene un diseño homogéneo para todos los usuarios, con la salvedad del menú de opciones que tiene distintas opciones para cada usuario.

Dispondrá de un menú en la parte derecha con las posibilidades de elección según el usuario. Dependiendo de la elección en el centro de la pantalla se mostrará un formulario adecuado para la entrada de información.

Si se produce un error, esto genera un mensaje mostrado en pantalla y en un formato adecuado, en caso contrario, volverá a la página principal de la aplicación.



Formato de una página general

Ilustración 2 ***Esquema de páginas de GIEDA***

3.- ANÁLISIS

3.1.- DIAGRAMA DE ESTADOS

Antes de realizar el diagrama de clases, definiremos el modelo conceptual. En POO, este modelo conceptual representa los objetos en el mundo real y sus relaciones entre ellos.

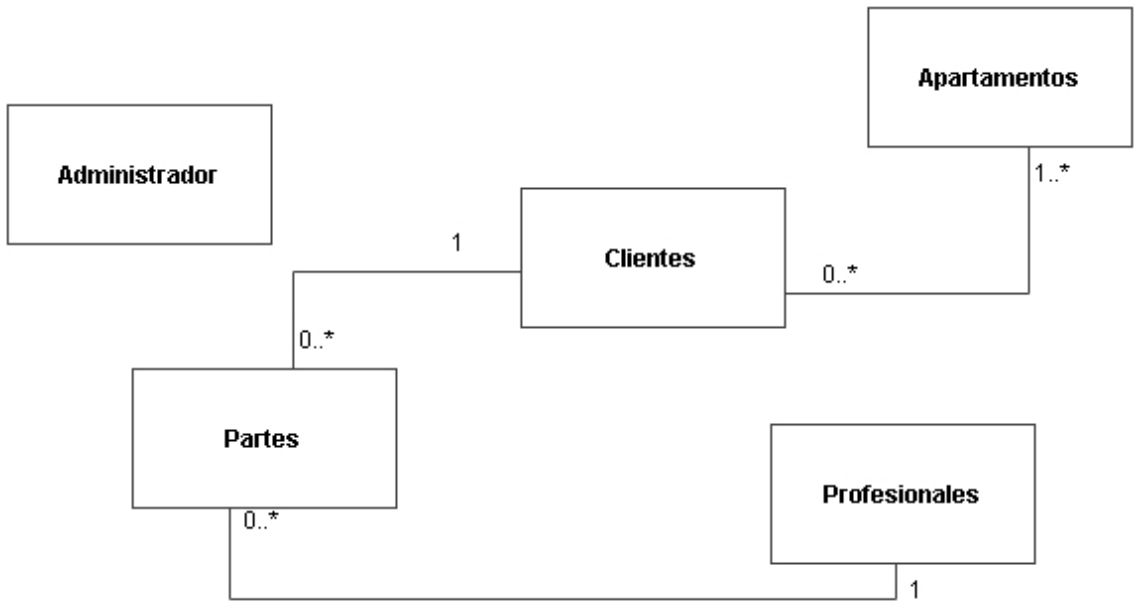


Ilustración 3 *Diagrama De Estados*

El objeto Administrador no tiene relación directa con ninguno de los otros objetos, pero es tan importante en el proceso de la aplicación, crea y gestiona los Clientes, Apartamentos y Profesionales, aunque después no interactúa directamente en el proceso de la gestión de los Partes.

3.2.- DIAGRAMAS DE CLASES

El siguiente diagrama estático nos muestra las principales clases del sistema, sus atributos y operaciones.

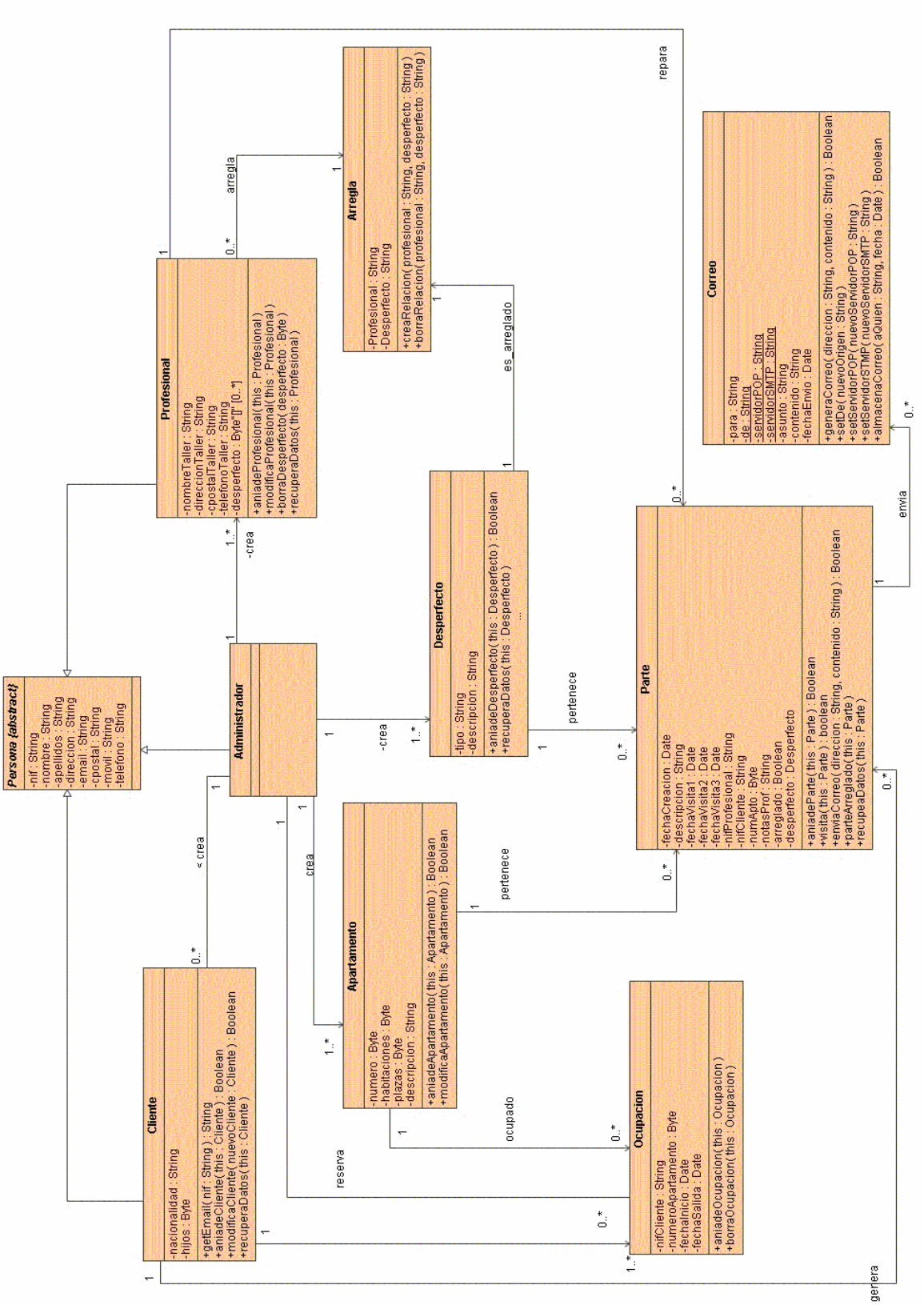


Ilustración 4 *Diagrama de Clases*

3.2.- DIAGRAMAS DE SECUENCIA

Vamos a mostrar los diagramas de secuencia más representativos, son las tres operaciones más complicadas:

- "Creación del Parte" por el cliente
- "Actualización del Parte" por el profesional
- "Asociación Cliente-Apartamento" por el administrador

3.2.1.- CREACIÓN DE PARTE

Creación de un parte por el cliente.

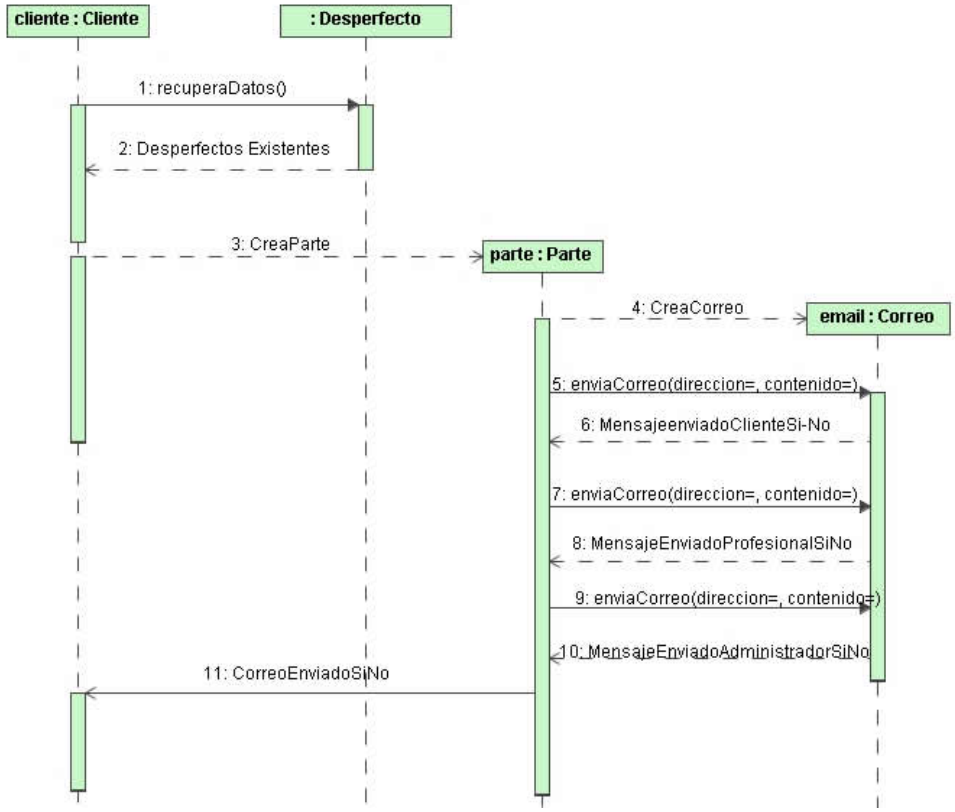


Ilustración 5 ***Creación de Parte***

3.2.2.- ACTUALIZACIÓN DE PARTE

Actualización de un parte por el profesional correspondiente.

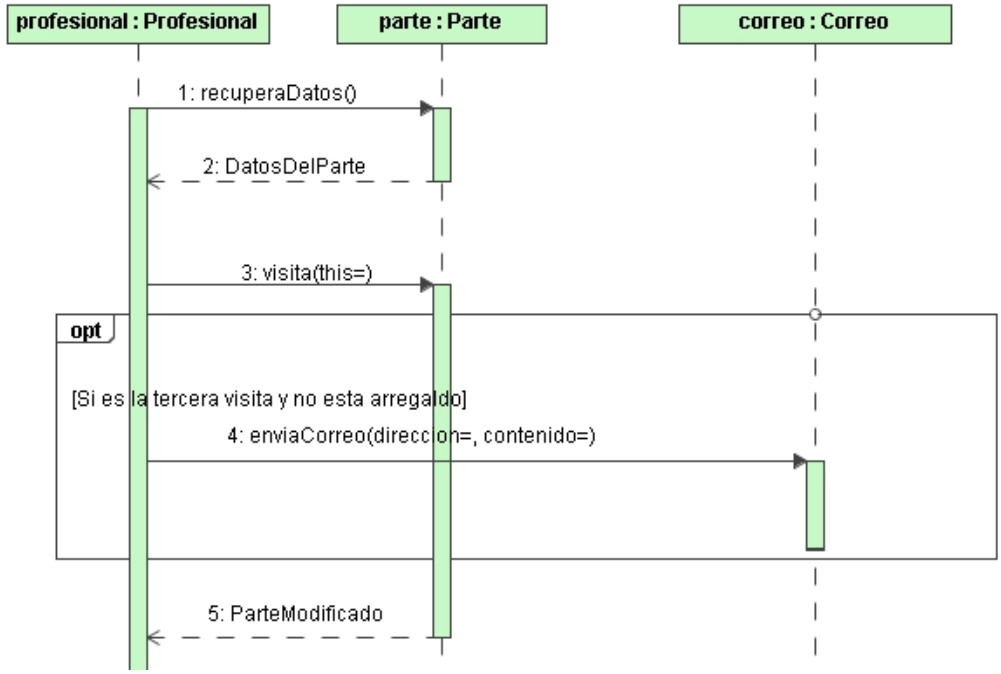


Ilustración 6 ***Rellena Parte***

3.2.3.- ASOCIACIÓN CLIENTE-APARTAMENTO

Asociación de un cliente al apartamento elegido por el administrador

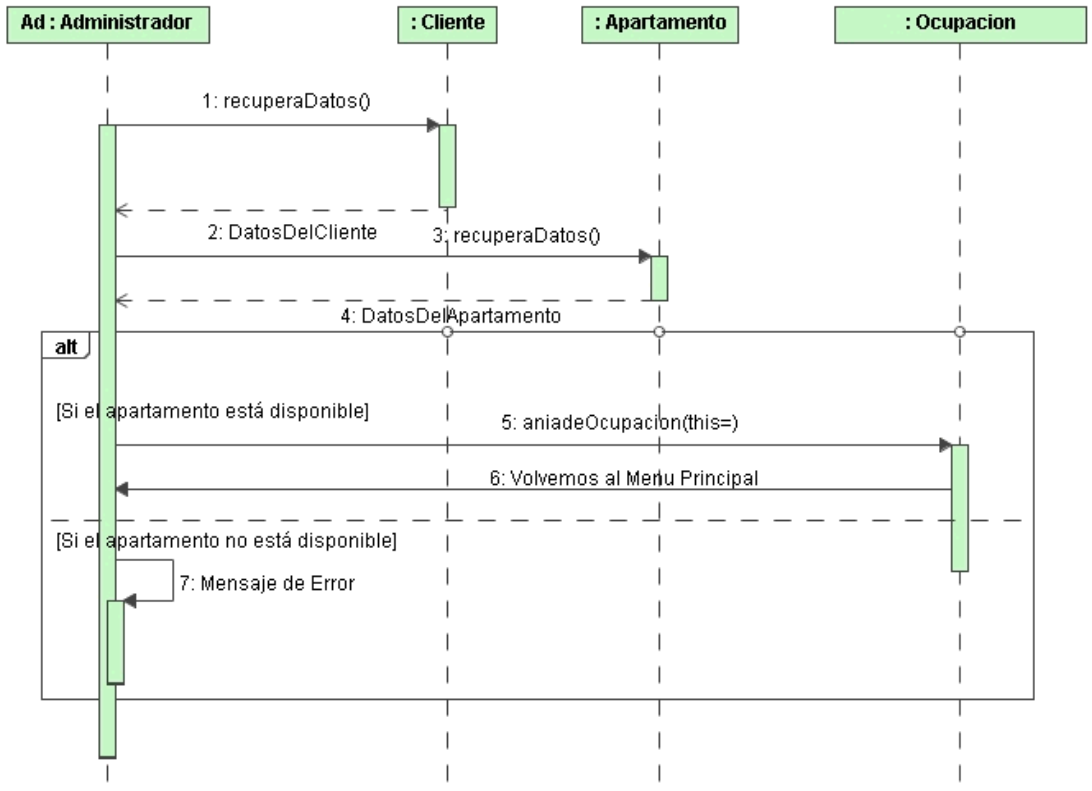


Ilustración 7 ***Asociación Cliente-Apartamento***

3.3.- MODELO ENTIDAD RELACIÓN

El siguiente esquema muestra las tablas y sus relaciones entre ellas.

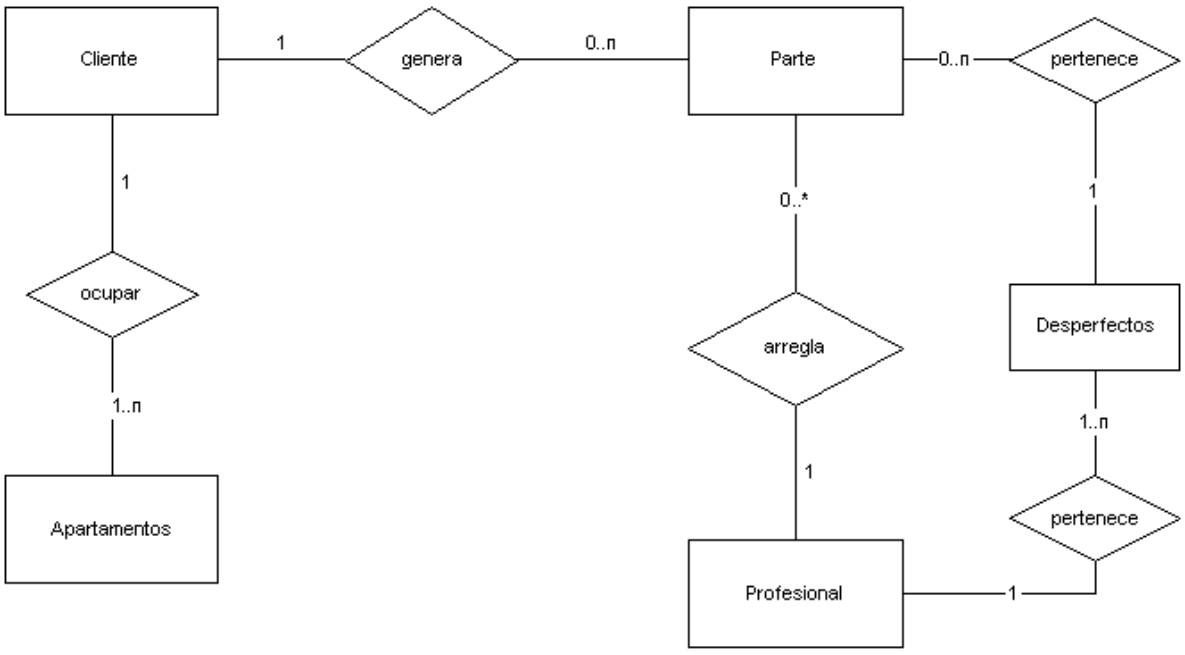


Ilustración 8 ***Modelo Entidad-Relación***

3.3.1.- DISEÑO DE TABLAS

Mostramos los campos y las claves de las tablas mostradas en el esquema anterior.

APARTAMENTO

- **NUMERO (clave principal)**
- HABITACIONES
- PLAZAS
- DESCRIPCION

CLAVES

- **EMAIL (clave)**
- **NIF (clave)**
- TIPO_USUARIO

} **Claves Principales**

CLIENTE

- **NIF (clave principal)**
- NOMBRE
- APELLIDOS
- DIRECCION
- EMAIL
- CPOSTAL
- MOVIL
- TELEFONO

- HIJOS
- NACIONALIDAD

DESPERFECTO
- TIPO (clave)
- DESCRIPCION

OCUPACIÓN
- **NIF_CLIENTE (clave)**
- **NUM_APTO (clave)**
- **FECHA_ENTRADA (clave)**
- FECHA_SALIDA

Claves Principales

Claves Foráneas

FK_ocupacion_apartamento (NUM_APTO, apartamento.NUMERO)
FK_ocupacion_cliente (NIF_CLIENTE cliente.NIF)

PARTE
- **NIF_CLIENTE**
- **NIF_PROFESIONAL**
- **DESPERFECTO**
- **FECHA_CREACION**
- FECHA_VISITA1
- FECHA_VISITA2
- FECHA_VISITA3
- ARREGLADO
- DESCRIPCION
- NOTAS_PROF
- NUMERO_APTO

Claves Principales

Claves foráneas
FK_parte_apto (NUMERO_APTO, apartamento.NUMERO)
FK_parte_cliente (NIF_CLIENTE, cliente.NIF)
FK_parte_desperfecto (DESPERFECTOm desperfecto.TIPO)
FK_parte_profesional (NIF_PROFESIONAL, profesional.NIF)

PROFESIONAL
- **NIF (clave principal)**
- NOMBRE
- APELLIDOS
- DIRECCION
- EMAIL
- CPOSTAL
- MOVIL
- TELEFONO
- NOMBRE_TALLER
- DIRECCION_TALLER
- CPOSTAL_TALLER
- TELEFONO_TALLER

ARREGLA
- **NIF_PROFESIONAL**
- **DESPERFECTO**

Claves Principales

Claves foraneas
FK_arregla_desperfecto (DESPERFECTO, desperfecto.TIPO)
FK_arregla_profesional (NIF_PROFESIONAL, profesional.NIF)

4.- DISEÑO TÉCNICO

La arquitectura de la aplicación se basa en los estándares de la especificación J2EE de Sun, permitiendo la utilización de patrones y beans. El lenguaje de programación de esta especificación es Java.

A través de J2EE usamos el modelo de arquitectura MVC (Modelo – Vista – Controlador) basado en el patrón del mismo nombre.

Este patrón se implementará a través del marco de trabajo Struts versión 1.2.9. Se ha elegido esta versión al ser la más estable y probada, siendo de la cual se ha encontrado más documentación.

4.1.- PATRONES UTILIZADOS

4.1.1.- PATRON MVC (Modelo Vista-Controlador)

Es el patrón principal sobre el que se realiza la aplicación. Dicta la arquitectura que seguiremos. Este patrón separa la interfaz del usuario (páginas HTML) de la lógica de negocio (controlador de respuesta) y de los datos de la aplicación (Sistema de Gestión de Base de Datos).

Este Patrón permite:

- Separar la lógica de negocio de la presentación.
- Controlar el flujo de la navegación.
- Gestionar la lógica del negocio aparte de los datos.
- Normalizar las acciones que procesa el sistema.

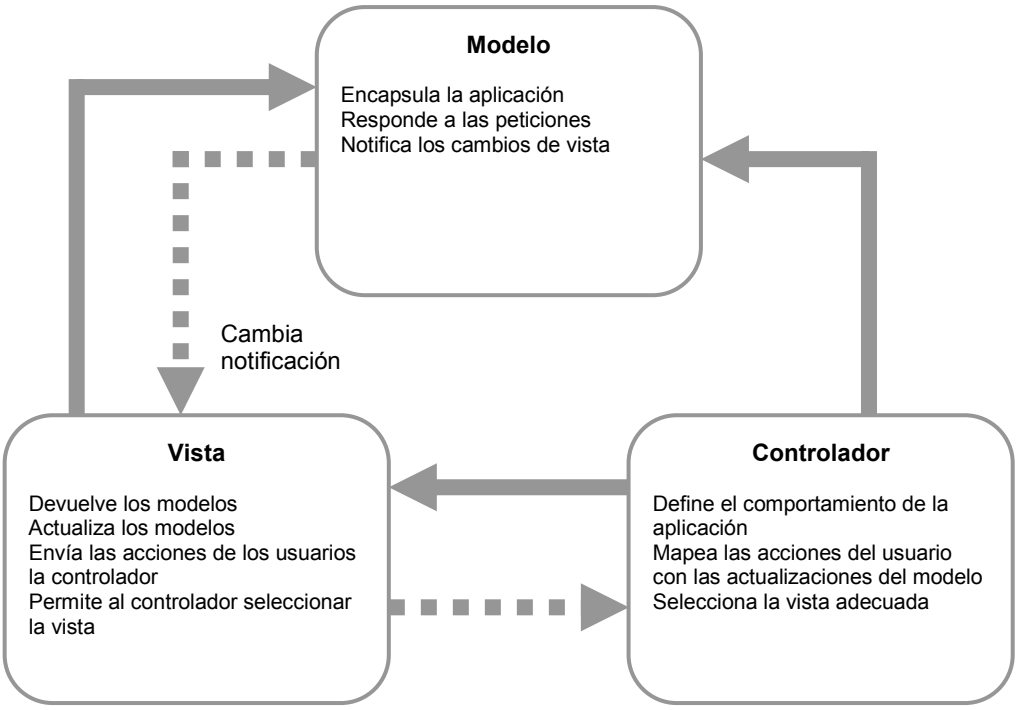


Ilustración 9 ***Patrón MVC***

El modelo se puede dividir en dos subsistemas: el **estado interno** del sistema y las **acciones** que se pueden tomar para cambiar el estado.

4.1.1.1- MARCO DE TRABAJO STRUTS (CONTROLADOR)

La parte controlador del diseño MVC realiza las siguientes funciones:

- Interceptar peticiones http desde un cliente.
- Traducir cada petición en una operación específica de negocio a realizar.
- Invocar la operación de negocio o delegarla a un manejador.
- Ayudar a seleccionar la siguiente vista a mostrar al cliente.
- Devolver la vista al cliente.

La forma en la que vamos a implementar este patrón de arquitectura es a través del *framework* Struts. Struts proporciona el componente **Controlador** (de la vista MVC) y se integra con otras tecnologías para proporcionar la Vista y el Modelo. Para el **Modelo**, Struts interactúa con otras tecnologías como JDBC y EJB, también puede interactuar con otros paquetes como Hibernate, iBATIS, u Object Relational Bridge. Para la **Vista,** Struts trabaja con JavaServer Pages, incluyendo JSTL y JSF, y otros sistemas de representación.

El Controlador del marco actúa como un puente entre el Modelo y la Vista de la aplicación. Cuando se recibe una petición, el Controlador invoca una clase **Action**. La clase Action consulta con el Modelo para examinar o actualizar el estado de la aplicación. Struts proporciona una clase **ActionForm** para ayudar a transferir datos entre el Modelo y la Vista.

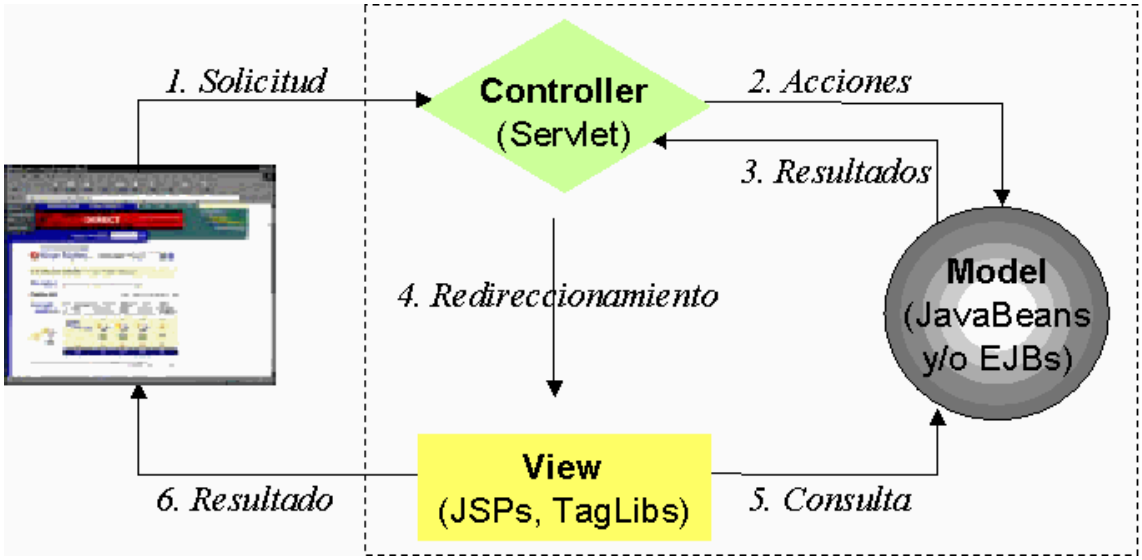


Ilustración 10 *Funcionamiento De Struts*

El controlador ya se encuentra implementado por Struts, pero por supuesto se puede heredar y ampliar o modificar si fuera necesario, el workflow de la aplicación se puede programar desde un archivo XML (struts-config.xml). Las acciones que se ejecutan sobre el modelo de objetos de negocio se implementan basándose en clases predefinidas por el framework y siguiendo el patrón Facade. Y la generación de interfaz se soporta mediante un conjunto de etiquetas predefinidos por Struts cuyo objetivo es evitar el uso de Scriplets (los trozos de código Java entre "<%" y "%>"), lo cual genera ventajas de mantenimiento y de ejecución (pooling de etiquetas, caching, etc).

Logísticamente, separa claramente el desarrollo de interfaz del workflow y lógica de negocio permitiendo desarrollar ambas en paralelo.

4.1.1.2.- PATRÓN COMPOSITE VIEW (VISTA)

La parte correspondiente a la **Vista** se implementa por medio del patrón **Composite View**. Este patrón nos permite descomponer las vistas (llamadas vistas compuestas) en subvistas atómicas. Cada componente de la plantilla se puede incluir dinámicamente dentro del todo y el diseño se puede mantener independientemente del contenido.

Al utilizar Struts como marco de trabajo, usaremos Tiles. Tiles es un sistema de plantillas que se puede usar para crear componentes de vista reutilizables. Aunque forma parte de Struts, también se puede utilizar independientemente.

4.1.2.- PATRÓN BUSSINES DELEGATE

Al desarrollar software por medio de un sistema multi-capa, existe el problema de que los componentes de una capa, interactúen directamente con los servicios de negocio. Esta interacción directa expone los detalles de la implementación del API de la capa de negocio a la capa de presentación. El resultado es que los componentes de la capa de presentación son vulnerables a los cambios en la implementación en la capa de negocio.

Este patrón permite reducir el acoplamiento entre la capa de presentación y la capa de negocio, ocultando los detalles de implementación de la parte de negocio. Actúa como una abstracción de negocio del lado del cliente, y oculta la implementación de la capa de negocio.

En nuestro caso nos permite abstraer las operaciones que realizan los distintos actores (Administrador, Cliente, Profesional y Sistema) de los actores en sí, pudiendo modificar cada capa por separado con solo modificar la interface del patrón y aislando la capa de negocio de las otras según el patrón MVC.

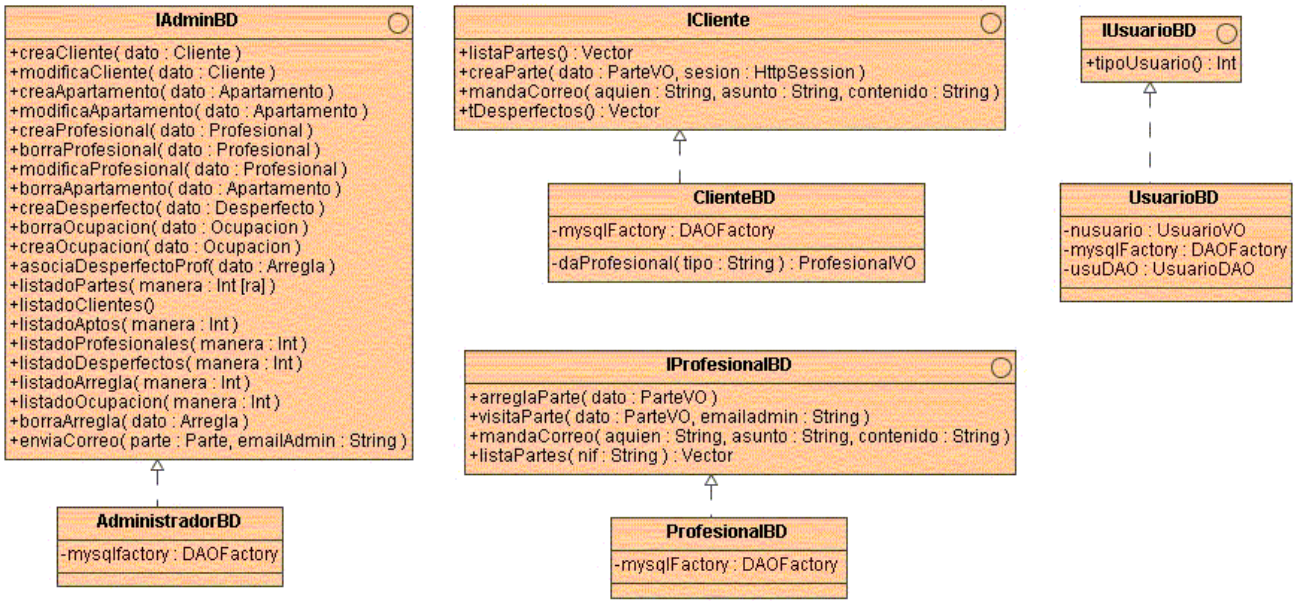


Ilustración 11 ***Patrón Bussines Delegate***

4.1.3.- CONTROL DE LA SESIÓN

Al no tener operaciones en común, excepto el envío de correos, se ha optado por un Listener para controlar el objeto session y no implementar el patrón Session Facade como tal.

El Listener es un objeto de Struts que nos permite crear un "oyente" para manejar la sesión. Podemos controlar las operaciones de la sesión:

- cada vez que se abra
- se cierre
- se almacene o se borre algún atributo
- se realice cualquier otra operación que se quiera realizar sobre la sesión.

4.1.4.- PATRÓN VALUE OBJECT

El patrón Value Object tiene por objeto abstraer los datos del sistema de almacenamiento que se use y de la lógica de negocio empleada.

Este tipo de objeto suelen ser "lightWeight" y de una naturaleza muy simple. No tienen ningún tipo de lógica de negocio y son simples contenedores de datos estructurales. Al tratarse de objetos transversales a todas las capas un cambio en la capa generadora de estos objetos, no implica un cambio en las demás capas.

Gracias a este patrón podemos migrar a otra base de datos cambiando solo la capa de consumo de datos (patrón DAO en nuestro caso) y no la lógica de la aplicación.

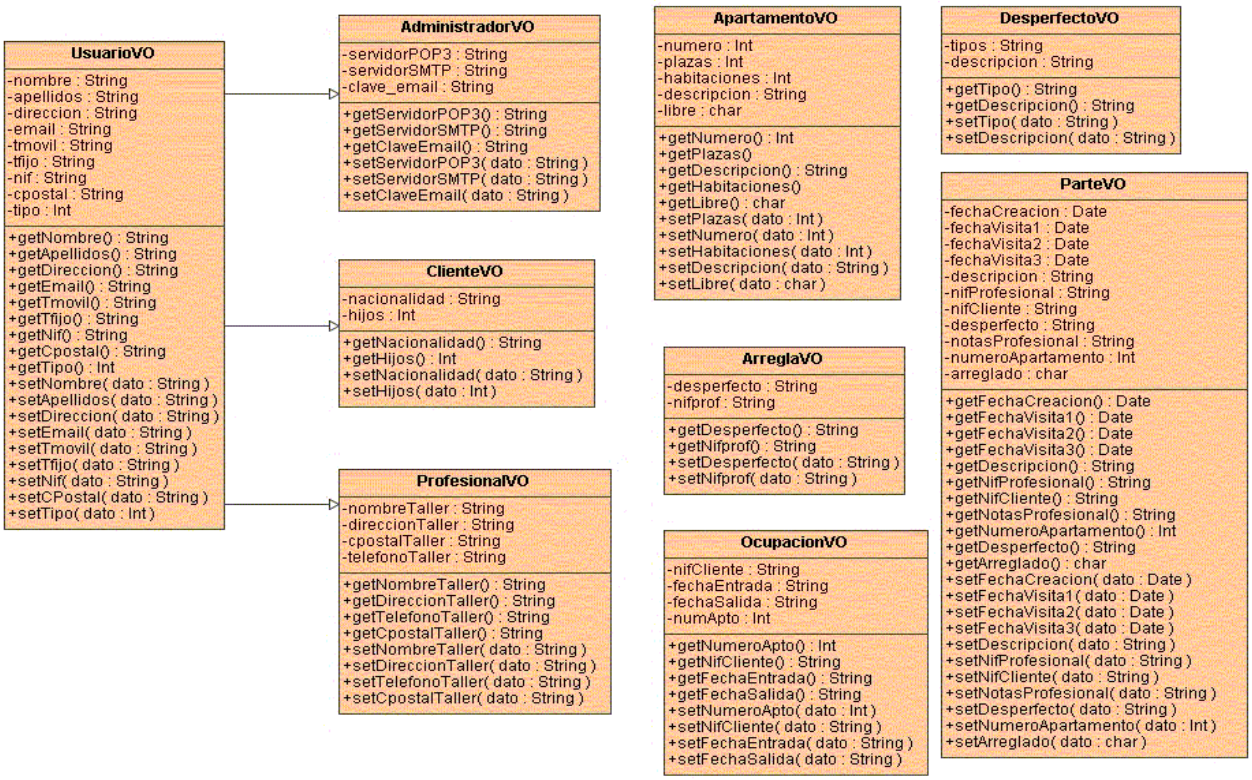


Ilustración 12 ***Patrón Value Object***

4.1.5.- PATRÓN DAO (Data Access Object)

Este patrón es muy útil para abstraer y encapsular todos los accesos a la base de datos, logrando desacoplar la lógica de negocio de la lógica de acceso a datos. DAO maneja la conexión con la fuente de datos para obtener y almacenar datos.

Este patrón permite:

- **Transparencia**: Se puede usar la fuente de datos sin conocer los detalles específicos de su implementación.
- **Facilita la Migración**: Es más fácil la migración a un Sistema Gestor de Base de Datos a otro.
- **Reduce la complejidad del código de los objetos de negocio**: El patrón DAO maneja toda la complejidad del acceso a los datos, simplificando el código de los objetos de negocio y de los clientes que usan DAO.
- **Centraliza todos los accesos a datos en una caja negra independiente**: Esto es debido a que todas las operaciones con los datos se delegan a esta caja negra DAO.

Puntos en contra de este patrón son:

- **No es útil para manejar persistencia con el contenedor**: Para los contenedores (CMP) que manejan persistencia este patrón no es necesario ya que es el servidor el que proporciona la transparencia. En nuestro caso la persistencia no se va a implementar.
- **Añade una capa extra**: Se añade una capa extra entre el cliente y la fuente de datos. Esta capa hay que diseñarla e implementarla.

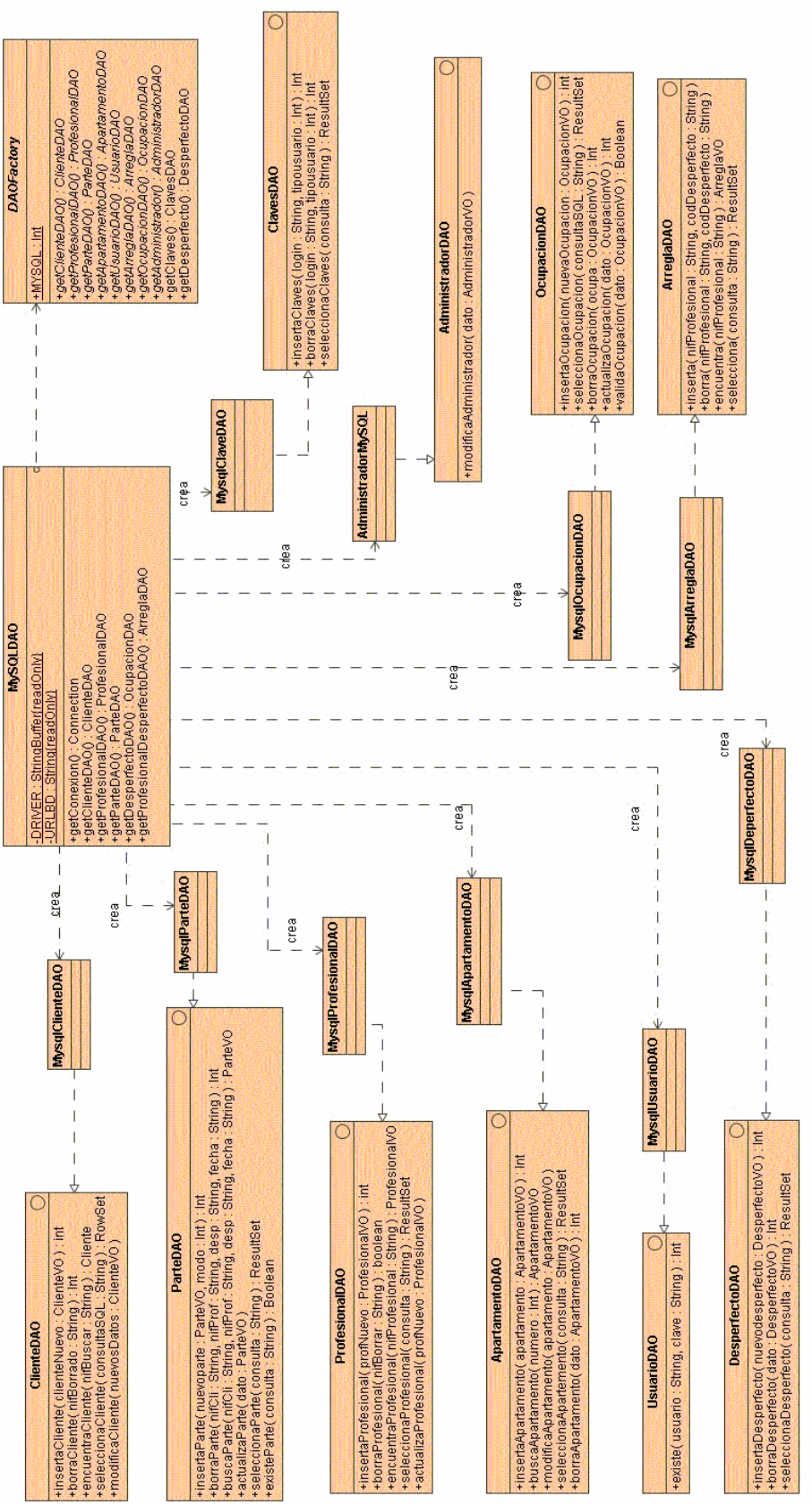


Ilustración 13 ***Patrón DAO (Data Access Object)***

5.- IMPLEMENTACIÓN

La implementación viene condicionada por los patrones usados, por el propio carácter de struts y de la aplicación.

5.1.- FUNCIONAMIENTO

La aplicación se divide en dos grandes capas; la capa WEB que corresponde a la vista, al controlador (estos dos apartados los ofrece struts) más el patrón Business Delegate que aparece como nexo de unión entre esta capa y la capa que contiene la lógica del negocio y de la base de datos (capa Negocio).

Dentro de la capa WEB se distinguen dos particularidades de Struts, que son: un plugin y un listener.

- El plugin llamado solamente se ejecuta al iniciar el servlet. En este plugin configuramos los parámetros que son comunes a toda la aplicación:
    - El nombre del archivo log de la aplicación y donde se almacena.
    - La cuenta de correo y la clave del administrador para poder enviar correo.

  Estos datos se recogen del web.xml.
- El listener es un "oyente" definido en el archivo de configuración de la aplicación "web.xml" que se encarga de recoger las incidencias de un objeto session. Con respecto al objeto session se controla:
    - Cuando se crea un objeto session, creando una serie de variables de dicho objeto como es la cadena de conexión con el SGBD recogiendo los parámetros del archivo web.xml, y añadiendo una reseña en el fichero log.
    - Cuando se destruye el objeto session, añadiendo un registro al fichero log de la aplicación.
    - Al crear, borrar o modificar un parámetro al objeto session

La arquitectura de la aplicación y la comunicación entre las capas será:

Paginas JSP → ActionForms (Struts) → Actions (Struts) → Business Delegate → Negocio (cliente, professional, administrador…) → bases de datos (DAO)

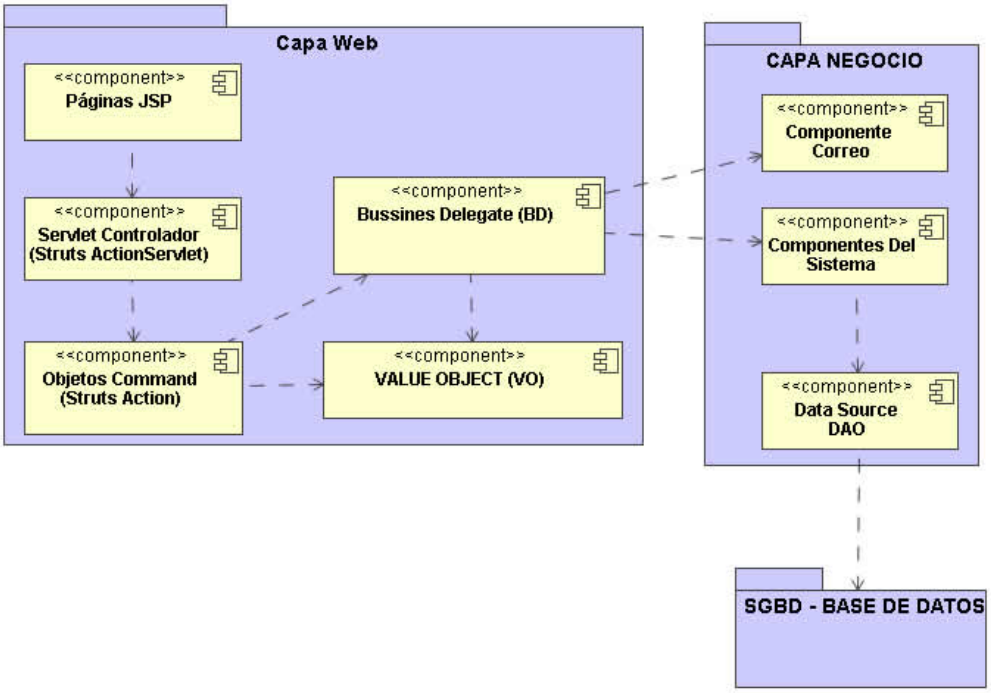


Ilustración 14 ***Implementación***

5.1.1.- CAPA WEB

Esta capa es la puerta de entrada a la aplicación, ya sea a través de las vistas, o del propio controlador. Cada vez que el servidor recibe una petición el servlet de la aplicación captura la petición y realiza lo necesario según el fichero de configuración (visto más abajo).

Una vez capturada la petición, se comprueba si existe algún mapeado en el fichero de configuración que mapee la petición con la instancia de **Action** apropiada. Esta clase **Action** sirve para desacoplar la petición de cliente de una operación de negocio.

Si la petición proviene de un form bean se usan los objetos **ActionForm** de Struts para pasar los datos de entrada del cliente entre el usuario y la capa de negocio. Struts recoge automáticamente la entrada de datos de la petición y los pasa al **Action** adecuado usando un *form bean*.

La clase Action correspondiente delegará en el patrón Business Delegate del usuario, que toma el control y realiza las operaciones necesarias.

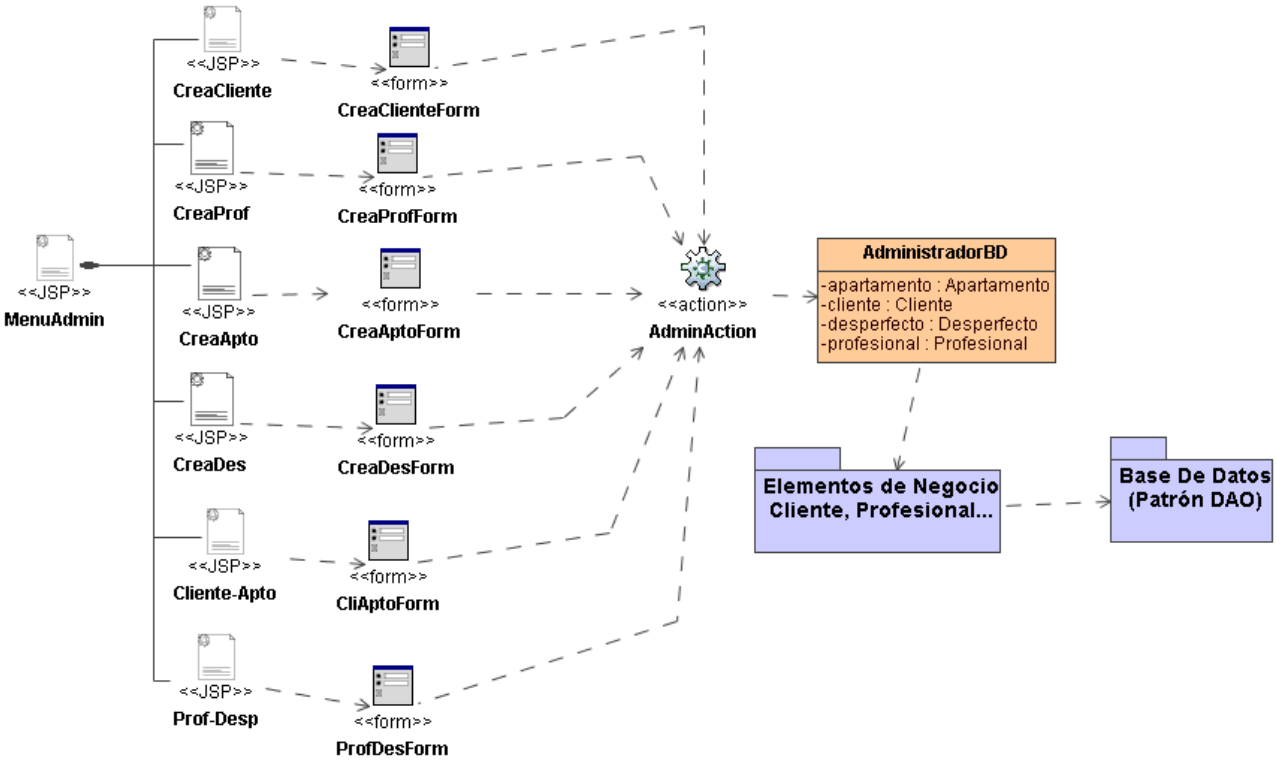


Ilustración 15 ***Funcionamiento del Usuario Administrador visto desde Struts***

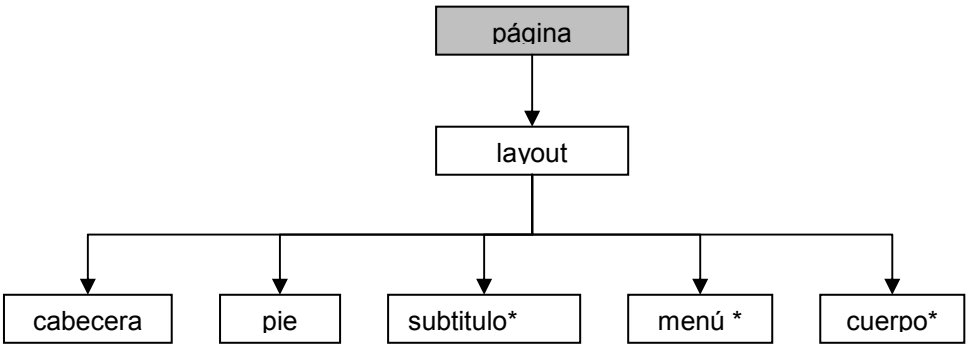
Para hacer del interfaz de usuario se ha usado Tiles como una implementación del patrón Composite View y JSP junto con HTML para la implementación de este interfaz.

Tiles esta configurado mediante un archivo llamado en este caso "`tiles-def.xml` " (se verá en el apartado de configuración), además debemos poner en cada archivo .jsp la directiva:

```
<%@ taglib uri="http://jakarta.apache.org/struts/tags-tiles"
prefix="tiles" %>
```

Necesaria para conocer las etiquetas que tiene Tiles.

La estructura que define las páginas es la siguiente:



Donde subtitulo*, menú* y cuerpo* son páginas que van cambiando dependiendo del usuario que esté usando la aplicación, y cabecera y pie son los mismos a lo largo de la aplicación.

5.1.2.- CAPA NEGOCIO

La lógica del negocio la componen las operaciones que pueden realizar los tres tipos de usuario que puede haber; Administrador, Cliente y Profesional (estas clases se han mostrado antes), además de Parte como eje fundamental sobre el que funciona la aplicación.

La realización de las operaciones contra la Base de Datos de la aplicación las lleva a cabo el patrón DAO y el envío de los emails a través de las clases necesarias conforman esta capa.

5.1.3.- MANEJO DE EXCEPCIONES

Se han creado dos excepciones específicas de la aplicación:

- GiedaException: Maneja las excepciones producidas por la aplicación, excepto las producidas por la ejecución de operaciones SQL. Se le pueden añadir mensajes. Sobrescribe getMessage() e implementa una función para poder añadirle mensajes nuevos AniadeMensaje(String mensaje) y para mostrarlo DaMensaje().
- ErrorSQL: Maneja las excepciones producidas por cualquier operación de tipo SQL contra las base de datos.

Estas dos son las principales excepciones que se controlan por la aplicación, una vez capturadas y tratadas se envían a la página adecuada (layouts/errores.jsp) que captura el mensaje de la excepción y lo formatea para mostrarlo.

5.2.- HERRAMIENTAS DE DESARROLLO

Se ha elegido Tomcat (en su versión 6.0.14) como servidor de la aplicación por varias razones:

- Es más fácil de instalar y mantener que los demás servidores de aplicación del mercado (Jboss, GlassFish, etc…).
- Necesita menos recursos que los anteriores.
- Está muy compenetrado con Struts y los servlets.
- Priorizar la velocidad sobre la seguridad, eligiendo los servlets más ligeros y rápidos.
- Nula conexión desde el exterior y poca al exterior (salvo emails).

El Sistema Gestor de Base de Datos seleccionado ha sido MySQL 5.0.16, por su eficacia, el ser software libre y ser uno de los gestores de datos más usados para aplicaciones.

No se ha creído oportuno utilizar persistencia ya que al ser una intranet, la conexión se considera segura, no habiendo cortes que puedan perder o deteriorar los datos.

Para el manejo de los datos se usa JDBC. Mediante el conector proporcionado por MySQL para java. Este conector usa directamente el protocolo MySQL para conectarse con el servidor, el lenguaje SQL es el estándar con las particularidades de MySQL.

El envío de los emails se realiza por medio de javamail 1.4.1. De todas las librerías que contiene solo usaremos:

```
mailapi.jar, smtp.jar
```

Ya que son las únicas necesarias para enviar mensajes. Para que javamail funcione es necesario el javabeans activation framework 1.1.1 JAF 1.1.1.

Por último el IDE para java usado ha sido NetBeans en su versión 6.0, y como JDK se eligió la versión 1.5, jdk-1-5_0.15 que viene con el jre (Java Runtime Environment) para poder compilar la aplicación y para que funcione.

5.3.- PROCESO DE INSTALACIÓN

Para instalar la aplicación, primero tenemos que descargar las librerías y el software que necesitamos para ejecutarlo, que son las siguientes:

- Struts 1.2.9: https://olex.openlogic.com/packages/struts#388
- Log4j: http://www.apache.org/dyn/closer.cgi/logging/log4j/1.2.15/apache-log4j-1.2.15.zip
- myslq-connector-java: http://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-java-5.1.6.zip/from/http://mysql.rediris.es/
- javamail: https://cds.sun.com/is-bin/INTERSHOP.enfinity/WFS/CDS-CDS_Developer-Site/en_US/-/USD/ViewProductDetail-Start?ProductRef=javamail-1.4.1-oth-JPR@CDS-CDS_Developer
- Java Beans Activation Framework (necesario para que javamail funcione): https://cds.sun.com/is-bin/INTERSHOP.enfinity/WFS/CDS-CDS_Developer-Site/en_US/-/USD/ViewProductDetail-Start?ProductRef=jaf-1.1.1-fcs-oth-JPR@CDS-CDS_Developer
- Standard Taglib 1.1: http://ftp.udc.es/apache-dist/jakarta/taglibs/standard/binaries/jakarta-taglibs-standard-1.1.2.zip

Para instalar el software MySQL 5.0, Tomcat 6.0 y jdk-1_5_0_15 ver los anexos donde se indica como realizarla.

Una vez descargadas las librerías, es recomendable instalarlas de la siguiente manera:

1. Se crea un directorio en C:; se le puede llamar “librerias-gieda”
2. Se descomprimen todos los archivos en este directorio.
3. Se copian los archivos “.jar” de cada uno de los subcarpetas que se generan en el directorio “lib” de Tomcat, que será: ”*/%TOMCAT_HOME%/lib/*”. Donde *%TOMCAT_HOME%* es el directorio donde se instaló el servidor Tomcat.

En la **Ilustración 16** se puede ver el directorio donde se deben copiar las librerías.

Una vez instalado y configurado el software previo así como las librerías en el directorio indicado, nos dispondremos a instalar la aplicación.

La distribución se realizará a través de un archivo .war. Este archivo se debe copiar en el directorio adecuado del servidor instalado anteriormente (Tomcat) para desplegar las aplicaciones. Dependiendo de que versión tengamos del servidor hará falta reiniciar o no, con Tomcat a partir de la versión 5 no es necesario reiniciar.

La estructura de directorios del servidor Tomcat, es la siguiente, y el directorio donde ha de copiarse el archivo .war, es el marcado con la línea roja, es el webapps:

- Tomcat: tomcat%home:\webapps

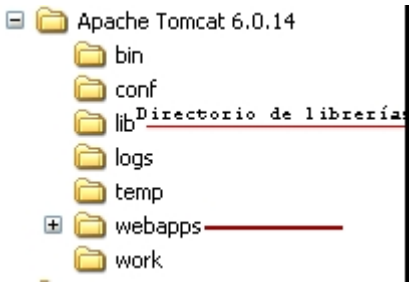


Ilustración 16 ***Estructura de directorios de Tomcat***

A la estructura anterior hay que asignarla un **context path** coincidente con el nombre con el que se quiera acceder a la aplicación mediante la URL. Cada aplicación web estará asociada a un contexto y todos los componentes existirán en relación a ese contexto.

5.3.2.- CONFIGURACIÓN

SERVIDOR DE BASE DE DATOS

Para conectarse a la base de datos se debe usar un nombre de usuario y una clave. Estos pueden variar dependiendo de como se haya instalado el servidor (Anexo I).

El valor del usuario y de la clave vienen establecidos en el archivo de configuración de la aplicación (`web.xml`):

- **`<param-name>gieda-passbd</param-name>`**
  `<param-value>valor de la clave</param-value>`
  Aquí escribiremos el valor de la clave con la que se accede a MySQL
- **`<param-name>gieda-logbd</param-name>`**
  `<param-value>valor del login</param-value>`
  Aquí escribiremos el valor del usuario para acceder a MySQL

La aplicación viene predeterminada con el usuario “root” y la clave “root”.

Las tablas y los datos escritos en sql, se encuentran en un archivo con extensión “`.sql`” (`gieda-sql.sql`). En el Anexo I se muestra las operaciones necesarias para poder crear las tablas y los datos de prueba.

APLICACIÓN

La aplicación dispone de varios ficheros de configuración:

- **`web.xml` fichero descriptor de despliegue**: el contenido de este fichero es fundamental para el despliegue de la aplicación ya que describe al contenedor web sus elementos y el modo en que se accede a los mismos. Además, define aspectos de seguridad, ficheros de bienvenida, parámetros iniciales, parámetros de contexto, etc. Cuando Tomcat (y, en general cualquier servidor J2EE compatible) se levanta, lee este fichero y, si contiene algún tipo de error, se lanzan excepciones que indican que el servidor no se ha levantado correctamente.

  En rojo aparecen los parámetros que se deben modificar:

  - **`<param-name>gieda-giedaflog</param-name>`**
    `<param-value>valor del parámetro </param-value>`
    Aquí escribiremos el nombre que tendría el archivo log (`.log`) de la aplicación.

- `<param-name>admin-email</param-name>`
  `<param-value>valor del parámetro</param-value>`
  Este será el email del Administrador de la aplicación con el que se mandarán los emails.

- `<param-name>admin-clave</param-name>`
  `<param-value>valor del parámetro</param-value>`
  La clave del Administrador para poder acceder al servidor SMTP y enviar los emails.

- `<param-name>smtp-server</param-name>`
  `<param-value>valor del parámetro</param-value>`
  El servidor SMTP elegido por el administrador para enviar los emails necesarios por la aplicación. La cuenta de correo y la clave deberán pertenecer a este servidor.

- **`struts-config.xml` archivo de configuración de struts:** Archivo de configuración de Struts. En él se mapean las acciones que va a realizar la aplicación en cada Action, se puede configurar los parámetros de inicialización del servlet, se definen los form-beans de cada formulario, se indica donde se ubica los ficheros de mensajes. También los plugin a utilizar como Tiles o Validator.
  Se configuran los parámetros generales como las excepciones y los forwards globales.

5.4.- JUEGOS DE PRUEBA

Usuarios para utilizar como juegos de prueba

email	nif	Tipo
mtojar@gmail.com	12345678A	0 (administrador)
mtojar@gmail.com	121212S	1 (cliente)
mtojar@ya.com	43345678I	1
mtojar@ya.com	345678H	1
mtojar@ya.com	23456378B	1
mtojar@gmail.com	500600976G	1
ra-ig@hotmail.com	12345678H	1
mtojar@ya.com	11123456W	2 (profesional)
ra-ig@hotmail.com	00123456G	2
ra-ig@hotmail.com	22123456F	2

Entrar como administrador:

Email: mtojar@gmail.com

Clave: 12345678A.

1. ENTRADA AL SISTEMA:
   a. LLAMADA: http://localhost:8080/gieda, entrada de datos del administrador, aceptación de los datos
   b. SALIDA: Menú del Administrador
2. MENU DEL ADMINISTRADOR

   a. LLAMADA: Selección “Creación de Apartamento”
   b. SALIDA: Formulario de Creación de Apartamento
3. AÑADIR APARTAMENTO
   a. LLAMADA: Completar el formulario y aceptar
   b. SALIDA CORRECTA: Vuelve a mostrar el menú del Administrador, si los datos son correctos.
   c. SALIDA ERROR: si el Apartamento ya existe.
4. AÑADIR CLIENTE
   a. LLAMADA: Completar el formulario y aceptar
   b. SALIDA CORRECTA: Vuelve a mostrar el menú del Administrador, si los datos son correctos.
   c. SALIDA ERROR: si el Apartamento ya existe.
5. AÑADIR PROFESIONAL
   a. LLAMADA: Completar el formulario y aceptar
   b. SALIDA CORRECTA: Vuelve a mostrar el menú del Administrador, si los datos son correctos.
   c. SALIDA ERROR: si el Profesional ya existe.
6. AÑADIR DESPERFECTO
   a. LLAMADA: Completar el formulario y aceptar
   b. SALIDA CORRECTA: Vuelve a mostrar el menú del Administrador, si los datos son correctos.
   c. SALIDA ERROR: si Desperfecto ya existe.
7. ASOCIA CLIENTE-APARTAMENTO
   a. LLAMADA: Completar el formulario y aceptar
   b. SALIDA CORRECTA: Vuelve a mostrar el menú del Administrador, si los datos son correctos.
   c. SALIDA ERROR: si el Apartamento está ocupado en las fechas elegidas.

Entrar como Cliente (vale cualquiera de los clientes que hay dados de alta o se puede crear uno):

Email: mtojar@gmail.com

Clave: 50060976G.

1. ENTRADA AL SISTEMA:
   a. LLAMADA: http://localhost:8080/gieda, entrada de datos del administrador, aceptación de los datos
   b. SALIDA: Menú del Cliente
2. CREACIÓN DE PARTE:
   a. LLAMADA: Completar el formulario y aceptar

b. SALIDA CORRECTA: Vuelve a mostrar el menú de Cliente, si se ha podido realizar la creación del parte y enviar los emails necesarios.

c. SALIDA ERROR: si algún correo no se ha podido enviar o ha ocurrido algún error al añadir el Parte.

Entrar como Profesional (vale cualquiera de los profesionales que hay dados de alta o se puede crear uno):

Email: mtojar@gmail.com

Clave: 50060976G.

1. ENTRADA AL SISTEMA:
   c. LLAMADA: http://localhost:8080/gieda, entrada de datos del administrador, aceptación de los datos
   d. SALIDA: Menú del Profesional
2. CREACIÓN DE PARTE:
   d. LLAMADA: Visita: Cuando el profesional realiza una visita para reparar un parte
      i. SALIDA CORRECTA: Vuelve a mostrar el menú del Profesional, si se ha podido realizar la creación del parte y enviar los emails necesarios.
      ii. SALIDA ERROR: si algún correo no se ha podido enviar o ha ocurrido algún error al añadir el Parte.
   e. LLAMADA: Listado de Partes por Arreglar: Mostrará un listado con los partes que le quedan por arreglar al profesional.
      i. SALIDA CORRECTA: Vuelve a mostrar el menú del Profesional, si se ha podido realizar la creación del parte y enviar los emails necesarios.
      ii. SALIDA ERROR: si algún correo no se ha podido enviar o ha ocurrido algún error al añadir el Parte.

6.- CONCLUSIONES

Cuando elegí el TFC de J2EE, mucha gente me comento que entraba en un mundo infernal y en efecto así es debido a la gran cantidad de posibilidades, entre las que escoger, una es realmente difícil. La complejidad de configuración, característica común de todo el software relacionado con esta tecnología.

Tema aparte es las posibilidades de diseño, los patrones. Todos parecen útiles, todos son fantásticos, pero no se pueden utilizar todos, hay que elegir bien cuales utilizar de entre ellos.

Pero no todo es malo, el TFC me ha servido como puerta de entrada para descubrir estas tecnologías, el software relacionado y lo mucho que puede ofrecer. Tanto que ahora estoy realizando aplicaciones en el trabajo desarrolladas sobre esta tecnología.

Respecto a GIEDA, tiene grandes posibilidades de ampliación:

- Añadiendo un sistema de reservas online.
- Una tienda para que los usuarios del edificio puedan comprar lo que necesiten.

7.- BIBLIOGRAFÍA

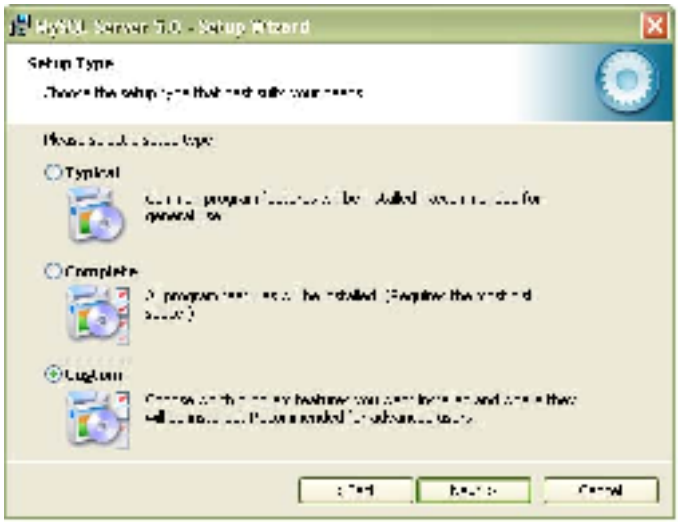
- Beginning Apache Struts: From Novice to Professional. Arnold Doray. Ed. Apress. ISBN: 1590596043.
- Jakarta Struts. Chuck Cavaness. Ed O´Reilly Anaya.8441518602.
- Jakarta-Struts Live. Rick Hightower. Ed. SourceBeat. ISBN: 097884308.
- Piensa en Java 4 Edición. Bruce Eckel. Ed: Pearson-Prentice may. ISBN: 9788489660342.
- Recogida y documentación de requisitos. Benet Campderrich Falgueras. UOC. P01/75007/00568.
- Software Architectura Design Patterns in Java. Partha Kuchana. Ed. Auerbach ISBN: 0849321425.
- UML (I): el modelo estático. Benet Campderrich Falgueras, Recerca Informatica, S.L. UOC P01/75007/00566.
- UML 2. Jim Arlow, Ila Neustadt. Ed. Anaya Multimedia. ISBN: 844152033X
- http://www.monografias.com/trabajos28/aplicacion-paso-paso-struts/aplicacion-paso-paso-struts.shtml
- http://www.tic.udc.es/~fbellas/teaching/is-2002-2003/
- http://www.programacion.net/
- http://www.adictosaltrabajo.com
- http://www.theserverside.com

Anexo I

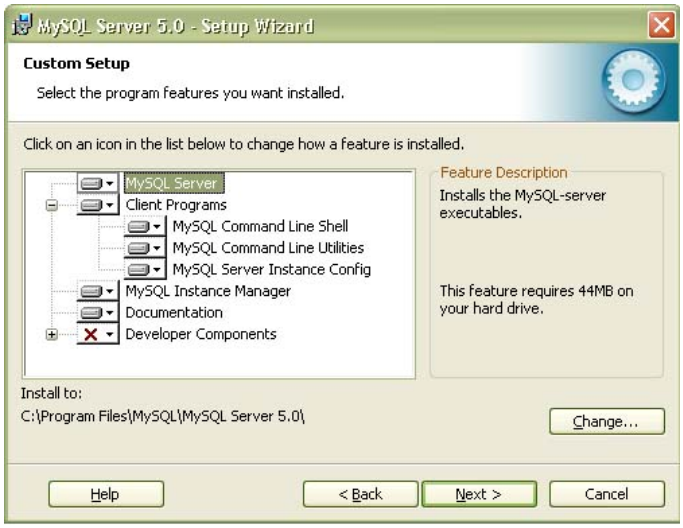
CÓMO INSTALAR Y CONFIGURAR MYSQL SERVER EN WINDOWS

Mostramos paso a paso, cómo instalar MySQL Server (Base de Datos SQL gratuita y muy difundida por Internet):

En primer lugar necesitaremos disponer del programa de instalación. Se puede descargar gratuitamente de "http://dev.mysql.com/downloads". Una vez descargado el programa de instalación de MySQL (versión que queramos, en nuestro caso instalaremos la " http://dev.mysql.com/get/Downloads/MySQL-5.0/mysql-5.0.51b-win32.zip/from/pick#mirrors" lo ejecutaremos y seguiremos las instrucciones que nos muestra el asistente de instalación:



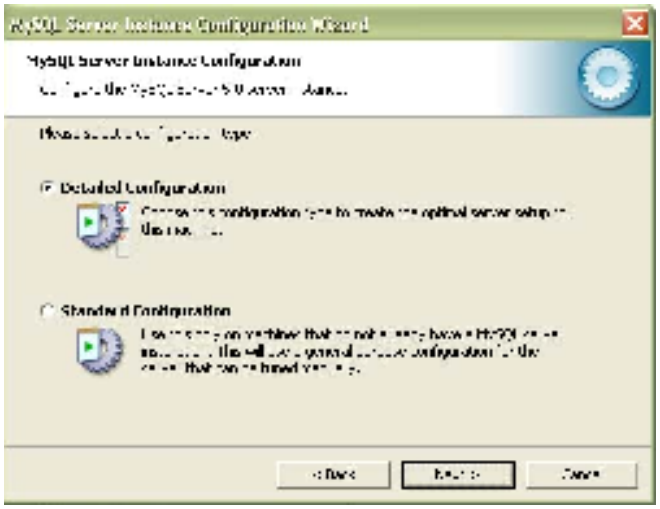
En la primera pantalla pulsamos "Next" aparece esta imagen y marcamos "Custom":



Seleccionamos las utilidades a instalar, por defecto se instalará todo salvo las herramientas para desarrolladores (sólo necesarias para desarrollos en Perl, C++ y MySQL Embedded Server):

Pulsamos en "Next" y a continuación en "Install":

A continuación se nos mostrará dos ventanas sobre publicidad de productos MySQL, pulsando “Next” en las dos. Si queremos configurar MySQL en este momento dejaremos marcada la opción "Configure the MySQL Server now" y pulsaremos en "Finish":



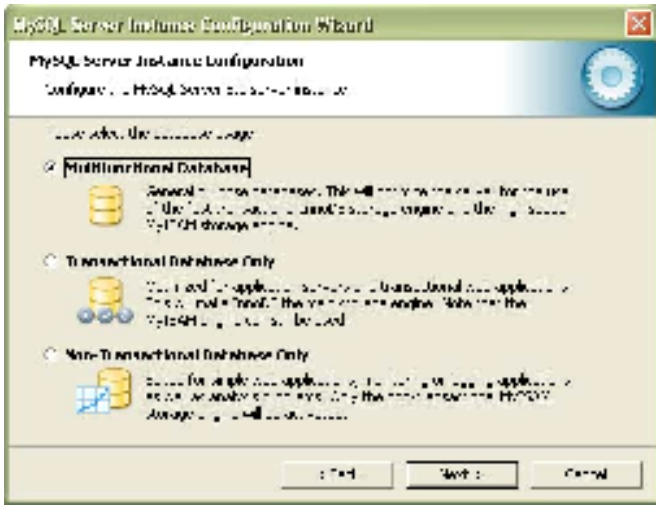
Seguidamente aparecerá un asistente para la configuración "MySQL Server Instance Configuration Wizard" y pulsaremos en "Next":

Marcaremos la opción "Detailed Configuration" y pulsaremos en "Next", de esta forma podremos configurar más opciones de MySQL utilizando el asistente. Si marcásemos "Standard Configuration" el asistente nos pediría menos información pero habría que configurar algunas opciones manualmente:

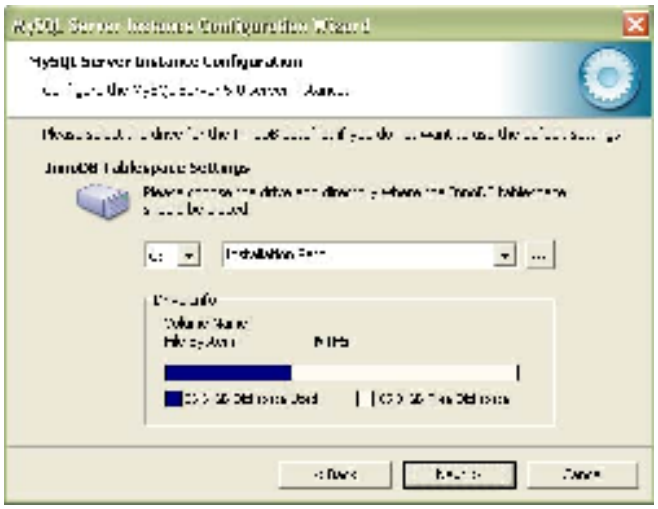
Dependiendo del uso que queramos dar al equipo en el que se instala, marcaremos una de las tres opciones:

- **Developer Machine**: marcaremos esta opción si en el equipo donde hemos instalado MySQL Server se utiliza también para otras aplicaciones. MySQL Server utilizará la memoria mínima necesaria.
- **Server Machine**: marcaremos esta opción si vamos a utilizar el equipo para algunas aplicaciones (no demasiadas). Con esta opción MySQL Server utilizará un nivel medio de memoria.
- **Dedicated MySQL Server Machine**: marcaremos esta opción sólo si queremos utilizar el equipo como un servidor dedicado exclusivamente a MySQL. Con esta opción MySQL Server utilizará el máximo de memoria disponble. Se obtendrá un rendimiento elevado pero el equipo sólo servirá para MySQL.

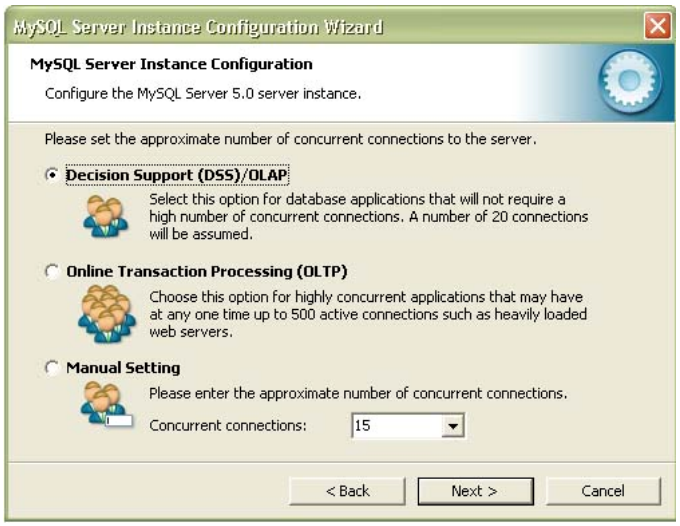
En nuestro caso marcaremos "Developer Machine" (consume el mínimo de memoria necesaria para su funcionamiento), este tipo de configuración de la instancia de MySQL no es recomendable si la base de datos va a soportar múltiples conexiones concurrentes con un volumen importante de información. Aunque puesto que nosotros la utilizaremos para desarrollar software será suficiente:



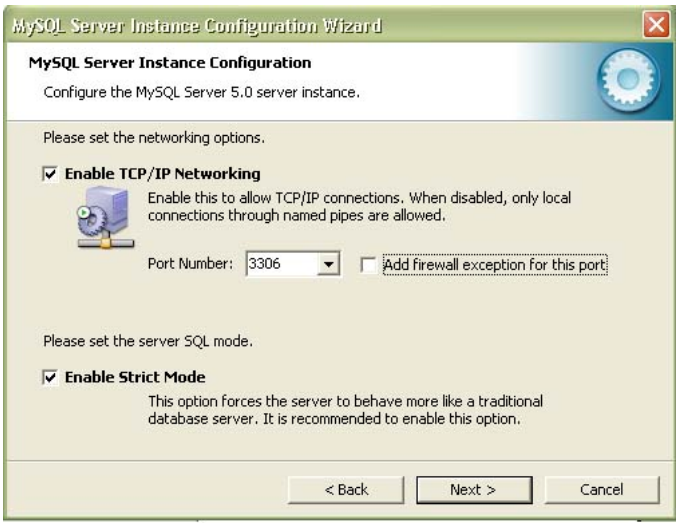
Dependiendo del uso que queramos dar a la Base de Datos marcaremos una de las tres opciones siguientes, normalmente se marcará "Multifunctional Database" salvo que queramos utilizar MySQL como base de datos para transacciones de otra Base de Datos MySQL:



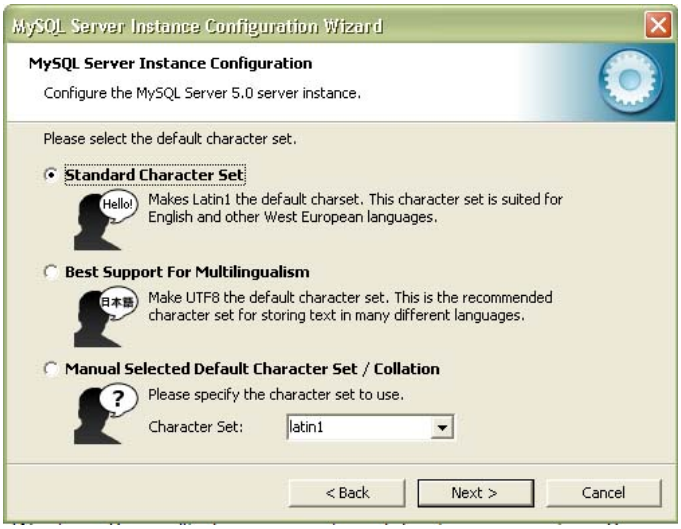
Seleccionaremos la unidad y la carpeta donde queramos guardar los ficheros de datos (Tablespace) de la Base de Datos. A partir de la versión 4.0 de MySQL incorpora soporte para el control de la integridad referencial. A este nuevo tipo de tablas lo llama InnoBD.



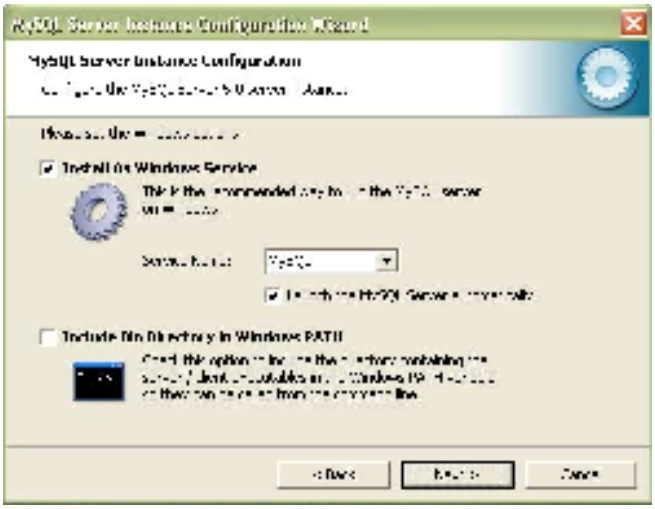
Elegimos ahora el número aproximado de conexiones concurrentes (varios clientes conectados a la vez) que tendrá nuestro servidor de MySQL). La primera opción asume unas 20, la segunda unas 500 y la tercera permite especificarlas manualmente. Este parámetro es aproximado no tiene por qué ser exacto:



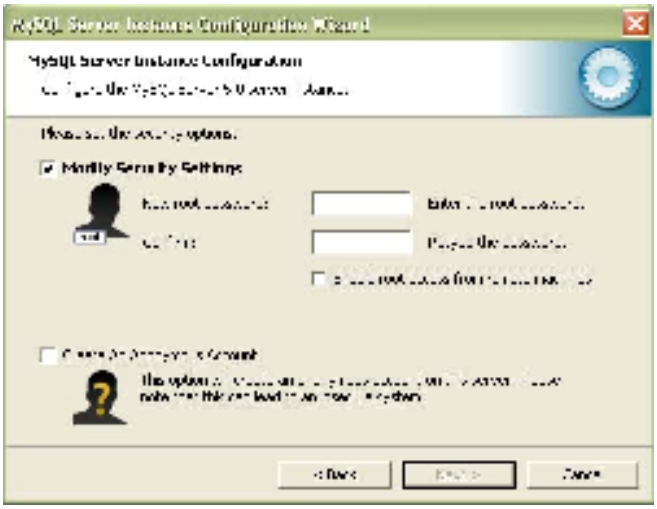
Dejaremos marcada la opción "Enable TCP/IP Networking" si queremos que los clientes se puedan conectar mediante TCP/IP al equipo servidor de MySQL. Podremos cambiar el puerto por el que lo harán, por defecto se suele dejar 3306 (si tenemos instalado algún cortafuegos deberemos abrir dicho puerto), a menos que seleccionemos "Add firewall exception for this port" que lo hará el mismo:



Seleccionaremos el juego de caracteres que queramos utilizar, por defecto está marcado "Latin1" válido para Inglaterra y Europa:

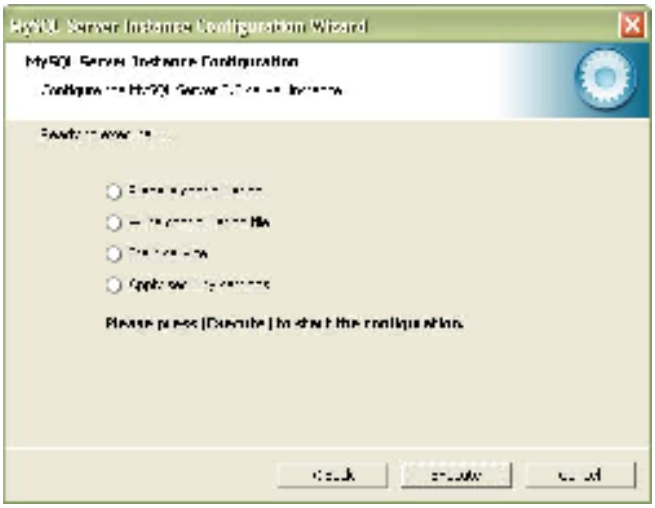


El siguiente paso es importante pues nos pide que especifiquemos el tipo de arranque de MySQL Server. Si seleccionamos la primera opción ("Install As Windows Service") el programa de instalación nos creará un Servicio que será el encargado de ejecutar MySQL Server, también nos permite especificar el nombre del servicio y si queremos que arranque automáticamente al iniciar el sistema ("Launch the MySQL Server automatically"). La segunda opción "Include Bin Directory in Windows PATH añadirá las variables de entorno necesarias para la ejecución de los ficheros necesarios para iniciar MySQL .

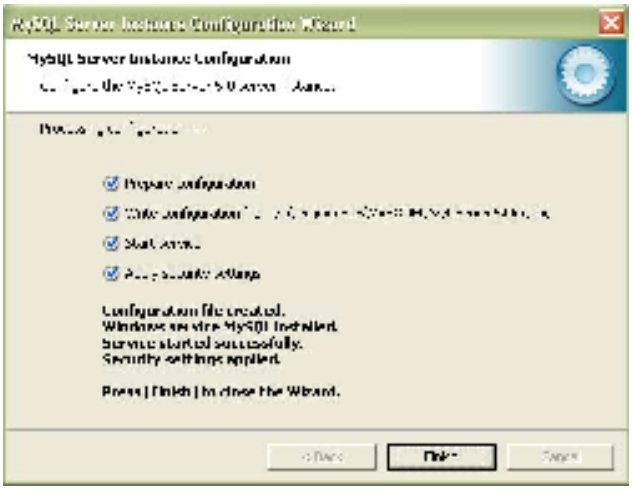


La opción recomendada es "Install As Windows Service":

Introduciremos la contraseña para el usuario administrador (root) y marcaremos la opción "Enable root access from remote machines" si queremos que se pueda acceder como administrador desde otros equipos. La aplicación viene preparada para que la contraseña sea "root", si la cambiamos deberemos hacerlo también en el archivo de configuración de la aplicación como se indica arriba":



Por último pulsaremos en "Execute" para fnalizar la configuración de MySQL:



Si no hay problemas mostrará esta ventana indicando que el proceso de instalación y configuración de MySQL Server ha terminado y se ha instalado e iniciado el Servicio que ejecutará MySQL:

Tras la instalación podemos comprobar (si hemos seleccionado la opción de iniciar MySQL como servicio) que el servicio se está ejecutando. Esto se puede ver en el administrador de tareas:

Nos aparecerá un servicio con el nombre "mysqld-nt.exe" que, como se puede observar, usa unas 12 MB de memoria RAM (sin conexiones de clientes).

Si lo deseamos podemos volver a configurar la instancia de MySQL desde `"Inicio" - "Programas" - "MySQL" - "MySQL Server 5.0" - "MySQL Server Instance Config Wizard"`. El asistente que aparecerá será similar al explicado en el programa de instalación.

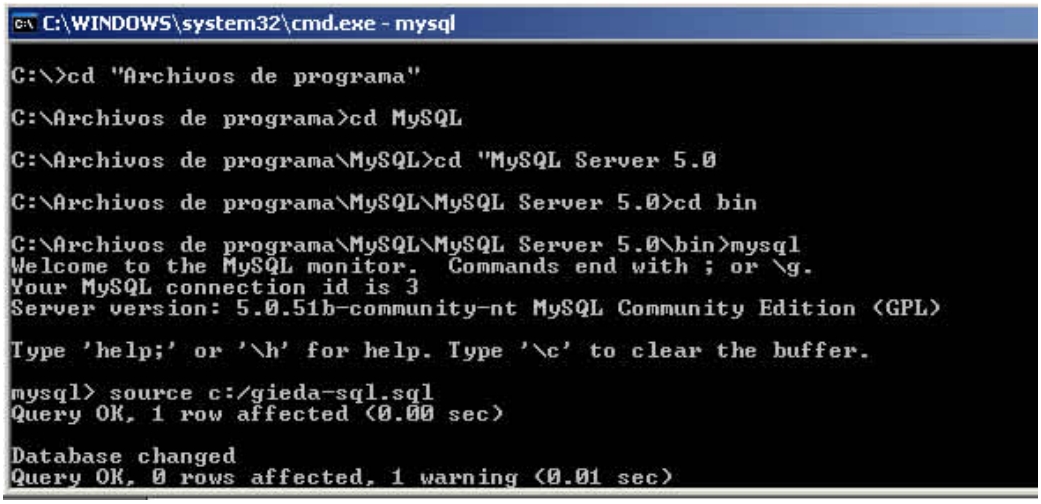
También podremos configurar mediante la línea de comandos MySQL, para ello iremos a `"Inicio" - "Programas" - "MySQL" - "MySQL Server 5.0" - "MySQL Command Line Client"`. Nos pedirá una contraseña (la que hayamos introducido anteriormente en la instalación):

Para crear la base de datos, así como crear las tablas y generar los datos de prueba para poder probar la aplicación usaremos el fichero "`gieda-sql.sql`" suministrado, una vez copiado al directorio raíz C. Mediante el comando:

```
source c:/gieda-sql.sql;
```

Hemos ejecutado todos los comandos almacenados en este fichero

En caso de no encontrar `"Inicio" - "Programas" - "MySQL" - "MySQL Server 5.0" - "MySQL Command Line Client"`, podremos hacerlo de la siguiente forma `"Inicio" - "Ejecutar" - "cmd"` a continuación se abrirá una ventana como la siguiente:



Se puede ver como se selecciona el directorio de instalación de Mysql Server 5.0, seleccionando el directorio "bin", y ejecutando "`mysql`" para poder conectarse a MySql, y a continuación ejecutar la instrucción:

`source c:/gieda-sql.sql;`

Otra manera es todo junto en una sola instrucción, realizando los siguientes pasos: "`Inicio`" - "`Ejecutar`", a continuación en el espacio que nos deja escribiremos los siguiente, con comillas inclusive:

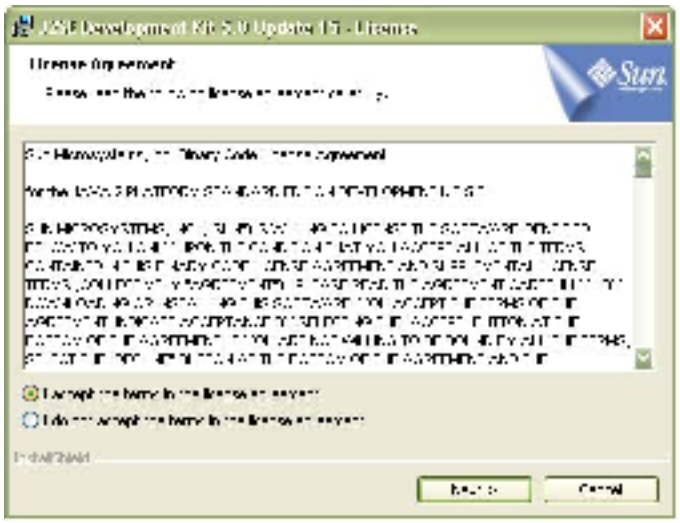
`"C:\Archivos de programa\MySQL\MySQL Server 5.0\bin\mysql.exe"`

Con lo que se abrirá la ventana anterior pudiendo teclear la instrucción para crear y rellenar las tablas con los datos de prueba.

Anexo II

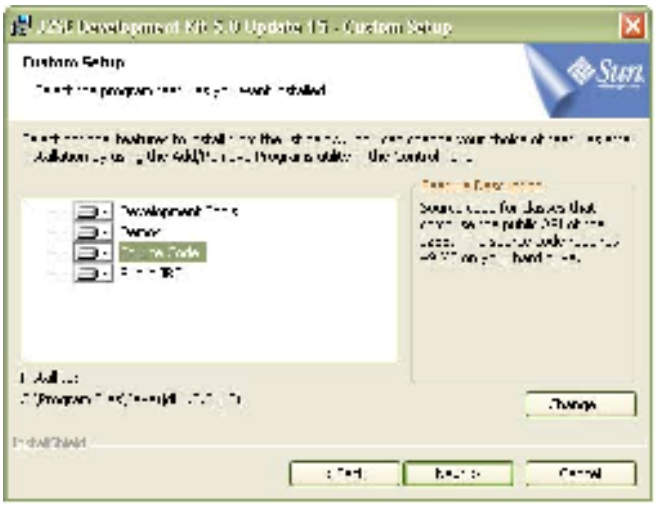
CÓMO INSTALAR Y CONFIGURAR jdk-1_5_0_15 EN WINDOWS

Mostramos paso a paso cómo instalar el jdk de Sun, en concreto la versión 1_5_0_15 en Windows:

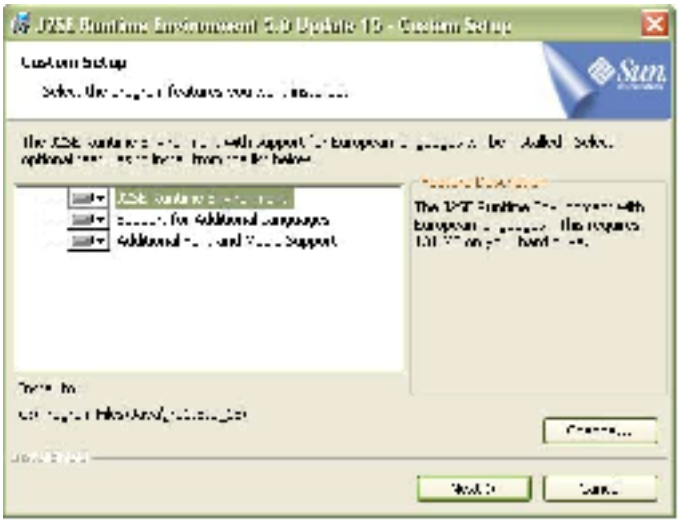


En primer lugar necesitaremos disponer del programa de instalación. Se puede descargar gratuitamente de "http://java.sun.com/javase/downloads/index_jdk5.jsp". Una vez descargado el programa de instalación lo ejecutaremos y seguiremos las instrucciones que nos muestra el asistente de instalación:

Aceptamos el acuerdo de licencia y pulsamos "Next"



Seleccionamos las opciones que deseemos, por defecto se instala todo. Dependiendo del uso que le queramos dar seleccionaremos la que más nos convenga.



Seguidamente pasamos a elegir las opciones para J2SE Runtime Environment . Por defecto está seleccionado todo, si no vamos a usar los lenguajes opcionales podemos decirle que no se instale, haciendo click en "Next" para continuar

Seleccione los componentes que desee o deje los que vienen por defecto. Marcando la opción "Next" para finalizar la instalación.

Posteriormente debemos configurar la variables para que java funcione adecuadamente que son: CLASSPATH y PATH:

Seleccione "`Inicio`" – "`Panel de Control`" – "`Sistema`. Seleccione la pestaña "`Opciones Avanzadas`" y posteriormente "`Variables de Entorno`" ("`Environment Variables`").

Localice "`Path`" en las variables del usuario ("`User Variables`") y variables del Sistema ("`System Variables`"). Debe agregar la ruta de acceso para el JDK **al final** de esta variable, un valor típico es el siguiente:

```
C:\%Directorio de Instalación%\Java\jdk1.5.0_<numero_version>\bin
En nuestro caso es C:\Program Files\Java\jdk1.5.0_15\bin
```

No altere los valores actuales de la variable PATH . La variable PATH es una serie de directorios separados por punto y comas (;).Windows localiza sus ejecutables a través de estas definiciones, por lo que debe tomar cautela con su modificación.

Para modificar o crear la variable CLASSPATH, actuaremos igual. Una vez que estamos en "Variables de Entorno" buscamos la variable, en caso de que no esté seleccionamos la opción "Nueva" y la crearemos. En la zona de "Nombre de Variable" pondremos "CLASSPATH" y en "Valor de la Variable" pondremos "C:\%directorio de

instalación de java%\jdk1.5.0_15\lib; C:\%directorio de instalación de java%\jre1.5.0_15\lib"

Una vez modificado seleccione `Set - OK - Apply`

Los cambios a la variable `PATH` tomaran efecto en cada ventana de comando ("`Command`"), inmediatamente confirmada su modificación.

Anexo III

CÓMO INSTALAR Y CONFIGURAR TOMCAT 6.0.16 EN WINDOWS

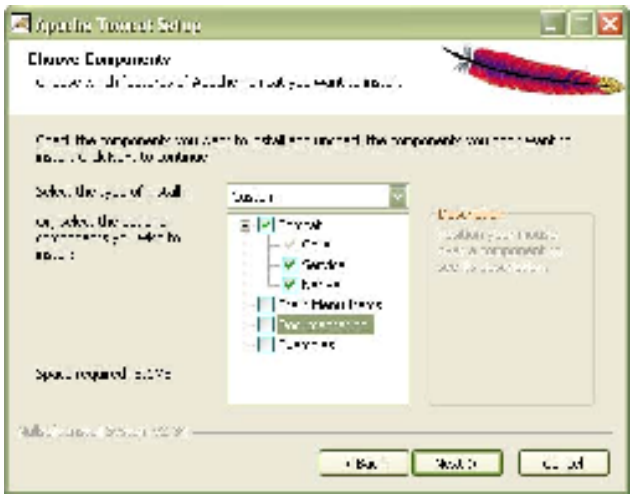
Mostramos como instalar Tomcat (servidor basado en Apache que nos permite ejecutar aplicaciones basadas en servlets, muy difundido por Internet y software libre) paso a paso.

En primer lugar tenemos que descargar tomcat

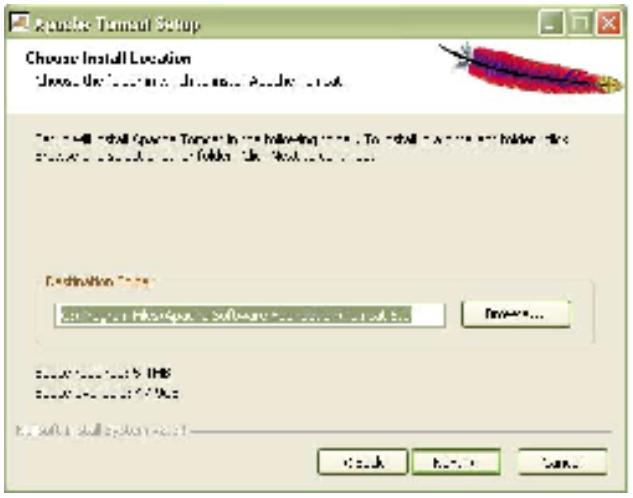
"http://tomcat.apache.org/download-60.cgi" y elegimos el mirror que este más cerca de nosotros. En nuestro caso elegiremos el de "ftp.udc.es: http://tomcat.apache.org/download-60.cgi?Preferred=http%3A%2F%2Fftp.udc.es%2Fapache-dist", a continuación elegimos, la distribución binaria, el formato windows service installer: "http://ftp.udc.es/apache-dist/tomcat/tomcat-6/v6.0.16/bin/apache-tomcat-6.0.16.exe" lo ejecutaremos y seguiremos las instrucciones que nos muestra el asistente de instalación:

Pulsaremos en "Next".

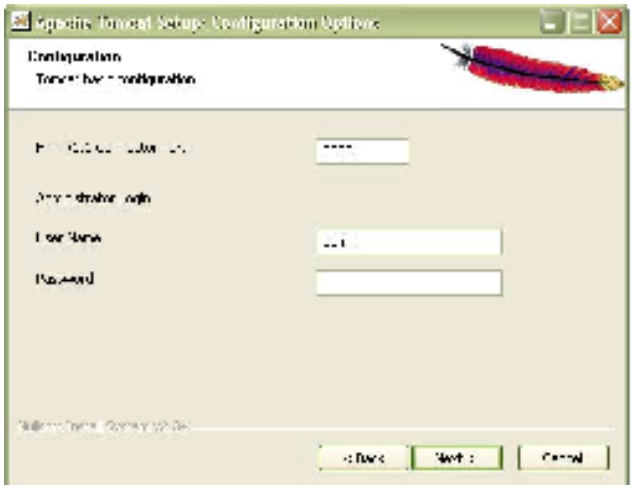
Aceptaremos la licencia pulsando en "I Agree" y continuamos.



Seleccione los componentes a instalar, por defecto se instalaran “Tomcat-Core”, “Start Menu Items” y “Documentation”. Es recomendable que seleccione “Tomcat-Service” para que funcione como un servicio de Windows y “Tomcat-Native” para que funcione mejor en entornos de producción. Pulsamos “Next” y continuamos:



Elegimos el directorio donde se instalará el servidor, si deseamos instalarlo en otro distinto del que viene por defecto, tendrá que teclearlo, o seleccionarlo a través del Explorador de Windows pulsando “Browse”. Seleccionamos la opción “Next” y continuaremos la instalación:



A continuación nos permite elegir el puerto donde escuchará el servidor, por defecto es 8080. También el nombre y la clave del usuario Administrador. Es conveniente que recuerde la clave. Para continuar seleccionamos “Next”:

Una vez que ha descargado e instalado los componentes que hemos seleccionado nos da la opción de ejecutar el servidor inmediatamente y leer el archivo Readme, no es necesario que seleccione nada. Pulsamos en “Finish” y terminamos la instalación.

Tras la instalación puede comprobar que el servidor está instalado como un servicio Windows, seleccione “Inicio” – “Panel de Control” – “Herramientas Administrativas” – “”Servicios”. Veremos un servicio que se llama “Apache Tomcat” como se muestra a continuación:

La dirección que nos permite saber si la instalación es correcta es la siguiente; http://localhost:8080, donde 8080 tiene que sustituirlo por el número de puerto que haya tecleado. La pantalla que debe de ver es la siguiente: