Universidad Carlos III de Madrid

Escuela Politécnica Superior

Ingeniería de Telecomunicación

Proyecto Fin de Carrera

Herramienta para el Servicio de Información Administrativa del Gobierno de Cantabria

Fecha: Diciembre de 2011

Autor: Elia Moreno Catalán

Tutor: Julio Villena Román

Título: Herramienta para el Servicio de Información Administrativa del Gobierno de Cantabria

Autor: Elia Moreno Catalán

Tutor: Julio Villena Román

EL TRIBUNAL

Presidente:

_____

Secretario:

_____

Vocal:

_____

Realizado el acto de defensa del Proyecto Fin de Carrera el día __ de _____ de _____ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de:

Fdo: Presidente Fdo: Secretario Fdo: Vocal

Resumen

El proyecto descrito en este documento consiste en el desarrollo de una herramienta de utilidad para los empleados del Servicio de Información Administrativa del Gobierno de Cantabria. Se trata de una aplicación web que permite llevar a cabo un conjunto de actividades a los trabajadores de dicho departamento.

Los operadores telefónicos que responden a las llamadas recibidas por el Servicio de Información Administrativa utilizan esta herramienta para realizar las consultas solicitadas, y para registrar la información obtenida en cada llamada atendida a un usuario.

Los responsables del servicio acceden a la aplicación para obtener conclusiones acerca de los datos introducidos por los operadores. Además, pueden también llevar a cabo determinadas tareas de administración de la herramienta.

La aplicación desarrollada se encarga de almacenar los datos recogidos por los operadores telefónicos, y tratar esta información con el fin de obtener los resultados solicitados por el administrador, presentando las conclusiones extraídas acerca del funcionamiento del servicio.

ÍNDICE GENERAL

APARTADO PÁGINA

1. INTRODUCCIÓN 1
1.1. MOTIVACIÓN 1
1.2. OBJETIVOS 2
1.3. ESTRUCTURA DEL DOCUMENTO 3

2. ESTADO DEL ARTE 5
2.1. APLICACIONES WEB 5
2.1.1. Definición de aplicación web *5*
2.1.2. Servidor web, contenedor web y servidor de aplicaciones *5*
2.1.3. Módulos web en J2EE *7*
2.1.4. Descriptor de despliegue *7*
2.1.5. Ciclo de vida de una aplicación web *7*
2.2. JAVA SERVLETS 8
2.2.1. Funciones de un servidor *8*
2.2.2. Implementación de un servidor *8*
2.2.2.1. Primera aproximación 8
2.2.2.2. Extensiones del servidor 8
2.2.3. Contenedor de servlets *9*
2.2.3.1. Esquema general 9
2.2.4. La API Servlet *9*
2.2.4.1. El paquete javax.servlet 10
2.2.4.2. El paquete javax.servlet.http 11
2.3. JAVASERVER PAGES 11
2.3.1. Concepto de página JSP *11*
2.3.1.1. HTML 11
2.3.1.2. Server-Side Includes 12
2.3.1.3. Definición de página JSP 12
2.3.2. Uso de servlet o JSP *12*
2.4. MODELOS DE ARQUITECTURA JSP 13
2.4.1. Arquitectura Modelo 1 ..... *13*
2.4.2. Arquitectura Modelo 2 ..... *13*
2.5. BASE DE DATOS 17
2.5.1. SQL *17*
2.5.1.1. MySQL 17
2.5.2. JDBC *17*
2.5.3. SQL y JDBC *17*
2.5.4. Drivers para JDBC *18*
2.5.5. Arquitectura JDBC *18*

3. DISEÑO DEL SISTEMA 20
3.1. CARACTERÍSTICAS DEL SISTEMA 20
3.1.1. Tipos de usuarios *20*
3.1.2. Perfiles de acceso *21*
3.1.3. Organización de operadores del servicio *21*
3.2. FUNCIONALIDADES 21
3.2.1. Descripción de Funcionalidades *21*
3.2.2. División de funcionalidades *23*
3.2.3. Estadísticas *24*
3.3. ESTRUCTURA DE LA APLICACIÓN 25
3.3.1. Estado y Almacenamiento de información *27*

3.3.1.1. Diseño global 27
3.3.1.2. Estructura de la base de datos 28
3.3.1.3. Componentes de estado y Capa de acceso a datos 30
3.3.2. Gestión y Presentación *32*
3.4. TECNOLOGÍA Y VERSIONES EMPLEADAS 37

4. IMPLEMENTACIÓN 38
4.1. ARQUITECTURA DEL SISTEMA 38
4.1.1. Estado y Almacenamiento de información *38*
4.1.1.1. Contenido de la base de datos 38
4.1.1.2. Componentes de estado y capa de acceso a datos 45
4.1.2. Gestión y Presentación *57*
4.2. ESQUEMA DE ACCESOS EN FUNCIÓN DEL PERFIL 88

5. RESULTADO OBTENIDO 90
5.1. USUARIO DE TIPO OPERADOR 90
5.2. USUARIO DE TIPO ADMINISTRADOR 101
5.3. USUARIO DE TIPO ADMINISTRADOR - OPERADOR 122
5.4. TODO TIPO DE USUARIOS 124

6. PRUEBAS DEL SISTEMA 126
6.1. GESTIÓN DE FALLOS 127
6.2. AUTENTICACIÓN 127
6.3. REGISTRO 127
6.4. MANTENIMIENTO 129
6.5. ESTADÍSTICAS 130

7. CONCLUSIONES Y TRABAJOS FUTUROS 131
7.1. ARQUITECTURA DEL SISTEMA 131
7.2. DISEÑO DEL SISTEMA 133
7.3. PRESENTACIÓN 134
7.4. FUNCIONALIDAD 134
7.5. POLÍTICAS DE SEGURIDAD 135
7.6. TECNOLOGÍAS AVANZADAS 135

PRESUPUESTO 137
A. FASES DEL PROYECTO 137
B. ROLES ESTABLECIDOS 138
C. ASIGNACIÓN DE RESPONSABILIDADES 138
D. PLANIFICACIÓN 139
E. COSTES DIRECTOS 139
F. COSTES INDIRECTOS 141
G. COSTES TOTALES 141

REFERENCIAS 142

ÍNDICE DE FIGURAS

FIGURA DESCRIPCIÓN PÁGINA

Figura 2.1 El servidor web 6
Figura 2.2 El servidor de aplicaciones 6
Figura 2.3 La visión general 9
Figura 2.4 Interacción del servlet con el contenedor de servlets a través de la API Servlet 10
Figura 2.5 Modelo 1 de arquitectura JSP 13
Figura 2.6 El patrón Modelo-Vista-Controlador 15
Figura 2.7. Modelo 2 de arquitectura JSP 16
Figura 2.8 Driver Java puro 18
Figura 2.9 Driver ODBC y bibliotecas de cliente de la base de datos 19
Figura 3.1 Arquitectura del sistema 26
Figura 3.2 Estado, Acceso a datos y Base de datos 28
Figura 3.3 Esquema estado y datos del sistema 32
Figura 3.4 Gestión y Presentación. Autenticación – Selección de puesto 33
Figura 3.5 Gestión y Presentación. Mantenimiento 34
Figura 3.6 Gestión y Presentación. Estadísticas 35
Figura 3.7 Gestión y Presentación. Registro 36
Figura 4.1 Asociación de campos mediante claves externas 42
Figura 4.2 Esquema estado y datos del sistema. 47
Figura 4.3 Flujo de entrada y salida a login.jsp 58
Figura 4.4 Flujo de entrada y salida a ServletLogin.java 59
Figura 4.5 Flujo de entrada y salida a errorlogin.java 60
Figura 4.6 Flujo de entrada y salida a error.java 60
Figura 4.7 Flujo de entrada y salida a puestos.jsp 61
Figura 4.8 Flujo de entrada y salida a menuRegistro.jsp 62
Figura 4.9 Flujo de entrada y salida a registrosolo.jsp 63
Figura 4.10 Flujo de entrada y salida a registro.jsp 64
Figura 4.11 Flujo de entrada y salida a ServletRegistro.jsp 65
Figura 4.12 Flujo de entrada y salida a mantenimiento.jsp 66
Figura 4.13 Flujo de entrada y salida a alta_categoria.jsp 67
Figura 4.14 Flujo entre alta_categoria.jsp y ServletCategoria.java 68
Figura 4.15 Flujo de entrada y salida a baja_categoria.jsp 69
Figura 4.16 Flujo entre baja_categoria.jsp y ServletCategoria.java 70
Figura 4.17 Flujo de entrada y salida a ServletCategoria.java 70
Figura 4.18 Flujo de entrada y salida a alta_temas.jsp 72
Figura 4.19 Flujo entre alta_temas.jsp y ServletTema.java 73

Figura 4.20 Flujo de entrada y salida a baja_temas.jsp 74
Figura 4.21 Flujo entre baja_temas.jsp y ServletTema.java 75
Figura 4.22 Flujo de entrada y salida a ServletTema.java 75
Figura 4.23 Parámetros de entrada y salida a alta_operador.jsp 77
Figura 4.24 Flujo entre alta_operador.jsp y ServletOperador.java 78
Figura 4.25 Flujo de entrada y salida a baja_operador.jsp 79
Figura 4.26 Flujo entre baja_operador.jsp y ServletOperador.java 80
Figura 4.27 Flujo de entrada y salida a ServletOperador.java 80
Figura 4.28 Flujo de entrada y salida a estadisticas.jsp 82
Figura 4.29 Flujo de entrada y salida a listadoN.jsp 83
Figura 4.30 Flujo entre listadoN.jsp y ServletListado.java 84
Figura 4.31 Flujo entre listado1.jsp y ServletListado.java 85
Figura 4.32 Flujo de entrada y salida a ServletListado.java 86
Figura 4.33 Flujo de entrada y salida a ServletSession.java 88
Figura 5.1 Página de Autenticación 90
Figura 5.2 Mensaje de Error: Operador sin puesto asignado 91
Figura 5.3 Página de Selección de puesto 91
Figura 5.4 Página de Opción: Registro 92
Figura 5.5 Página de Registro 93
Figura 5.6 Mensaje de advertencia: Inicio de llamada 94
Figura 5.7 Mensaje de advertencia: Operación iniciada 94
Figura 5.8 Selección de Asunto 95
Figura 5.9 Selección de Categoría 95
Figura 5.10 Mensaje interrogatorio: Selección de Tema 96
Figura 5.11 Selección de Tema 96
Figura 5.12 Mensaje de confirmación: Selección de Tema 97
Figura 5.13 Selección realizada 97
Figura 5.14 Mensaje de advertencia: Formato incorrecto de NIF 98
Figura 5.15 Mensaje de advertencia: Formato incorrecto de correo electrónico 99
Figura 5.16 Mensaje de advertencia: Sugerencia no introducida 99
Figura 5.17 Mensaje de advertencia: Categoría no seleccionada 100
Figura 5.18 Mensaje informativo: Informe creado 100
Figura 5.19 Página de Autenticación 101
Figura 5.20 Página de Opciones de administración 102
Figura 5.21 Página de Opciones de mantenimiento 102
Figura 5.22 Página de Alta de categoría 103
Figura 5.23 Mensaje de advertencia: Categoría no especificada 103
Figura 5.24 Mensaje interrogatorio: Categoría existente 104
Figura 5.25 Mensaje informativo: Categoría añadida 104

Figura 5.26 Página de Opciones de Mantenimiento 105
Figura 5.27 Página de Baja de categoría 105
Figura 5.28 Mensaje interrogatorio: Confirmación de baja de categoría 106
Figura 5.29 Mensaje informativo: Categoría eliminada 106
Figura 5.30 Página de "Opciones de Mantenimiento" 107
Figura 5.31 Página de "Alta de tema" 107
Figura 5.32 Mensaje interrogatorio: Tema existente 108
Figura 5.33 Mensaje informativo: Tema añadido 108
Figura 5.34 Página de "Opciones de Mantenimiento" 109
Figura 5.35 Página de "Baja de tema" 109
Figura 5.36 Mensaje interrogatorio: Confirmación de baja de tema 110
Figura 5.37 Mensaje informativo: Tema eliminado 110
Figura 5.38 Página de "Opciones de Mantenimiento" 111
Figura 5.39 Página de "Alta de operador" 111
Figura 5.40 Mensaje de advertencia: Puesto no asignado 112
Figura 5.41 Mensaje informativo: Operador añadido 113
Figura 5.42 Página de "Baja de operador" 113
Figura 5.43 Página de "Baja de operador" 114
Figura 5.44 Mensaje informativo: Operador eliminado 114
Figura 5.45 Página de Estadísticas 115
Figura 5.46 Formulario de cálculo estadístico 115
Figura 5.47 Resultado Estadística 1: Asuntos 116
Figura 5.48 Resultado Estadística 1: Categorías 117
Figura 5.49 Resultado Estadística 1: Temas 117
Figura 5.50 Resultado Estadística 2 118
Figura 5.51 Resultado Estadística 3 119
Figura 5.52 Resultado Estadística 4 120
Figura 5.53 Resultado Estadística 5 121
Figura 5.54 Resultado Estadística 6 122
Figura 5.55 Página de Autenticación 122
Figura 5.56 Página de "Selección de puesto" 123
Figura 5.57 Página de Opciones de administración y registro 123
Figura 5.58 Página de Error de Autenticación 124
Figura 5.59 Página de Error: Sesión expirada 125
Figura 5.60 Página de Error: Acceso a Base de Datos 125
Figura 7.1. Modelo 2 de arquitectura JSP 131
Figura 7.2. Arquitectura del sistema 131
Figura P.1 Diagrama de Gantt del proyecto 139

ÍNDICE DE TABLAS

TABLA DESCRIPCIÓN PÁGINA

Tabla 2.1 Métodos de la interfaz javax.servlet.Servlet 10
Tabla 3.1 Asociación: Perfil – Tipo de usuario 29
Tabla 4.1 Elementos de la tabla perfil. 39
Tabla 4.2 Elementos de la tabla usuario. 39
Tabla 4.3 Elementos de la tabla usuario-puesto. 40
Tabla 4.4 Elementos de la tabla puesto. 40
Tabla 4.5 Elementos de la tabla grupo. 40
Tabla 4.6 Elementos de la tabla oficina. 41
Tabla 4.7 Elementos de la tabla entidad. 41
Tabla 4.8 Elementos de la tabla asunto. 42
Tabla 4.9. Elementos de la tabla categoria. 42
Tabla 4.10 Elementos de la tabla tema. 43
Tabla 4.11. Elementos de la tabla informe. 44
Tabla 4.12 Elementos de la tabla consulta. 45
Tabla 4.13 Representación: Componente de Estado – Tablas en Base de datos 45
Tabla 4.14 Equivalencia: Componente de Estado – Clase DAO 46
Tabla 4.15 Equivalencia: Método de acceso – Sentencia preparada 51
Tabla 4.16 Equivalencia: Método de acceso – Método de preparación de la petición 51
Tabla 4.17 Equivalencia: Método de acceso – Sentencia preparada completa 52
Tabla 4.18 Equivalencia: Método de búsqueda – Método de carga 52
Tabla 4.19 Métodos de acceso a base de datos utilizados por cada componente de estado 55
Tabla 4.20 Equivalencia Página JSP – Estadística relacionada 83
Tabla 6.1 Batería de pruebas – Gestión de fallos 127
Tabla 6.2 Batería de pruebas – Autenticación 127
Tabla 6.3 Batería de pruebas – Registro 128
Tabla 6.4 Batería de pruebas – Mantenimiento 129
Tabla 6.5 Batería de pruebas – Estadísticas 130
Tabla P.1 Responsabilidad y dedicación 138
Tabla P.2 Coste de los recursos 139
Tabla P.3 Coste de los equipos 140
Tabla P.4 Coste del material 140

1. INTRODUCCIÓN

1.1. MOTIVACIÓN

La aplicación sobre la que trata este documento está llevada a cabo con la finalidad de ser implantada por el Gobierno de Cantabria como herramienta de trabajo para los empleados del Servicio de Información Administrativa proporcionado por esta Comunidad Autónoma.

El Gobierno de Cantabria ofrece información actualizada a los ciudadanos a través de diversos canales de distribución informativa como Twitter [66], Facebook [19], Linkedin [46] o Youtube [71], además de hacerlo a través del Portal Institucional del Gobierno de Cantabria [7] y de la Oficina de Atención a la Ciudadanía, por medio del teléfono 012.

El Servicio de Información Administrativa forma parte de esta Oficina de Atención a la Ciudadanía. Es un servicio telefónico de atención al ciudadano, cuyo cometido es resolver las dudas sobre diferentes aspectos de la Administración de la Comunidad Autónoma de Cantabria.

Cada Comunidad Autónoma en España ofrece este servicio de información al ciudadano a través del teléfono 012 si la llamada se efectúa desde dicha región.

En el caso de llamar desde fuera del ámbito territorial correspondiente a la Comunidad Autónoma en cuestión, se proporciona un número de teléfono para cada una de ellas, siendo el número 902139012 el asignado a Cantabria.

El coste del servicio es independiente de la duración de la comunicación, siendo de 0,3426 euros por llamada.

Este servicio está disponible en el siguiente horario:

- De lunes a viernes no festivos: de 9:00 a 21:00 horas.
- Sábados no festivos: de 9:00 a 14:00 horas

En caso de que un usuario realice la llamada fuera de este rango horario, la atención telefónica no se podrá realizar a través de un agente informador en ese momento, sino que se grabará la locución en un buzón de voz. Posteriormente, en horario disponible del servicio, se devolverá la llamada al ciudadano desde el Teléfono de Información 012 para responder a su solicitud.

A través del Servicio de Información Administrativa, un equipo de operadores telefónicos especialmente formados atenderá las demandas de información formuladas por los ciudadanos, respondiendo en el momento si es posible o anotando los datos necesarios para contactar con el ciudadano en un momento posterior.

Para poder llevar a cabo esta tarea y suministrar a los usuarios la información solicitada es necesario proporcionar a los operadores la documentación pertinente, de manera que puedan consultarla y responder adecuadamente a la solicitud del ciudadano que realiza la llamada.

Además de esta necesidad básica, surgen algunas ideas para mejorar la calidad del servicio.

La primera consiste en adaptarse a las necesidades de información de los ciudadanos. Para detectar estas preferencias en la demanda de tipos de información, se plantea monitorizar la frecuencia con la que se consulta cada sección disponible. Esto proporciona los datos necesarios para que por parte de la administración del servicio se puedan realizar funciones de mantenimiento, añadiendo nuevas secciones de información, ampliando aquellos apartados que tengan más consultas, conservando sus contenidos siempre actualizados, y reduciendo o eliminando aquellos menos consultados u obsoletos.

También se pretende evaluar la calidad del servicio prestado, considerando el tiempo que se dedica a resolver las dudas de los ciudadanos en función de diferentes parámetros que puedan resultar de utilidad a la hora de realizar cambios administrativos.

La consultora Vass se encarga de cubrir estas necesidades llevando a cabo este proyecto para el Gobierno de Cantabria. Se decide incluir este desarrollo en el listado de Proyectos Fin de Carrera a desarrollar en la empresa, al coincidir la demanda de la aplicación con la incorporación en la empresa de un puesto de becario y resultando adecuado el tiempo de duración de una beca en relación al proyecto a desarrollar.

El proyecto se ha llevado a cabo por un equipo de tres personas, dentro del departamento de desarrollo. Los perfiles de los otros dos integrantes del equipo son el de un consultor senior, encargado de la gestión del proyecto y de la labor comercial y trato con el cliente, y un analista programador, responsable del proyecto.

1.2. OBJETIVOS

El objetivo del Proyecto Fin de Carrera es el diseño, desarrollo y pruebas de una herramienta web para el soporte del Servicio de Información Administrativa de Cantabria.

La aplicación solicitada por el Gobierno de Cantabria debe proporcionar, en primer lugar, un repositorio de información adecuadamente ordenado y completo, disponible para todos los operadores telefónicos usuarios de la herramienta.

La información referente al Servicio de Información Administrativa se debe presentar al operador estructurada en una división por categorías y subdivisiones dentro de éstas, facilitando el acceso al contenido específico a la hora de realizar búsquedas de información.

También se espera de la aplicación que aporte la capacidad de recopilar de manera organizada la información relativa a las consultas de los ciudadanos que realizan una llamada al servicio. Los responsables de recoger esta información e introducirla en la herramienta son los operadores telefónicos, que además de consultar la información solicitada, deben realizar este registro para cada una de las llamadas que asisten.

Una tercera funcionalidad de la aplicación consiste en tratar los datos recogidos por los operadores, de forma que se puedan extraer conclusiones acerca del tipo de información que se solicita y sobre la eficiencia del servicio.

Esta funcionalidad se ofrece a los usuarios de la aplicación con perfil de administrador del servicio. Podrán analizar las secciones de información más consultadas, y estadísticas sobre el tiempo dedicado a estas consultas.

Las estadísticas se pueden realizar teniendo en cuenta tres aspectos diferentes, que se tendrán en cuenta para realizar cada clasificación.

El primero de los aspectos a considerar hace referencia a los temas consultados, calculándose el tiempo medio dedicado a informar sobre cada tema. Esto aporta a la clasificación por temas más consultados una visión que incluye un nuevo parámetro, el tiempo dedicado a cada uno de ellos.

La segunda clasificación se centra en la diferenciación entre operadores telefónicos, pudiendo consultarse el tiempo medio que dedica cada uno de ellos a procesar las consultas recibidas. De esta manera se puede llevar un control sobre si la carga de trabajo entre los operadores está equilibrada.

El último aspecto que se debe considerar está relacionado con el estudio entre los diferentes momentos en los que se realizan las consultas. Se podrá obtener información del tiempo medio dedicado a las consultas en tres escenarios: cada día concreto en un periodo de tiempo determinado, por franjas horarias y por días de la semana.

Los usuarios administradores, con la información proporcionada por las estadísticas, podrán tomar decisiones acerca de las modificaciones a realizar sobre la información que se consulta por los operadores, añadiendo o eliminando secciones en función de los resultados obtenidos tras el estudio de la información recogida en consultas previas.

Además de gestionar las secciones de información, el usuario administrador también podrá gestionar las altas y bajas de operadores telefónicos del servicio.

Por tanto, la aplicación debe realizar una diferenciación entre los tipos de usuarios del servicio, y proporcionar a cada uno de ellos las funciones descritas.

1.3. ESTRUCTURA DEL DOCUMENTO

El documento de la memoria se ha estructurado en varios apartados, que se describen brevemente a continuación.

El primero de ellos es el estado del arte, apartado en el que se realiza una investigación de carácter bibliográfico acerca de las tecnologías empleadas en el campo del desarrollo de aplicaciones web utilizando J2EE.

Se ha comenzado con este apartado, que permite recabar información sobre el conocimiento adquirido previamente en la materia, identificar las tecnologías que pueden emplearse en el desarrollo de este tipo de aplicaciones y decidir qué metodologías conviene utilizar en el caso que nos ocupa.

A este primer apartado le sigue el diseño del sistema, en el que se tienen en cuenta diferentes aspectos relacionados con el diseño de la aplicación. Se realiza una descripción detallada de cada uno de estos aspectos, y se analiza y decide la estructura a la que se va a ajustar el sistema.

En primer lugar se describen las características que va a tener el sistema. Considerando los posibles usuarios de la herramienta, se detallan sus particularidades y se definen los perfiles de acceso al sistema. También se describe la jerarquía organizativa del servicio, indicando las posibles localizaciones de los usuarios dentro de la organización.

A continuación se indican las operaciones que ofrece la herramienta para cada tipo de usuario, y se definen las funcionalidades, detallando las posibilidades que ofrece cada una de ellas a los usuarios del sistema. Es conveniente entrar en detalle sobre una de estas funcionalidades, la generación de estadísticas, indicando qué cálculos se pueden realizar y cómo se determina cada uno de ellos.

Una vez se han definido las características y funcionalidades que ofrece el sistema, se detalla el diseño de la estructura de la aplicación, indicando los módulos de los que va a constar y el patrón de diseño que se ha decidido implementar.

Se diseña cada uno de estos módulos y las relaciones entre ellos, definiendo por un lado el estado de la aplicación, el almacenamiento de información en base de datos y el acceso entre estas entidades, y por otro lado los módulos de presentación y gestión del sistema, cuyo diseño se ha realizado de manera conjunta, realizando una división por bloques, que representan cada fase de la aplicación.

Tras el diseño, se describen los detalles de la implementación de todos los componentes de la arquitectura del sistema.

Al igual que se ha estructurado en la fase de diseño del sistema, se detalla en primer lugar la implementación de los módulos de estado y base de datos de la aplicación, y la interacción entre ambos, y a continuación la implementación de los módulos de presentación y gestión.

En cuanto a la base de datos, se detalla el contenido de las tablas y la finalidad de cada uno de los campos creados y sobre el módulo de estado se indican los componentes, y la equivalencia con las tablas de la base de datos.

En cuanto al bloque de acceso a datos, se describe la implementación de los mecanismos que permiten el acceso y la traducción de contenidos entre los objetos de la aplicación y los elementos de la base de datos, indicando los conjuntos de métodos utilizados para cada tarea y detallando el procedimiento para el acceso a la base de datos, especificando los elementos involucrados.

En la descripción de la implementación de los bloques de gestión y presentación se ha detallado cada elemento de estos módulos, estructurados en las fases de la aplicación definidas en el diseño.

Finalmente, una vez descritos los elementos que componen estos bloques, se detalla con carácter aclaratorio el esquema de accesos a las diferentes páginas en función del perfil del usuario.

Tras detallar la implementación del sistema, se muestra el resultado obtenido, y las posibles operaciones que los diferentes usuarios pueden realizar con la herramienta desarrollada.

También se detalla el proceso de pruebas realizado para la validación del sistema, indicando, aparte de las pruebas llevadas a cabo durante el desarrollo del sistema, la batería de pruebas pasada una vez finalizada la implementación.

Por último, se presentan el Presupuesto e Historia del Proyecto Fin de Carrera y las Conclusiones y trabajos futuros.

2. ESTADO DEL ARTE

En este apartado se describen las tecnologías utilizadas para el desarrollo de la herramienta que constituye este proyecto.

Se trata de una aplicación web que se ha diseñado e implementado utilizando un modelo específico de arquitectura basado en páginas JSP y servlets, tecnologías que se describen brevemente en esta sección.

2.1. APLICACIONES WEB

2.1.1. Definición de aplicación web

Se denomina aplicaciones web [56] a aquellas aplicaciones que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador. Un ejemplo común de aplicación web es un sitio web que proporciona un servicio gratuito de correo electrónico. Ofrece todas las características de un cliente de correo electrónico como podía ser Outlook Express [53] pero está basado en la web en su totalidad. Un beneficio clave de las aplicaciones web consiste en la facilidad con la que los usuarios pueden acceder a ellas. Todo lo que un usuario necesita es un navegador web, no requiere de ninguna instalación adicional en la máquina del usuario, lo que incrementa enormemente el alcance de este tipo de aplicaciones.

Una aplicación web está formada por diversos componentes; cada uno de ellos desempeña una tarea específica y tiene la capacidad de exponer sus servicios a través de la web. Todos estos componentes se coordinan entre sí para proporcionar un conjunto completo de servicios a los usuarios.

Los recursos utilizados por una aplicación web se pueden clasificar en activos y pasivos. Los recursos pasivos o estáticos no incluyen en sí mismos un procesamiento; cuando uno de estos recursos se solicita al servidor web, éste lo recupera y lo devuelve al navegador que hizo la petición. Un ejemplo de recurso pasivo puede ser uno de los archivos htlm contenido en el servidor. Los recursos activos incorporan su propia capacidad de procesamiento, de manera que cuando se solicita un recurso activo, éste ejecuta su código y el resultado obtenido es enviado por el servidor web al navegador. Los servlets y las páginas JSP son ejemplos de recursos activos, también conocidos como componentes web.

Generalmente, las aplicaciones web contienen un conjunto de recursos entre los que se encuentran recursos activos y pasivos, y son los recursos activos los que proporcionan a los usuarios la posibilidad de ejecución de una lógica de negocio, que aporta el dinamismo requerido por este tipo de aplicaciones.

2.1.2. Servidor web, contenedor web y servidor de aplicaciones

En este apartado se definen los conceptos de servidor web, contenedor web y servidor de aplicaciones, indicando la relación entre dichos elementos y las diferencias existentes entre ellos [73, 15].

Una aplicación web reside en un servidor web. El servidor web proporciona a la aplicación web el acceso fácil y controlado a los recursos del sistema, y también provee de servicios de bajo nivel, como la implementación del protocolo HTTP o la gestión de la conexión con la base de datos.

El protocolo HTTP [25], o Hypertext Transfer Protocol es un protocolo sin estado basado en un modelo de petición-respuesta. Un cliente envía una petición HTTP de un recurso y el servidor responde con una respuesta HTTP con el recurso deseado.

La mayoría de los clientes web utilizan el protocolo HTTP para comunicarse con un servidor web. HTTP define las peticiones que un cliente puede enviar a un servidor, y las respuestas que el servidor puede enviar como contestación. En el caso de Internet, el navegador web es el cliente HTTP, el servidor web es el servidor HTTP, y los recursos deseados son ficheros HTML [65], servlets, archivos de imágenes, u otros archivos. Cada recurso es identificado por un único identificador URI, o *Uniform Resource Identifier* [68].

Un servidor web es capaz de recibir peticiones HTTP e interpretarlas, y procesar respuestas HTTP y enviarlas al cliente correspondiente, en este caso, el navegador web.

El servidor web convierte la petición HTTP en un objeto de petición HTTP y lo entrega al componente web identificado por el URI de la petición. El componente web rellena un objeto de respuesta HTTP, que el servidor convierte en una respuesta HTTP y envía al cliente.

Un ejemplo de servidor web es Apache Web Server [2].

El contenedor web forma parte del servidor web, y procura el escenario adecuado para que los componentes web se ejecuten, proporcionando la implementación de J2EE que sirve de motor a los componentes web, típicamente servlets y páginas JSP. En la *Figura 2.1* se representa el esquema de un servidor web.

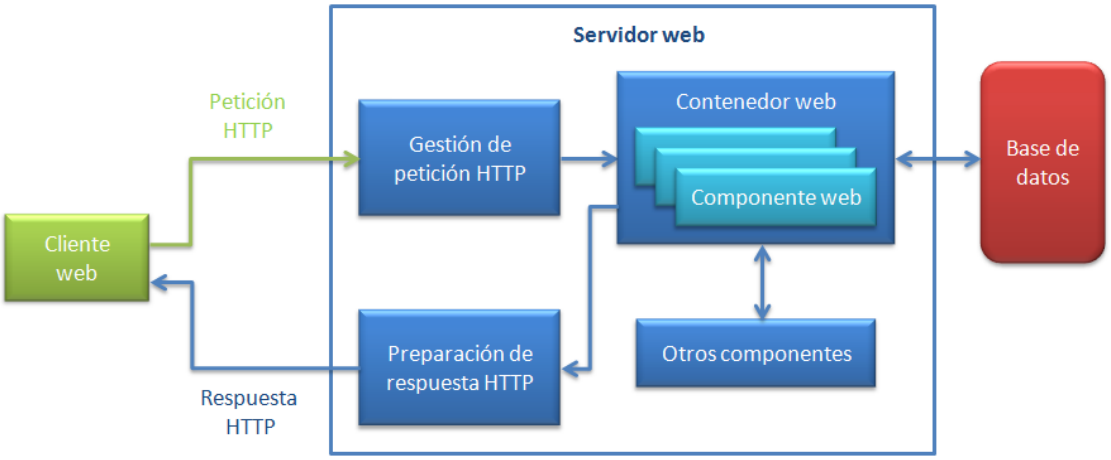


Figura 2.1 El servidor web

Si una petición HTTP está dirigida a un componente web, el servidor web la reenvía al contenedor web, que entrega el resultado de la petición de nuevo al servidor web. Éste genera la respuesta HTTP a partir del resultado recibido del contenedor web y la envía al cliente, como respuesta a la petición HTTP recibida.

Un ejemplo de contenedor web es Tomcat [3]. El escenario típico completo sería en este caso el servidor web Apache HTTP Server como servidor web, y Tomcat como contenedor web.

El servidor de aplicaciones, cuyo esquema se ilustra en la *Figura 2.2*, es un servidor completo, que proporciona el escenario para la ejecución de los componentes de negocio, como pueden ser los Enterprise JavaBeans [18], además de poseer de las capacidades de un servidor web y un contenedor web.

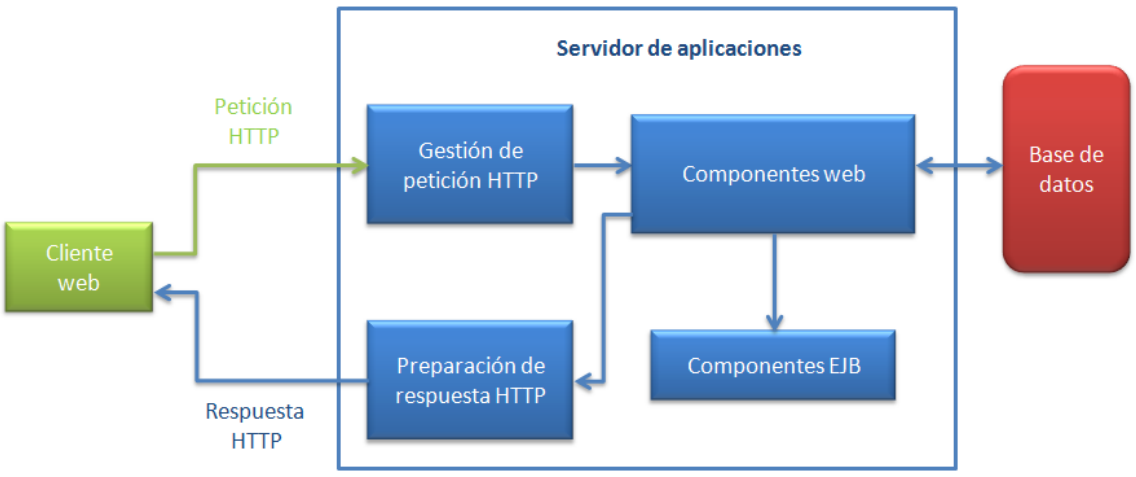


Figura 2.2 El servidor de aplicaciones

Algunos ejemplos de servidores de aplicaciones pueden ser *WebSphere* de *IBM* [36] o *Java System Application Server* de *Sun* [61].

2.1.3. Módulos web en J2EE

Como se ha detallado en la definición de una aplicación web, en la arquitectura J2EE los recursos web están formados por el conjunto de los archivos estáticos de contenido web y los componentes web. Un módulo web es la menor unidad desplegable y utilizable de recursos web.

Además de los componentes web y los recursos estáticos, un módulo web puede contener otros archivos. En el lado del cliente puede contener *applets* y clases de utilidad, y en el lado del servidor puede también contener clases de utilidad, como son los *beans* de la base de datos.

Los módulos web tienen una estructura específica. El directorio de más alto nivel de un módulo web es el directorio raíz de la aplicación. En él se almacenan las páginas JSP, las clases y archivos del lado del cliente, y los recursos web estáticos. También es posible crear subdirectorios específicos de cada aplicación en el directorio raíz y en el directorio de clases.

Un módulo web puede desplegarse como una estructura de ficheros sin empaquetar, o puede ser empaquetado en un fichero comprimido, como se indica a continuación.

El formato de archivo JAR (Java ARchive) es un formato desarrollado por Sun [63, 37], que permite agrupar el conjunto de clases escritas en lenguaje Java, que constituyen una aplicación, comprimiéndolas en un solo archivo.

El archivo WAR (Web Application aRchive) [72, 62] o archivo de aplicación web es un archivo de tipo JAR utilizado para empaquetar los elementos necesarios para la instalación y ejecución de una aplicación web Java. Estos elementos están formados por una colección de páginas JSP, servlets, clases Java, archivos XML, librerías de tags y páginas web estáticas (HTML y archivos relacionados) que constituyen la aplicación web.

De esta forma, el archivo WAR es utilizado para distribuir el conjunto de elementos que forman una aplicación web, pudiendo ser ejecutado en los distintos motores de servlets, o servidores J2EE, de forma sencilla.

2.1.4. Descriptor de despliegue

Para definir una aplicación web se utiliza un descriptor de despliegue [56]. Se trata de un documento XML con el nombre *web.xml*, que contiene la descripción de todos los componentes dinámicos de la aplicación web.

El descriptor de despliegue contiene una entrada por cada servlet utilizado en la aplicación web, y también permite declarar otro tipo de elementos, como filtros, controladores de error, o establecer determinados parámetros de configuración de seguridad para elementos de la aplicación [67].

El servidor de aplicaciones utiliza el descriptor de despliegue para inicializar los componentes de la aplicación web, y para hacer que estén disponibles para los clientes.

2.1.5. Ciclo de vida de una aplicación web

Una aplicación web está compuesta por componentes web, archivos de recursos estáticos, y clases y librerías de ayuda. El contenedor web proporciona servicios de soporte que incrementan la capacidad de los componentes web y facilitan su desarrollo. La aplicación web debe tener en cuenta estos servicios, y esto hace que el proceso de creación y ejecución de una aplicación web difiera de las tradicionales clases Java autónomas.

El proceso de creación, despliegue y ejecución de una aplicación web sigue los pasos que se enumeran a continuación:

- Desarrollo del código del componente web.
- Desarrollo del descriptor de despliegue de la aplicación web.

- Compilación de los componentes de la aplicación web y las clases de ayuda referenciadas por dichos componentes.
- Empaquetado de la aplicación en una unidad desplegable, de manera opcional.
- Despliegue de la aplicación en un contenedor web.
- Acceso a la URL que referencia a la aplicación web.

2.2. JAVA SERVLETS

Un servlet es una entidad perteneciente al lado del servidor. Resulta interesante conocer las funciones que desempeña un servidor para comprender las razones de la creación de este concepto.

2.2.1. Funciones de un servidor

Un servidor que proporciona servicios a clientes remotos tiene principalmente dos responsabilidades.

La primera de ellas consiste en manejar las peticiones del cliente. Esta tarea conlleva programar a nivel de socket, extraer información de los mensajes de petición recibidos y la implementación de protocolos como FTP [64] o HTTP.

La segunda tarea, la generación una respuesta para ser devuelta al cliente, varía de un servicio a otro. En el caso de servidores FTP que sirven peticiones de transferencias de ficheros, la creación de la respuesta es tan simple como encontrar un fichero en la máquina local. En cambio, servidores HTTP que albergan aplicaciones web complejas necesitan crear la respuesta dinámicamente, lo que puede suponer llevar a cabo tareas complicadas, como recuperar datos de la base de datos, aplicar normas de la lógica de negocio, o presentar la salida en el formato deseado por cada cliente.

2.2.2. Implementación de un servidor

2.2.2.1. *Primera aproximación*

Una forma de escribir un servidor sencillo, que sirviera solamente datos estáticos, sería codificarlo en un solo programa ejecutable. Este programa se encargaría de todas las tareas necesarias, como la gestión de redes, implementación de protocolos, localización de datos, y generación de respuestas.

Sin embargo, con servidores que sirvan datos distribuidos, se requiere un diseño más completo y que proporcione mayor flexibilidad. En casos en los que la lógica de la aplicación esté en continua evolución, los clientes necesiten soluciones personalizadas o los socios de negocio precisen de reglas de procesamiento que se adapten a sus especificaciones, resulta imposible escribir un solo programa que lleve a cabo todas estas tareas. Incluso en el caso de que fuera posible, para cualquier modificación como podría ser cambiar el formato de los datos o añadir una nueva funcionalidad, sería necesario modificar el código fuente, lo que resulta indeseable en cualquier aplicación.

2.2.2.2. *Extensiones del servidor*

Una solución consiste en dividir el servidor en módulos ejecutables independientes, que puedan cargarse en la memoria y ser inicializados una sola vez, en el arranque del servidor. Cada petición podrá ser servida por la copia de los módulos, en memoria y lista para servir peticiones.

Estos módulos ejecutables independientes se conocen como *extensiones del servidor.* En plataformas diferentes de Java, las extensiones del servidor se escriben utilizando las APIs (Application Programming Interfaces) proporcionadas por los suministradores del servidor. Por ejemplo, un Netscape Server proporciona la Netscape Server Application Programming Interface, NSAPI [50].

En Java, las extensiones del servidor se escriben utilizando la API Servlet [58], y los módulos de extensión del servidor se llaman *servlets*.

Cada vez que sea necesario añadir una funcionalidad al servidor, todo lo que hay que hacer es escribir un nuevo servlet específico para ese conjunto de requerimientos e incorporarlo al servidor, sin modificar el propio servidor.

Ahora el servidor principal solo necesita preocuparse de las comunicaciones y conexiones de red. La tarea de interpretar peticiones y crear las respuestas apropiadas es delegada a los servlets.

2.2.3. Contenedor de servlets

Los servidores web utilizan un módulo separado donde se cargan y ejecutan los servlets. Este módulo especializado, dedicado a la gestión de los servlets, se llama contenedor de servlets o motor de servlets [51].

2.2.3.1. *Esquema general*

La *Figura 2.3* muestra cómo los diferentes componentes de una aplicación encajan en el esquema general. Los archivos HTML se almacenan en el sistema de ficheros, los servlets se ejecutan en el contenedor de servlets, y los datos de la aplicación se almacenan en la base de datos.

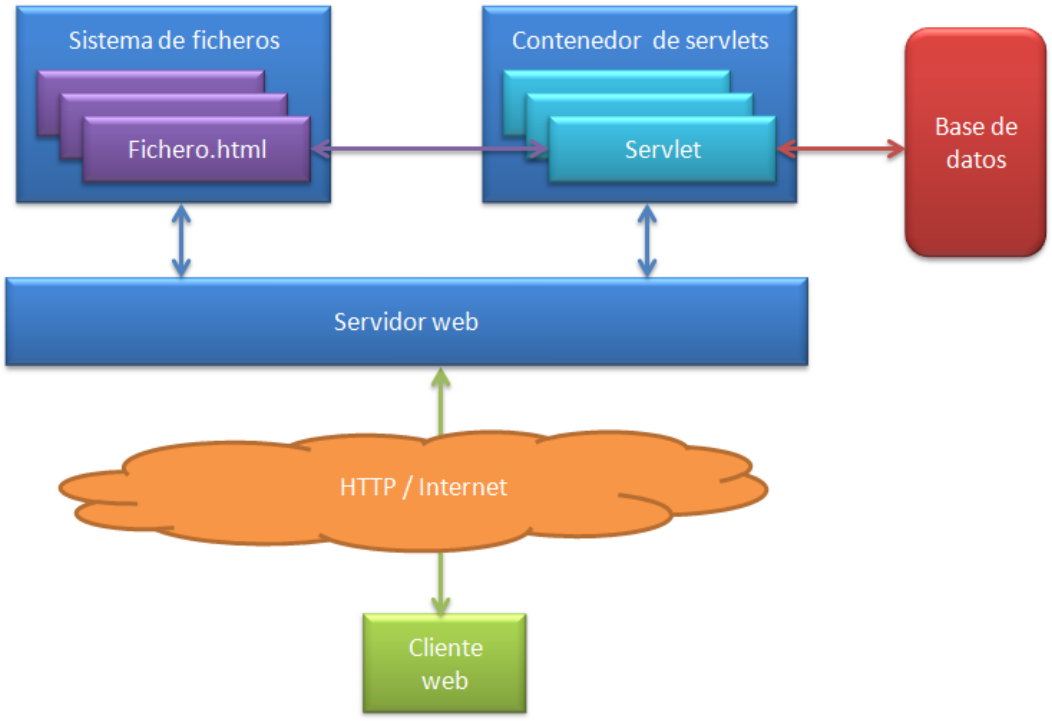


Figura 2.3 La visión general

El navegador envía peticiones al servidor web. Si el objetivo es un recurso estático, como puede ser un archivo HTML, el servidor lo procesa directamente. En el caso de que el objetivo sea un servlet, el servidor delega la petición al contenedor de servlets, que a su vez la reenvía al servlet. El servlet puede utilizar los recursos estáticos y la base de datos, generando una salida dinámica.

2.2.4. La API Servlet

La especificación Servlet de Sun proporciona un estándar y un entorno de desarrollo independiente de la plataforma para la comunicación entre los servlets y sus contenedores. Este entorno de desarrollo está formado por un conjunto de interfaces y de clases de Java, y este conjunto es denominado Servlet Application Programming Intefaces, o API Servlet [58]. Los servlets se desarrollan utilizando esta API, que es implementado por el contenedor de servlets, tal y como se muestra en la *Figura 2.4*.

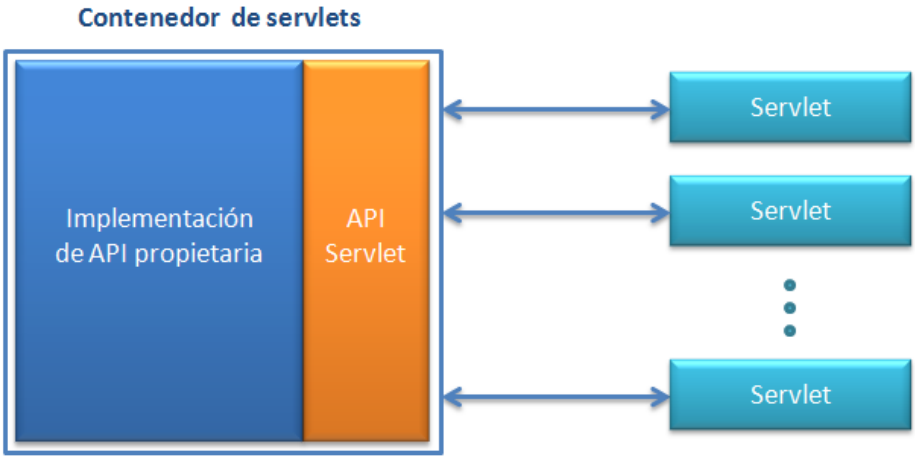


Figura 2.4 Interacción del servlet con el contenedor de servlets a través de la API Servlet

La API Servlet es todo lo que los desarrolladores necesitan conocer. Basta con comprender sus reglas y la funcionalidad que proporciona para programar un servlet.

Es posible ejecutar un servlet en cualquier contenedor de servlets, solamente se necesita que soporte la API Servlet. Puesto que todo contenedor de servlets debe proporcionar esta API, los servlets son independientes tanto de la plataforma como del contenedor de servlets.

El API Servlet se divide en dos paquetes: *javax.servlet* y *javax.servlet.http*. A continuación se exponen brevemente sus principales características.

2.2.4.1. *El paquete* javax.servlet

Este paquete contiene las interfaces y clases genéricas de un servlet, que son independientes de cualquier protocolo.

La interfaz central de la API Servlet es *Servlet* [31] y está contenida en este paquete. Cada clase servlet debe, directa o indirectamente, implementar esta interfaz.

Contiene cinco métodos, indicados en la *Tabla 2.1*.

MÉTODO	DESCRIPCIÓN
init()	Método llamado por el contenedor de servlets para indicar al servlet que debe inicializarse y prepararse para dar servicio. El contenedor pasa por parámetro un objeto de la clase ServletConfig.
service()	Método llamado por el contenedor de servlets por cada petición del cliente destinada a ese servlet, para permitir al servlet manejar la petición y generar una respuesta.
destroy()	Método llamado por el contenedor de servlets para indicar al servlet que debe borrarse, liberar los recursos requeridos, y prepararse para dejar de dar servicio.
getServletConfig()	Devuelve información sobre el servlet, como parámetro para el método init()
getServletInfo()	La clase de implementación debe devolver información sobre el servlet, como el autor, la versión, e información de copyright.

Tabla 2.1 Métodos de la interfaz javax.servlet.Servlet

La clase *GenericServlet* [8] implementa la interfaz *Servlet*. Es una clase abstracta que proporciona implementación para todos los métodos excepto el método *service()* de la interfaz *Servlet*. También añade algunos métodos para permitir la autenticación. Es posible heredar esta clase e implementar el método *service()* para escribir cualquier tipo de servlet.

Otras interfaces que intervienen en la utilización de los servlets y pertenecen a este paquete son:

ServletRequest

La interfaz *ServletRequest* [33] proporciona una visión genérica de la petición enviada por el cliente y define los métodos que extraen información de dicha petición.

ServletResponse

La interfaz *ServletResponse* [34] proporciona una manera genérica de enviar respuestas, definiendo los métodos necesarios para enviar una respuesta apropiada al cliente.

ServletContext

La interfaz *ServletContext* [32] define un conjunto de métodos que permiten al servlet comunicarse con el contenedor de servlets e intercambiar información, teniendo en cuenta que existe un contexto por cada aplicación web.

RequestDispatcher

La interfaz *RequestDispatcher* [30] define un objeto que recibe peticiones del cliente y las reenvía a un recurso del servidor, como un servlet, una página JSP, o un archivo HTML. El contenedor de servlets crea el objeto *RequestDispatcher*, que entrega la petición al recurso indicado, identificado por su nombre o su localización.

2.2.4.2. *El paquete* javax.servlet.http

Este paquete proporciona la funcionalidad básica requerida para un servlet que reciba peticiones HTTP. Las interfaces y clases de este paquete heredan de las correspondientes interfaces y clases del paquete *javax.servlet*, además de proporcionar soporte para el protocolo HTTP.

La clase *HttpServlet* [27] es una clase abstracta que hereda de *GenericServlet* y las interfaces *HttpServletRequest* [28] y *HttpServletResponse* [29] heredan respectivamente de *ServletRequest* y *ServletResponse, y* proporcionan una particularización de la petición y la respuesta específicas para HTTP.

La interfaz *HttpServletRequest* define métodos que extraen información de la petición, y la interfaz *HttpServletResponse* define métodos necesarios para la colocación de información en la respuesta.

2.3. JAVASERVER PAGES

2.3.1. Concepto de página JSP

Antes de abordar la definición de una página JSP, es interesante introducir la necesidad de dicha tecnología.

2.3.1.1. *HTML*

HTML, o HyperText Markup Language [65], es un lenguaje de marcado que especifica cómo etiquetar las porciones de datos para su presentación visual. Los hipervínculos permiten saltar de una porción de información a otra. Sin embargo, su contenido está ya dentro de las etiquetas HTML. Las etiquetas no lo crean, sino que meramente lo decoran para su presentación.

HTML por sí mismo produce páginas web estáticas. Pero hoy en día, es necesario para la mayor parte de los sitios web tener contenido dinámico. Para generar el contenido dinámicamente, se necesita especificar una lógica de negocio, y poder generar datos en respuesta a una petición. Los datos pueden ser luego formateados utilizando HTML.

2.3.1.2.*Server-Side Includes*

Una página web dinámica está compuesta por código de lenguaje de etiquetado y también por código de lenguaje de programación. En lugar de servir la página como es a los clientes, un servidor procesa el código de lenguaje de programación, reemplaza el código por los datos generados por éste, y luego envía la página al cliente.

Esta metodología de incluir lenguajes de programación en el código HTML se denomina *server-side include*, y el lenguaje de programación que es incluido en el código HTML se denomina *lenguaje de scripting*. Por ejemplo, Server-Side JavaScript (SSJS) de Netscape [57] y Active Server Pages (ASP) de Microsoft [1] son ejemplos de server-side includes. Utilizan JavaScript [39] y VBScript [70], respectivamente, como lenguajes de scripting.

JavaServer Pages es el nombre de la tecnología que proporciona una especificación estándar para combinar Java como lenguaje de scripting con HTML y conforma la capa de presentación de la arquitectura de Sun Java 2 Enterprise Edition (J2EE) [56].

2.3.1.3.*Definición de página JSP*

Por tanto, una página JSP (JavaServer Pages) es una página web que contiene código Java junto con las etiquetas HTML. Al igual que cualquier otra página web, una página JSP tiene una única URL, utilizada por los clientes para acceder a la página. Cuando es accedido por un cliente, el código Java de la página es ejecutado en el lado del servidor, produciendo datos textuales. Estos datos, encapsulados entre etiquetas HTML, son enviados al cliente como una página HTML normal.

Como el código Java de una página JSP se procesa en el lado del servidor, el cliente no tiene conocimiento del código. El código es reemplazado por el HTML generado por el código Java antes de que la página se envíe al cliente.

La especificación JSP lista la sintaxis y describe la semántica del conjunto de elementos que componen una página JSP. Estos elementos se denominan etiquetas JSP. Por tanto, una página JSP es una plantilla HTML compuesta por etiquetas JSP activas y etiquetas HTML pasivas entremezcladas. En tiempo de ejecución, la plantilla es utilizada para generar una página puramente HTML, que es enviada al cliente.

2.3.2. Uso de servlet o JSP

Si los servlets pueden hacer lo que hacen las páginas JSP, y viceversa, es necesario conocer la diferencia entre ambos.

Como se explica anteriormente en la sección dedicada a los servlets, se trata de extensiones del servidor y su función consiste en proporcionar una funcionalidad extra al servidor principal. Esto podría incluir la implementación de servicios especializados como autenticación, autorización, validación de bases de datos o gestión de transacciones. Los servlets actúan como componentes controladores que gestionan la lógica de negocio.

Por otro lado, las páginas JSP son páginas web. Son similares en estructura a las páginas HTML en cuestión de diseño.

Las aplicaciones web comúnmente consisten en una combinación de servlets y páginas JSP. Un ejemplo ilustrativo es un proceso de autenticación de usuario que acepta información de nombre de usuario y contraseña. El código que genera el formulario HTML o los mensajes de éxito y error debería estar en

una página JSP, mientras que el código que accede a la base de datos, valida la contraseña o autentica al usuario debería estar en un servlet.

En resumen, las páginas JSP están pensadas para la presentación visual y la lógica de negocio se asocia a los servlets.

2.4. MODELOS DE ARQUITECTURA JSP

Los tutoriales de Sun de JSP [56] describen dos aproximaciones de arquitectura para la construcción de aplicaciones utilizando la tecnología de JSP y servlet. Estas aproximaciones reciben los nombres de arquitecturas Modelo JSP 1 y Modelo JSP 2. La diferencia entre ellas radica en el modo en el que procesan las peticiones.

2.4.1. Arquitectura Modelo 1

En la arquitectura Modelo 1, el objetivo de toda petición es una página JSP. Esta página es completamente responsable de llevar a cabo todas las tareas requeridas para servir la petición. Esto incluye tareas como la autenticación del cliente, utilizando JavaBeans para acceder a los datos, o la gestión del estado del cliente. Esta arquitectura se ilustra en la *Figura 2.5*.

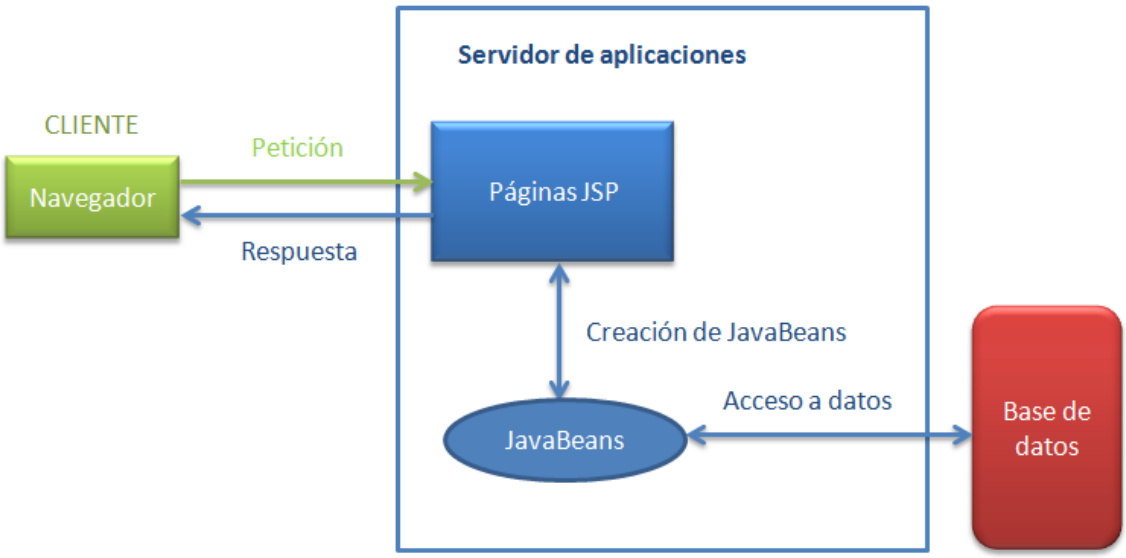


Figura 2.5 Modelo 1 de arquitectura JSP

Como se puede observar en la *Figura 2.5*, no hay un componente central que controle el flujo de trabajo de la aplicación. Esta arquitectura es apropiada para aplicaciones sencillas. Sin embargo, presenta inconvenientes relevantes que limitan su utilización para aplicaciones complejas. En primer lugar, requiere incluir la lógica de negocio utilizando grandes secciones de código Java en la página JSP. Esto supone un problema para los diseñadores de páginas web, quienes normalmente prefieren evitar la programación del lado del servidor. En segundo lugar, esta aproximación no promueve la reutilización de los componentes de la aplicación. Por ejemplo, el código escrito en una página JSP para autenticar a un usuario no puede ser reutilizado en otras páginas JSP.

2.4.2. Arquitectura Modelo 2

Esta arquitectura se ajusta al patrón de diseño Modelo-Vista-Controlador (MVC) [4]. Antes de abordar la descripción de la arquitectura Modelo 2, conviene definir dicho patrón, la motivación de su aparición y la estructura que presenta.

Patrón Modelo-Vista-Controlador.

En el ámbito de las de aplicaciones web, en sistemas con interfaces de usuario, una situación habitual consiste en que el sistema acepte datos del usuario, actualice la base de datos, y devuelva datos al usuario en un momento posterior en el tiempo.

Existen diferentes maneras tanto de aceptar como de presentar los datos a los usuarios por parte del sistema, y los datos que son entregados al sistema de una forma determinada deberían poder ser recuperables en otra forma.

En esta situación el sistema puede desplegar un componente individual que interactúe con el usuario además de mantener la base de datos, pero esto conlleva una limitación. Si se quiere realizar una petición que soporte un tipo nuevo de visualización, la modificación necesaria requerirá rediseñar el componente.

Por ejemplo, la página web de la Universidad Carlos III proporciona a los alumnos un servicio que permite consultar el calendario de exámenes. Cuando el usuario se registra en el sitio, la aplicación web permite al usuario visualizar de dos maneras diferentes el calendario de exámenes de la titulación correspondiente a la actual convocatoria. Una de las posibles presentaciones consiste en mostrar la información en forma de listado de asignaturas de la titulación, por orden alfabético, y la otra forma consiste en presentar la programación del calendario por orden cronológico. En este ejemplo, los mismos datos pueden ser visualizados de diferente manera, pero está controlado por una única entidad, la aplicación web.

En el contexto y el ejemplo presentados tienen que llevarse a cabo las siguientes tres tareas:

- Gestionar la interacción del usuario con el sistema.
- Gestionar los datos actuales.
- Dar formato a la información y presentársela al usuario.

En lugar de que un solo componente realice todas las tareas, este componente se puede dividir en tres componentes independientes, de manera que cada uno de ellos lleve a cabo una de estas actividades.

La solución propuesta por este patrón consiste en separar la presentación de los datos de su mantenimiento y tener un tercer componente que coordine las dos tareas. Estos tres componentes son denominados el Modelo, la Vista y el Controlador, y componen la base del patrón.

La característica principal de la arquitectura MVC consiste en que los tres componentes del patrón se tratan como entidades separadas. Esto presenta una gran ventaja, hay una separación definida entre los componentes del programa, lo que permite llevar a cabo su implementación por separado.

Otra de las ventajas de este patrón reside en que la conexión entre el Modelo y la Vista es dinámica, se produce en tiempo de ejecución y no en tiempo de compilación.

Gracias a estas propiedades, al basar en el modelo de arquitectura MVC el diseño de una aplicación, las piezas que la componen se pueden construir por separado y luego unirse en tiempo de ejecución. Además, es posible reemplazar posteriormente uno de los componentes sin que las otras piezas se vean afectadas.

Cada uno de los componentes tiene sus funciones dentro del patrón:

- Modelo. Es el responsable de mantener los datos o el estado de la aplicación. También gestiona el almacenaje y la recuperación de datos de la fuente de datos, y controla todas sus transformaciones. El Modelo no tiene conocimiento específico del Controlador o de la Vista, ni contiene referencias a ellos. El propio sistema es el responsable de mantener enlaces entre el Modelo y su Vista, y notificar a la Vista cuando cambia el Modelo.
- Vista. Contiene la lógica de presentación. Extrae el estado actual del sistema del Modelo y maneja la presentación de los datos, generando una representación visual del Modelo para mostrar los datos al usuario. También permite al usuario interactuar con el sistema y notifica al Controlador sobre las acciones de éste.

- Controlador. Gestiona la aplicación al completo. Instancia el Modelo y la Vista y asocia la Vista con el Modelo. Está a la espera de las acciones de los usuarios y actúa sobre los datos representados por el Modelo según el dictamen de las reglas del negocio.

La *Figura 2.6* muestra la relación entre los componentes del patrón MVC.

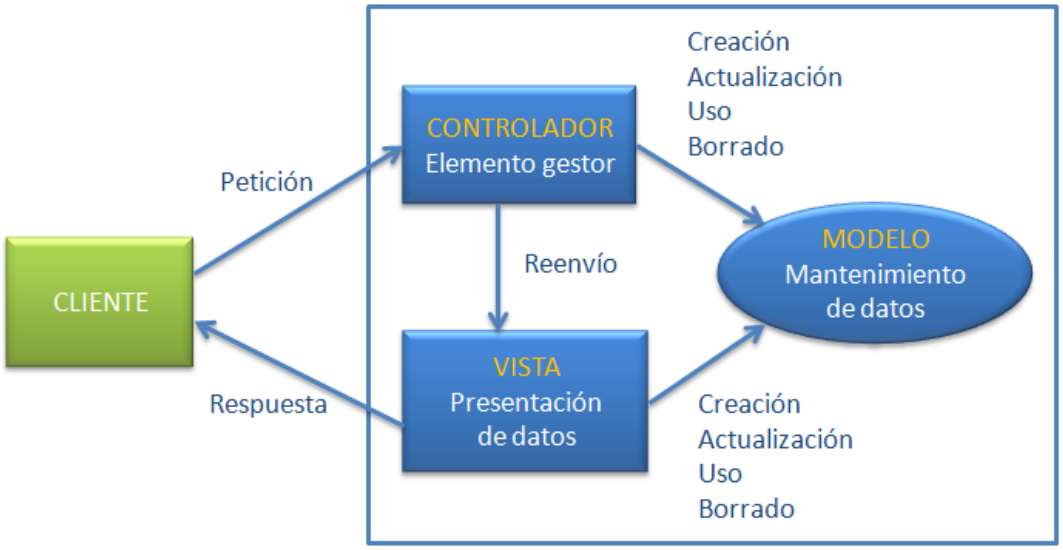


Figura 2.6 El patrón Modelo-Vista-Controlador

El ciclo de funcionamiento del patrón MVC involucra a los componentes anteriores y al cliente o usuario de la aplicación, y puede representarse esquemáticamente según el diagrama de la *Figura 2.6*.

El ciclo comienza cuando el usuario realiza una petición al Controlador, con la información necesaria sobre la acción que el usuario quiere llevar a cabo.

El Controlador gestiona la petición recibida, decidiendo a qué componente debe delegar la tarea.

Si la petición implica que se debe realizar un acceso a los datos que maneja la aplicación, el Controlador delega la petición al Modelo. El Modelo se encarga de realizar las operaciones pertinentes sobre la información de la aplicación para realizar la tarea que se le ha solicitado, y responder al Controlador con la información resultante de sus operaciones. El Controlador redirige la información recibida del Modelo a la Vista.

En caso de que la petición del usuario no requiera un acceso a la información gestionada por la aplicación, el Controlador redirige directamente la petición a la Vista.

En ambos casos, la Vista transforma los datos recibidos en información que resulta visualmente entendible para el usuario.

Existen una serie de consecuencias e implicaciones que surgen a partir de este patrón.

- Separar la representación de los datos (Modelo) de la presentación de los datos (Vista) permite múltiples Vistas para los mismos datos. Es posible realizar cambios en los componentes Modelo y Vista de forma independiente siempre que sus interfaces permanezcan sin modificar. Esto incrementa la sostenibilidad y extensibilidad del sistema.
- Separar el comportamiento de la aplicación (Controlador) de la presentación de los datos (Vista) permite al Controlador crear una Vista adecuada en tiempo de ejecución basada en el Modelo.
- Separar el comportamiento de la aplicación (Controlador) de la representación de los datos (Modelo) permite a las peticiones de los usuarios ser trasladadas del Controlador a funciones específicas de nivel de aplicación en el Modelo.

Aunque el patrón MVC trata la comunicación entre los componentes Modelo, Vista y Controlador, no se trata de un patrón conductual, ya que no especifica cómo deben comunicarse estos tres componentes. El patrón MVC solamente especifica que la estructura del sistema de componentes sea tal que cada

componente individual cumpla con uno de los tres roles – Modelo, Vista o Controlador – y solamente proporcione la funcionalidad correspondiente a su propio rol. Por tanto, se trata de un patrón estructural.

En la arquitectura Modelo 2, basada en este patrón, los objetivos de todas las peticiones son servlets que actúan como controladores de la aplicación. Analizan la petición y recolectan los datos requeridos para generar una respuesta en objetos del modelo. Finalmente, los servlets controladores envían la petición a las páginas JSP. Estas páginas utilizan los datos almacenados en el modelo para generar una respuesta. De este modo, las páginas JSP componen la vista de la aplicación. La *Figura 2.7* ilustra dicha arquitectura.

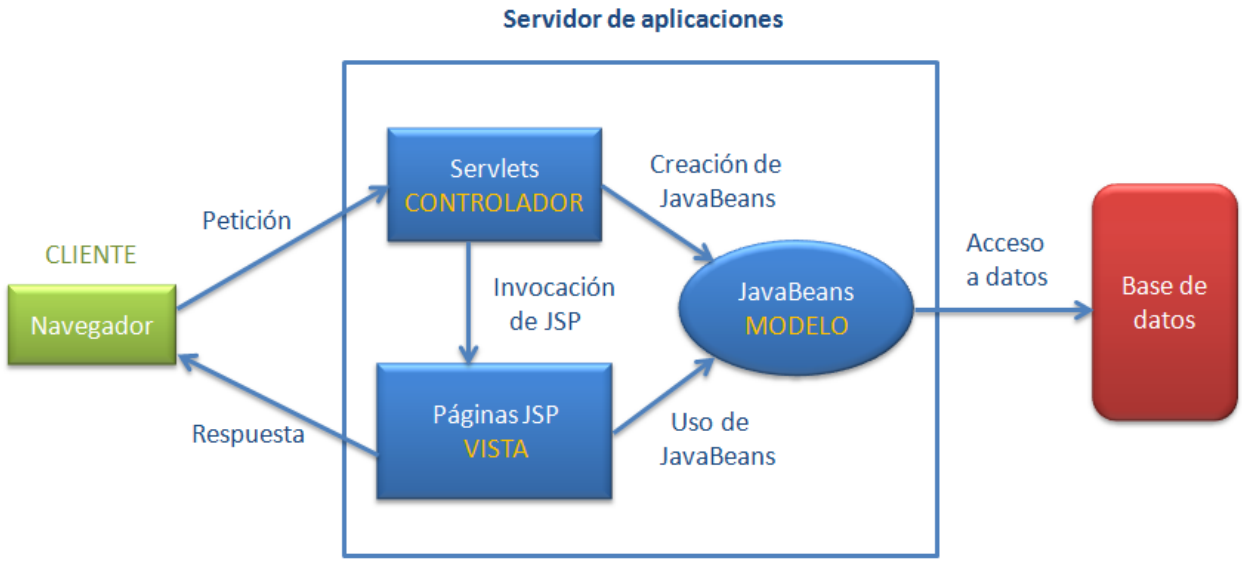


Figura 2.7. Modelo 2 de arquitectura JSP

En esta arquitectura, el modelo representa la lógica de negocio de la aplicación y puede dividirse en dos tipos de componentes:

– Componentes de estado.

Estos componentes encapsulan el estado de la aplicación y ofrecen métodos para el acceso y modificaciones de éste.

Por definición, los componentes de estado deben ser completamente independientes del protocolo de acceso a la base de datos. Así, podrán ser reutilizados en otro tipo de aplicaciones que utilicen bases de datos diferentes.

– Componentes de acción.

Los componentes de acción definen los posibles cambios del estado en respuesta a los eventos.

Este tipo de componentes no pueden ser completamente independientes del protocolo de acceso a la base de datos, pero, a pesar de esto, lo ideal es reducir esta dependencia lo máximo posible. Para ello, se puede recurrir a la construcción de dos subcapas, una de ellas dependiente del protocolo, que transforme los eventos y delegue el procesamiento a otra capa de componentes de acción independientes del protocolo.

La arquitectura Modelo 2 resuelve de manera satisfactoria los problemas asociados con la arquitectura del Modelo 1, como se explica a continuación.

- Mediante la separación de responsabilidades, aporta facilidad de mantenimiento a la aplicación que lo implementa.
- El controlador presenta un único punto de entrada a esta aplicación, proporcionando un medio más claro para implementar la seguridad y llevar a cabo la gestión de peticiones del cliente.
- Dependiendo de la petición recibida, el controlador la reenvía al componente de presentación adecuado, que a su vez es quien responde al cliente. Esto simplifica el trabajo a los diseñadores

de páginas web, permitiéndoles trabajar solamente con la presentación de los datos, ya que con este modelo las páginas JSP no requieren una lógica de negocio compleja.

2.5. BASE DE DATOS

2.5.1. SQL

SQL [60], acrónimo de *Structured Query Lenguaje*, es un lenguaje estándar de comunicación con bases de datos. Sirve para crear, manipular, examinar y gestionar bases de datos relacionales. SQL es un lenguaje normalizado que permite trabajar con cualquier lenguaje de programación en combinación con cualquier base de datos.

El hecho de que se trate de un lenguaje estándar no implica que sea idéntico para todas las bases de datos. Existen dos razones que explican que SQL presente distintas formas. La primera es que el estándar SQL es complejo, por lo que no resulta práctico implementar el estándar completo en todos los casos. La segunda razón es que determinadas bases de datos implementan funciones específicas, que no tienen por qué ser implementadas por todas ellas. Esto permite que los proveedores de bases de datos diferencien su producto unos de otros.

2.5.1.1.*MySQL*

El software MySQL [48] proporciona un servidor de base de datos SQL muy rápido, multihilo, multiusuario y robusto. El servidor MySQL está diseñado para entornos de producción críticos, con alta carga de trabajo así como para integrarse en software para ser distribuido.

Debido a su rapidez y a la facilidad de su uso, es uno de los motores de base de datos más usados en Internet.

2.5.2. JDBC

El lenguaje Java está completamente especificado, y, por definición, una plataforma Java debe soportar un conjunto conocido de bibliotecas. Una de estas bibliotecas es JDBC, *Java Database Connectivity* [38]. La API JDBC es el estándar de Java que proporciona conectividad entre el lenguaje de programación Java y un amplio rango de bases de datos. Puede acceder a cualquier conjunto de datos tabulados, especialmente datos almacenados en bases de datos relacionales.

2.5.3. SQL y JDBC

La API JDBC [17, 43] define el envío de peticiones SQL a un Sistema de Gestión de Bases de Datos (DBMS) y el posterior tratamiento del conjunto de recursos por parte de la aplicación, para el acceso a bases de datos basadas en SQL.

JDBC hace posible a los desarrolladores escribir aplicaciones en Java que gestionen las siguientes tareas:

- Establecer una conexión con una base de datos o cualquier fuente de datos tabulada.
- Enviar consultas, en nuestro caso de SQL, a la base de datos.
- Recuperar y procesar los resultados recibidos de la base de datos en respuesta a la consulta.

El siguiente fragmento de código muestra un ejemplo sencillo que incluye estos tres pasos:

```
Class.forName("com.mysql.jdbc.Driver");
      Connection con = DriverManager.getConnection("jdbc:Driver:BaseDatos",
"login", "password" );
      Statement stm = con.createStatement();
      ResultSet rs = stm.executeQuery("SELECT a, b FROM Table");
      while (rs.next()){
            int x = rs.getInt("a");
```

```
            String s = rs.getString("b");
        }
```

Este fragmento de código registra el `Driver`, instancia un objeto `DriverManager` para conectarse a la base de datos y registrarse en la base de datos, e instancia un objeto `Statement` que lleva la petición SQL a la base de datos. A continuación, instancia un objeto `ResultSet` que recupera los resultados de la petición, y ejecuta un bucle `while` sencillo, que recupera y muestra dichos resultados.

2.5.4. Drivers para JDBC

Un *driver* es un conjunto de clases que implementa las clases e interfaces de la API JDBC que son necesarias para que una aplicación Java pueda conectarse con una base de datos.

Si los desarrolladores de una base de datos desean que pueda ser accesible mediante JDBC, deben implementar un driver para dicho fin. El driver se encargará de traducir comandos estándar de la API JDBC al protocolo nativo de la base de datos.

2.5.5. Arquitectura JDBC

Los drivers de la tecnología JDBC [44] pueden ser de cuatro categorías, lo que determina los diferentes tipos de arquitectura.

Las aplicaciones pueden acceder a la base de datos a través de la API JDBC utilizando drivers de JDBC basados en tecnología puramente Java, como se muestra en la *Figura 2.8*.

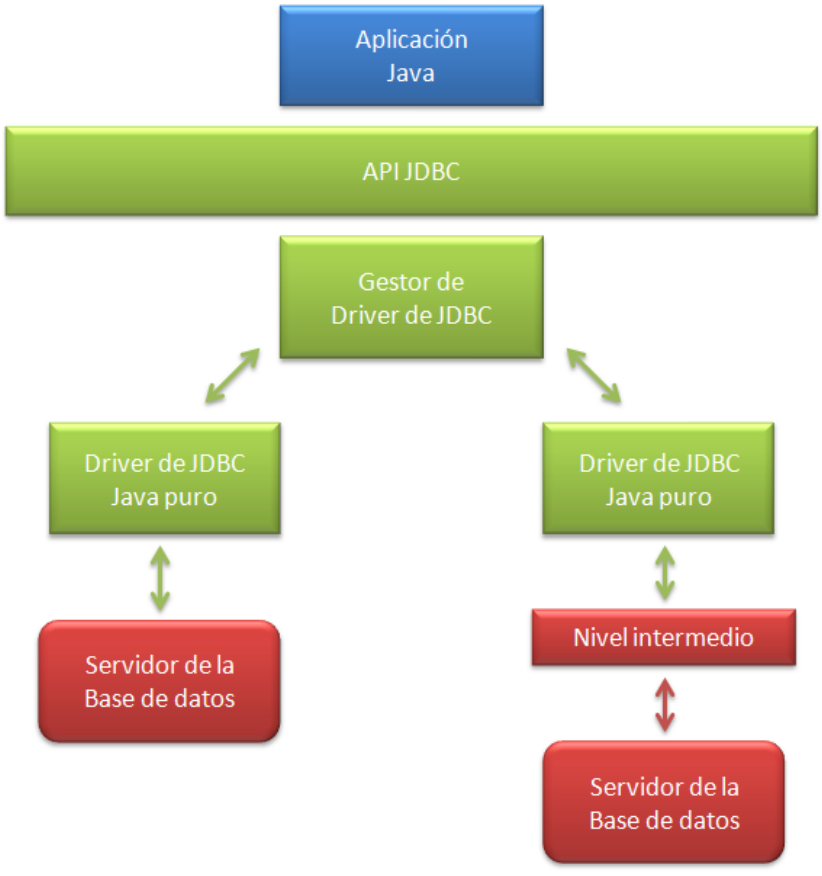


Figura 2.8 Driver Java puro

En el lado izquierdo de la *Figura 2.8* se observa una conexión directa entre la base de datos y el driver de JDBC Java puro. En cambio, en la derecha, esta conexión se realiza a través de un nivel intermedio, que proporciona conectividad a diferentes bases de datos.

La *Figura 2.9* muestra la conectividad JDBC utilizando drivers de ODBC y librerías de cliente de la base de datos existentes. ODBC, *Open Database Connectivity*, es un estándar de acceso a bases de datos desde cualquier aplicación, sin restringir a que se trate de una aplicación Java, como sucede en el estándar JDBC.

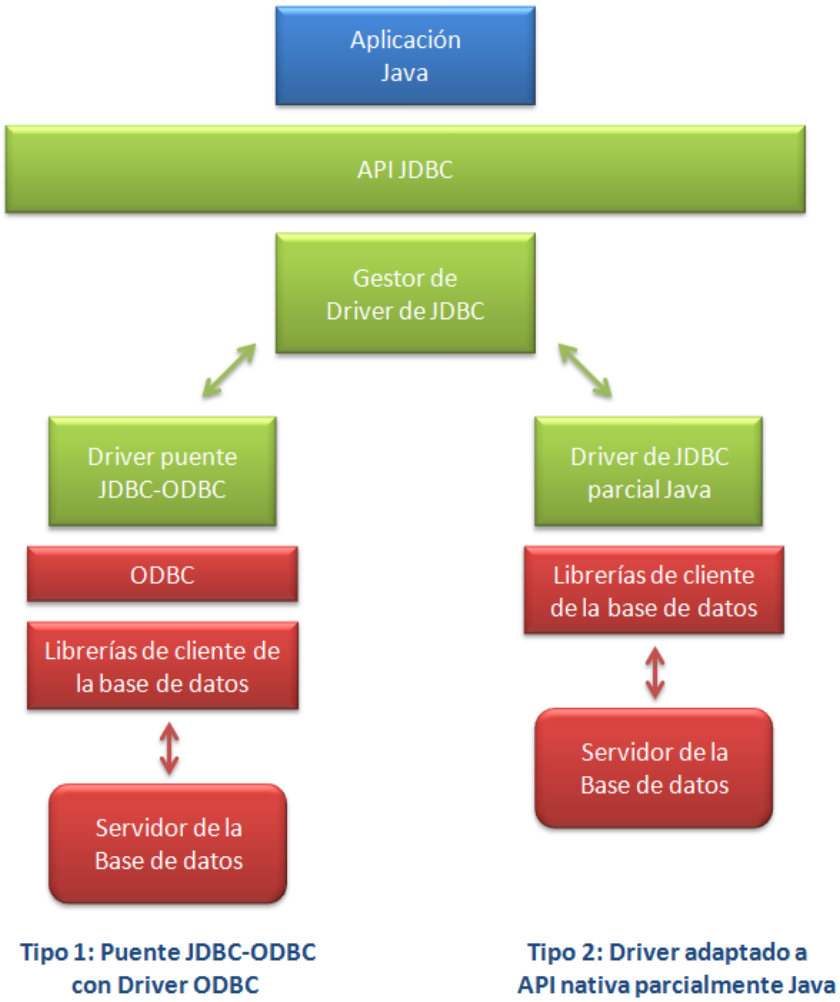


Figura 2.9 Driver ODBC y bibliotecas de cliente de la base de datos

En el lado izquierdo, además de las bibliotecas de cliente de la base de datos, se utiliza un driver de ODBC. Esto añade la necesidad de utilizar un driver que haga de puente JDBC-ODBC entre el driver de ODBC y el gestor de drivers JDBC. En el caso de la derecha, se observa que la tecnología del driver utilizado es parcialmente Java. Este driver convierte las llamadas JDBC en llamadas de la API del cliente.

3. DISEÑO DEL SISTEMA

3.1. CARACTERÍSTICAS DEL SISTEMA

La estructura y funcionalidades de esta aplicación han sido propuestas en base a las características solicitadas por el Servicio de Información Administrativa del Gobierno de Cantabria a la empresa encargada del desarrollo de la aplicación, a través del consultor senior encargado de la gestión del proyecto.

Los objetivos principales de la aplicación son, por un lado proporcionar una herramienta de trabajo a los operadores telefónicos del servicio, en la que puedan consultar la información necesaria para llevar a cabo su tarea informativa, y por otro el estudio del funcionamiento del servicio, orientado a mejorar la calidad de éste.

Para ello, el sistema debe contar con un entorno en el que los operadores tengan disponible la información correspondiente a la administración en el Gobierno de Cantabria, y dispongan además de una interfaz para poder registrar los datos relativos a las llamadas recibidas.

La aplicación almacenará estos datos, y los tratará, de manera que se puedan obtener conclusiones acerca de los parámetros que se consideren relevantes para el estudio de la eficiencia del servicio.

Los administradores del servicio tendrán acceso a estos resultados, además de llevar a cabo tareas de mantenimiento de la herramienta, como la actualización de información en base a las conclusiones obtenidas del estudio, o la gestión de los usuarios registrados en la herramienta.

A continuación se describen los diferentes aspectos considerados en el planteamiento del diseño del sistema y las decisiones tomadas en cada caso. Dichas decisiones han sido tomadas por el responsable del proyecto en la empresa, el analista programador encargado del proyecto.

3.1.1. Tipos de usuarios

Los principales usuarios de esta aplicación son los **operadores**, los cuales atienden las llamadas telefónicas de los habitantes que realizan una consulta al Servicio de Información Administrativa del Gobierno de Cantabria.

Su interacción con la aplicación consiste en recopilar la información relacionada con la consulta telefónica realizada. Para cada llamada, el operador debe introducir en un formulario ciertos aspectos tratados durante la consulta.

Existe un segundo tipo de usuario, el **administrador**. Es el encargado de gestionar la información contenida en la aplicación. Debe administrar el acceso de operadores a la aplicación y establecer las diferentes categorías de información a consultar.

El administrador añade y elimina operadores del sistema, en función de las necesidades de acceso a la aplicación por parte de los diferentes empleados del Servicio de Información Administrativa.

Por otro lado, también maneja el catálogo que recopila la información que puede ser consultada por los operadores.

Además de estas funciones de gestión, el administrador tiene acceso a las estadísticas generadas por la aplicación, que le permiten extraer conclusiones acerca de la utilización del servicio por parte de los operadores.

Se establece que el sistema debe posibilite el acceso a dos administradores. Uno de ellos realizará las funciones anteriormente descritas, propias de un administrador, y el otro podrá acceder tanto a las funciones de administración como a las correspondientes a los operadores.

3.1.2. Perfiles de acceso

El acceso al sistema y las posibilidades que se ofrecen a un usuario vienen determinados por del tipo de usuario que realice el registro. Para establecer esta diferenciación en la gestión del acceso a la aplicación se definen los siguientes perfiles de acceso.

Usuario Operador

Los usuarios con este perfil acceden a las operaciones relacionadas con el registro de información recopilada durante las llamadas telefónicas atendidas.

Usuario Administrador

El usuario con perfil de administrador accede a las operaciones de gestión y de consulta de información sobre el uso del sistema.

Puede manejar el acceso de usuarios a la aplicación y las diferentes categorías de información a consultar, y acceder a los resultados de las estadísticas generadas por la aplicación.

Usuario Mixto

Para cubrir la posibilidad de que un usuario del sistema realice funciones de operador y de administrador se crea este tipo de usuario, que tiene acceso a las operaciones de los dos perfiles anteriores.

3.1.3. Organización de operadores del servicio

El Servicio de Información Administrativa consiste en una entidad que se compone en un primer momento de una oficina, existiendo la posibilidad de ampliar el personal del servicio, y pudiendo surgir con ello la necesidad de añadir más oficinas.

En la oficina se establece la organización de operadores por grupos, y dentro de cada grupo puede haber una serie de puestos.

Los operadores creados son incluidos en un grupo, e inicialmente pertenecen todos al mismo. Los puestos que puede ocupar cada uno los seleccionará el administrador a la hora de añadirle al sistema.

Un operador puede tener más de un puesto, y esto puede suceder también en el caso de que el usuario sea de tipo mixto, con funciones de administrador y operador. Al registrarse en la aplicación un operador que tiene más de un puesto, tiene la posibilidad de indicar el puesto con el que desea registrar las operaciones de la sesión que se dispone a comenzar.

Una vez desplegado el sistema, se ofrece la posibilidad de solicitar como parte del mantenimiento de la aplicación, al igual que en el caso de la oficina, la ampliación del número de grupos y puestos, en función de las necesidades de la organización.

3.2. FUNCIONALIDADES

3.2.1. Descripción de Funcionalidades

A continuación se expone una descripción más detallada de las funcionalidades a las que puede acceder cada tipo de usuario.

Usuario Operador

El operador realiza un registro de los datos que obtiene del ciudadano cuya llamada telefónica atiende, y para ello el sistema le proporciona una interfaz de trabajo que le permite indicar dicha información.

Cuando el operador recibe una llamada, comienza un nuevo registro, y con esta acción se inicia un cronómetro, que contabiliza la duración de esta llamada.

De manera opcional, el ciudadano que realiza la llamada puede indicar sus datos personales al operador. En un primer momento no se tratará de una práctica común, ya que no es el objetivo principal de la aplicación registrar este tipo de información. Se utilizará en casos particulares que puedan ir surgiendo, por ejemplo en el caso de que el ciudadano desee que se le envíe un boletín mensual informativo por correo electrónico. En este tipo de situaciones, el operador puede anotar datos como el nombre, el NIF y el correo electrónico del ciudadano.

Si el operador considera algún otro dato a destacar acerca del ciudadano y sus características particulares, o alguna circunstancia personal relevante, también puede indicarlo.

A continuación, el operador identifica el asunto relativo a la llamada telefónica, diferenciando entre dos posibles clases.

La primera de ellas hace referencia a las llamadas de ciudadanos cuya intención es **proponer una sugerencia** o **plantear una queja**, ambas relacionadas con la Administración del Gobierno de Cantabria. El operador puede detallar la información proporcionada por el usuario para que ésta quede registrada.

El segundo tipo de llamada, la más común, consiste en la **solicitud de información**, por parte del ciudadano, relativa a la Administración del Gobierno de Cantabria. La información disponible para las consultas de los operadores se ofrece por temas, que están agrupados en diferentes categorías, como se detalla a continuación.

El operador dispone de la información necesaria, estructurada en una jerarquía de dos niveles. El primer nivel es el de categoría, de forma que al operador se le ofrece una lista con las categorías de consulta existentes, entre las que selecciona la categoría correspondiente a la solicitud recibida. Además de la categoría de la consulta, se proporciona un segundo nivel de especificación a la hora de acceder a la información requerida, pudiendo cada una de las categorías disponibles tener asociados uno o varios temas, que posibilitan acotar el campo abarcado por la categoría, y permiten un acceso más rápido a porciones de menor volumen de información.

Cada tema lleva asociado un enlace. Estos enlaces consisten en vínculos a direcciones donde está alojada la información, y pueden referenciar a una página pública de Internet, o bien a una dirección de un repositorio, donde se almacena la mayor parte de esta información.

El operador puede realizar varias consultas, hasta un máximo de diez por cada llamada. En caso de que el ciudadano deseara consultar más de diez temas, el operador debe iniciar un nuevo registro.

Adicionalmente, en cada registro realizado se visualizan determinados datos que identifican al operador y cierta información temporal.

Los datos del operador que se presentan en pantalla son su nombre y su ubicación dentro de la organización, es decir, la entidad, oficina, grupo y puesto de trabajo.

En cuanto a la información temporal, se muestran la fecha y hora de inicio del registro y un cronómetro, que contabiliza el tiempo que dura el registro desde que el operador señaliza que se ha iniciado la llamada telefónica hasta que se finaliza el proceso.

Un operador tiene un tiempo máximo de 100 minutos para realizar un registro. Se ha fijado este límite superando ampliamente la posible duración de una llamada, entendiendo que si se ha alcanzado ha sido debido a que el operador no ha finalizado el registro, pero sí se ha terminado la consulta telefónica. En caso de superarse este límite, automáticamente se da por finalizado el registro correspondiente a esa llamada.

Usuario Administrador

Como se ha mencionado anteriormente, el administrador se encarga del mantenimiento del sistema en cuanto al control de la información relativa a la aplicación, y también puede consultar las estadísticas relacionadas con los registros de consultas de los ciudadanos, calculadas por el sistema.

Ambas tareas se describen por separado, pues se trata de dos funcionalidades distintas.

El **mantenimiento** consiste en la gestión por parte del administrador de los datos manejados por la aplicación.

Respecto al catálogo que contiene la información consultada por los operadores, puede añadir y eliminar tanto categorías como los temas relacionados con cada categoría.

En cuanto a la gestión de los usuarios del sistema, tiene la capacidad de dar de alta y de baja a los operadores, controlando su acceso a la aplicación y estableciendo su posición dentro de la organización.

La segunda funcionalidad del administrador consiste en la **consulta de las estadísticas** obtenidas a partir de los registros realizados por los operadores.

Estas estadísticas se calculan de manera dinámica, cada vez que el administrador lleva a cabo una consulta se realiza el cálculo correspondiente, en función del valor de los parámetros almacenados en el sistema en ese instante.

El administrador determina el margen temporal para el cálculo de las estadísticas, y también indica si desea computar el cálculo teniendo en cuenta las acciones de todos los puestos de operadores, o la de un determinado puesto.

Una vez seleccionados el tipo de estadística y los parámetros anteriores, se presenta el resultado del cálculo solicitado.

A continuación se detalla la descripción de las estadísticas que ofrece el sistema.

Por un lado se ofrecen datos sobre el número de consultas realizadas.

- El administrador debe poder conocer el asunto relativo a las diferentes llamadas recibidas, es decir, el número de consultas de información y de sugerencias que se hayan realizado durante el periodo seleccionado.

  Dentro de las consultas de información, el administrador puede acceder al número de consultas realizadas de cada categoría.

  Del mismo modo, en cada categoría de información, se puede consultar el número de accesos por cada uno de los temas pertenecientes a dicha categoría.

También se ofrece al administrador el cálculo de los tiempos medios de consulta, en función de varios parámetros.

- Un dato interesante es el tiempo medio por cada uno de los temas consultados, para gestionar la información ofrecida para cada uno de ellos.
- Para el control tanto de la eficiencia como de la dedicación de los diferentes operarios, el administrador también tiene acceso a la media del tiempo dedicado por cada operario a consultar las peticiones de los ciudadanos.
- Por último, para poder estudiar las necesidades del servicio y seguir su evolución, se ofrecen los tiempos medios de las consultas por días, por franjas horarias y por días de la semana.

3.2.2. División de funcionalidades

De acuerdo con las operaciones requeridas y el diseño del acceso de usuarios a la aplicación, se plantea la siguiente estructura de funcionalidades del sistema.

Registro

El registro de la información asociada a una llamada telefónica es una función que lleva a cabo el operador, por lo que esta funcionalidad está presente tanto en el menú del usuario con perfil de operador como en el menú del usuario con perfil mixto.

Mantenimiento

La gestión de la información relacionada con las posibilidades de las consultas de los operadores y su acceso a la aplicación es responsabilidad del usuario con perfil de administrador, por lo que el mantenimiento forma parte del menú del usuario con perfil de administrador y con perfil mixto.

Estadísticas

Al igual que el mantenimiento, el acceso a los resultados de las estadísticas extraídas a partir de los datos de la aplicación es una función propia del administrador. También está presente en el menú del usuario con perfil de administrador y con perfil mixto.

En el siguiente apartado se explican en detalle las estadísticas que proporciona la herramienta.

3.2.3. Estadísticas

Las estadísticas suministradas por el sistema constituyen valores dinámicos, es decir, cada vez que se lleva a cabo una petición, se calcula el valor solicitado.

Para cada tipo de estadística se escoge el tiempo de inicio y fin del cálculo estadístico, y también es posible seleccionar si se desea realizar el cálculo para todos los puestos de operadores, o para un puesto concreto.

En esta sección se muestra el listado de estadísticas disponibles, compuesto por los elementos que se describen a continuación:

- Asuntos, categorías, y temas más consultados

  Se muestra este la información de manera escalonada, comenzando por los tipos de asunto consultados, que pueden ser *Consulta de Información*, *Sugerencia* o ambos, indicando el número de consultas de cada tipo que se han realizado.

  Sobre las *Sugerencias* no se proporcionará más información. En cambio, sí es posible investigar acerca del tipo de *Consultas de Información* realizadas, seleccionando dicho asunto. En este caso se muestran en una nueva pantalla las *Categorías de Consulta de Información*, detallando el número de consultas por cada una de dichas categorías.

  Finalmente, si se selecciona una de dichas categorías, se muestra una pantalla en la que se enumeran los *Temas* consultados dentro de dicha Categoría, y el número de consultas correspondiente a cada *Tema*.

  Es posible avanzar y retroceder en estas pantallas, ofreciendo así la posibilidad de consultar varias estadísticas relacionadas sin tener que comenzar de nuevo la selección desde el inicio.

  En aquellos casos en los que no se ha llevado a cabo ninguna consulta conforme a los criterios establecidos, aparecerá un mensaje indicando que no se han encontrado ocurrencias en el rango seleccionado.

- Tiempos medios por temas

  Se ofrece el tiempo medio, en segundos, dedicado a cada tema perteneciente al sistema. Este valor almacena la media de los tiempos de consulta correspondientes a cada uno de los temas.

- Tiempos medios por operario

  En este caso se muestra el tiempo medio, en segundos, dedicado por cada uno de los operarios a realizar consultas. Este tiempo medio es la media de los tiempos de las consultas atendidas por el operario correspondiente.

- Tiempos medios por días

  Esta estadística proporciona el tiempo medio dedicado a las consultas cada día, para los días pertenecientes al rango de fechas seleccionado para la consulta. Este tiempo se calcula como la media aritmética de los tiempos dedicados a las consultas realizadas ese día.

- Tiempos medios por franjas horarias

  En este caso se realiza el cálculo de la media aritmética de los tiempos dedicados a las consultas por cada franja horaria, teniendo en cuenta los días que abarque el periodo seleccionado entre *Fecha inicio* y *Fecha fin*.

  Como resultado se muestra una tabla en la que la primera columna lista las *Franjas Horarias* en intervalos de 60 minutos, y en la segunda columna aparece el *Tiempo medio*, en segundos, correspondiente a cada intervalo.

- Tiempos medios por días de la semana

  Se muestra para esta estadística una tabla en la que aparecen los *Días de la semana* en la primera columna, y a la derecha, el *Tiempo medio*, en segundos, dedicado a las consultas cada día.

  Independientemente del rango de fechas seleccionado, aparecerán los días de la semana, en orden, de lunes a domingo.

A la hora de acceder el administrador a las estadísticas, como se ha indicado previamente, se le ofrecerá la posibilidad de elegir si desea tener en cuenta en el cálculo estadístico las actividades de un puesto concreto, o de todos los puestos del sistema.

Una vez que el usuario haya seleccionado el puesto a monitorizar dentro de la organización, se mostrarán los datos de localización relacionados con éste. Es decir, el puesto, grupo, oficina y entidad a los que atañe el cálculo se indicarán en pantalla, junto con el nombre de usuario del administrador, cuando se muestre el resultado de la estadística consultada.

Para ello se utilizan los parámetros enviados en los formularios de las páginas JSP a los servlets, y los atributos de la petición que el servlet reenvía al JSP.

Para cada cálculo estadístico ofrecido se necesitará mostrar diferentes datos, por lo que el diseño de esta área consistirá en una página JSP para cada tipo de estadística y un servlet que sea capaz de gestionar de manera genérica estas páginas JSP.

En el proceso de consulta de una estadística, la primera página contendrá un formulario en el que el administrador indicará los límites deseados para el cálculo estadístico y el puesto o puestos a monitorizar. Una vez que el administrador indique su selección, se le mostrará una página con el resumen de las estadísticas solicitadas.

Se utilizará una página JSP que en primer lugar muestre un formulario, los datos del formulario se recogerán en el servlet, que se encargará de gestionar el acceso a la base de datos, y del posterior cálculo estadístico. El servlet hará un reenvío de la información en la petición de vuelta a la página JSP, que esta segunda vez ya no mostrará el formulario, sino los resultados de la estadística consultada.

3.3. ESTRUCTURA DE LA APLICACIÓN

Tras el análisis de las características y requerimientos del proyecto a realizar se ha decidido estructurar los componentes del sistema en los siguientes bloques:

- Gestión

  El bloque de gestión está formado por los servlets, que se ocupan de recuperar la información proveniente de peticiones del usuario, obtener y actualizar el estado del sistema y gestionar la interacción con la base de datos, para finalmente reenviar la petición a la vista, con los datos necesarios para la presentación.

- Presentación

  Este bloque está compuesto por un conjunto de páginas JSP, desarrolladas para permitir la interacción del usuario con la aplicación. Por un lado se encargan de generar la correcta

visualización del resultado de una petición del usuario, y a su vez permiten al usuario interactuar con la aplicación, realizando una nueva solicitud.

Estas solicitudes pueden estar dirigidas a un servlet, del bloque de gestión, o a otra página JSP, solicitando una nueva vista, cuya generación no requiere de un procesamiento complejo.

- Estado

  Contiene las clases Java que definen los objetos que tienen una representación en la base de datos. Estos objetos Java, con sus propiedades y métodos de tipo get y set correspondientes, se utilizan para representar el contenido de la base de datos en objetos y propiedades que pueden ser manejados por el código Java de la aplicación.

Los componentes de este esquema siguen un patrón que se asemeja al patrón MVC. Como se ha descrito en el apartado dedicado al estado del arte, el patrón MVC es un patrón estructural, no conductual. Es decir, se define la funcionalidad de cada elemento por separado, pero no la interacción entre dichos elementos.

- Capa de acceso a datos

  Por otro lado, es necesaria la implementación de la conexión con la base de datos, que permita realizar las transacciones solicitadas por la aplicación. Este módulo se encarga de llevar a cabo los accesos a la base de datos y de la interpretación de la información para cambiar de una forma de representación a otra, entre los componentes de estado de la aplicación y los objetos de la base de datos.

En la *Figura 3.1* se detalla el modelo de arquitectura del sistema, indicando los diferentes bloques y la interacción entre ellos.

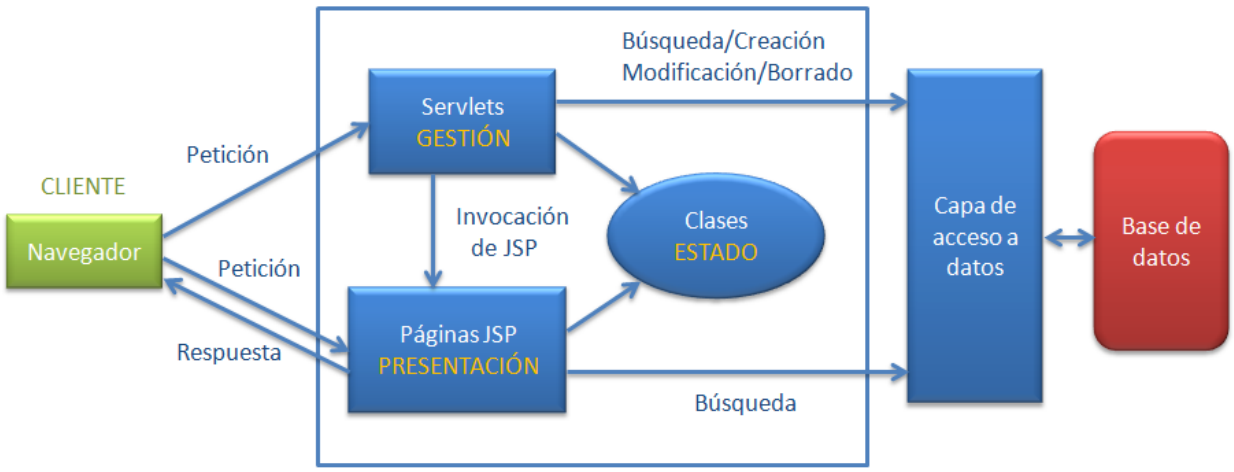


Figura 3.1 Arquitectura del sistema

La funcionalidad de los elementos descritos en el esquema de diseño del sistema es similar al patrón MVC, con algunas discrepancias, que se describen a continuación.

1. Las peticiones del cliente en el sistema diseñado no siempre van dirigidas al bloque de gestión. Estas peticiones pueden bien solicitar la ejecución de un servlet perteneciente al bloque de gestión, o bien solicitar una página JSP.

   Por tanto, en cuanto al destino de las peticiones, la arquitectura del sistema mezcla el Modelo 1 y el Modelo 2 de arquitectura JSP descritos en la sección dedicada al estado del arte, resultando el esquema de peticiones y respuestas entre el cliente y el sistema representado en la *Figura 3.1*.

   La razón por la que se ha implementado de esta manera ha sido para disminuir la carga de gestión de solicitudes del cliente, desviando directamente a la página JSP aquellas peticiones cuyo destino es una página JSP, y que no requieren una gestión compleja por parte de un servlet.

Dado que la gestión de estas solicitudes por parte del bloque gestor consistiría simplemente en redirigir la petición a una página JSP, se envía la petición directamente a la página JSP correspondiente, simplificando así el flujo a través de elementos de la aplicación.

2. El acceso a la base de datos se realiza principalmente desde el bloque de gestión, y a través de la capa de acceso a datos. Estos accesos gestionan operaciones de búsqueda, inserción, modificación y eliminación de contenido de la base de datos del sistema.

   También se llevan a cabo accesos a la base de datos desde el bloque de presentación. Estos accesos son menos habituales, y consisten siempre en consultas simples de contenido, con la finalidad de ser mostrado al cliente en determinados puntos de la aplicación. Con consultas simples se hace referencia a aquellas consultas de información que no requieren una búsqueda por parámetros o valores, sino que se trata de consultas en bloque de todas las entradas del campo solicitado de una tabla.

   Un ejemplo de este tipo de acceso puede ser el que se realiza en la página JSP desde la que se permite al usuario dar de baja una categoría. Consiste en la búsqueda de los nombres de todas las categorías del sistema, para ser mostrados en el formulario de la página JSP, de forma que el usuario pueda seleccionar la categoría que quiere dar de baja.

3. El modelo del sistema está formado por componentes de estado. Para encapsular el estado de la aplicación se definen las clases Java que representan los elementos del dominio.

   No es necesario definir componentes de acción en el modelo, ya que los posibles cambios del estado de la aplicación están recogidos en el bloque gestor. Por ello, al modelo del sistema se le denomina directamente bloque de estado.

A continuación se detallan las decisiones de diseño tomadas en cada uno de los bloques del modelo de arquitectura diseñado para el sistema.

3.3.1. Estado y Almacenamiento de información

3.3.1.1. *Diseño global*

El diseño de los componentes del bloque que engloba el estado de esta aplicación está íntimamente relacionado con el diseño de la estructura de la base de datos, y en base a estas dos entidades se ha diseñado la capa de acceso a datos.

Bloque de estado

El estado del sistema está representado por un conjunto de clases con el mismo esquema. Tienen estructura de *beans* [5], cada una de ellas contiene un conjunto de propiedades, y proporciona como únicos métodos los correspondientes *get* y *set* para acceder a las propiedades descritas.

Capa de acceso a datos

El acceso a base de datos se realiza mediante la capa de acceso a datos. Los métodos de de esta capa devuelven objetos del tipo de las clases del bloque de estado de la aplicación, que posteriormente se emplean como objetos de transferencia entre capas.

La capa de acceso a datos se ha dividido en dos subcapas.

Una de ellas representa el acceso genérico a la base de datos, implementando las funciones principales de acceso, válidas para todos los elementos que forman el bloque de estado del sistema, y cuya implementación es independiente del tipo de base de datos utilizado.

La segunda capa está formada por los elementos que particularizan estos accesos para cada tipo de objeto del bloque de estado del sistema. En ella se definen componentes que describen los diferentes accesos en el lenguaje del protocolo utilizado, MySQL, teniendo en cuenta la estructura de la base de datos que se ha diseñado y la interacción con los elementos de la aplicación.

La *Figura 3.2* muestra por un lado la relación entre los componentes de estado de la aplicación, que se comunican con la capa de acceso a datos utilizando el leguaje de programación Java, y por otro la comunicación entre la capa de acceso a datos y la base de datos del sistema, que se realiza utilizando el lenguaje de la base de datos MySQL, a través de conectores JDBC.

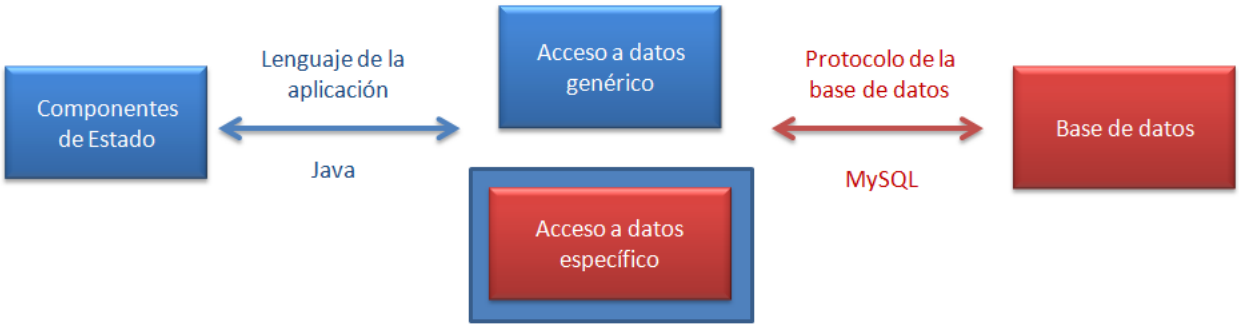


Figura 3.2 Estado, Acceso a datos y Base de datos

3.3.1.2. *Estructura de la base de datos*

La base de datos de la aplicación contiene la información utilizada por el sistema, organizada en bloques de datos relacionados entre sí, que se almacenan en forma de tablas.

Se ha decidido que estas tablas, además del contenido específico de cada una de ellas, contengan los siguientes parámetros comunes.

Identificador y nombre.

Cada entrada de una tabla está representada por estos dos parámetros, su nombre y su identificador.

El nombre de la entrada de una tabla la representa, identificándola de manera intuitiva. Este nombre se utilizará para ser mostrado al usuario de la herramienta cuando sea conveniente. Por ejemplo, al operador se le ofrecen los nombres de las categorías y temas que puede consultar, o el administrador que esté creando un usuario, podrá ver los puestos del sistema para escoger los que le asignará.

El identificador es un dato numérico, único para cada entrada de una tabla. Este campo en algunos casos estará contenido en otra tabla para hacer referencia a la entrada correspondiente de la tabla que lo define.

Estado

El indicador de estado establece si la entrada de la tabla está activa o inactiva. Las entradas en uso pertenecientes al sistema tendrán estado activo, mientras que se lleva a cabo un borrado lógico de aquellas que se eliminen, pasando a tener estado inactivo, en lugar de ser borradas definitivamente de la base de datos.

Esto puede ser de utilidad en los casos en los que se desea recuperar contenido eliminado por error. Esta recuperación de datos formaría parte del mantenimiento de la aplicación.

Fecha de la creación y fecha de la última modificación.

El primer campo, fecha de creación, se rellena automáticamente al crear una nueva entrada en la tabla.

El campo correspondiente a la fecha de la última modificación también se establecerá automáticamente, en el momento en el que se produzca una modificación. En esta aplicación las posibles modificaciones de las entradas consisten en cambios de estado. Por tanto, se actualizará este valor cuando una entrada cambie su estado de activa a inactiva o viceversa. Inicialmente este parámetro es igual al de la fecha de creación.

A continuación se estudia qué tablas es necesario implementar para albergar la información del sistema.

Los tipos de usuario se identifican en función de su perfil, por lo que se define la tabla de dicho nombre, *perfil*, para almacenar los diferentes perfiles que existen en el sistema, asociando un identificador a cada nombre de perfil. Se trata de una tabla estática, cuyo contenido no se modifica en transcurso de la utilización de la herramienta.

Los datos de identificación de los diferentes usuarios que tienen acceso al sistema se almacenan en la tabla *usuario*. Contiene en un primer momento los dos usuarios administradores del sistema especificados previamente (en el apartado "3.1.1 Tipos de usuarios"), el primero con perfil únicamente de administrador y el segundo con perfil conjunto de administrador y operador.

Estos administradores pueden crear nuevos usuarios de la aplicación con perfil de operador, que se almacenarán en esta tabla de manera dinámica, pudiendo ser también eliminados posteriormente por los administradores.

Cada usuario llevará asociado en la tabla el identificador de perfil correspondiente de los definidos en la tabla *perfil*.

Los perfiles asociados a los diferentes usuarios se han establecido según se indica en la *Tabla 3.1.*

Identificador de Perfil	Tipo de usuario
1	Administrador – Operador
2	Administrador
3	Operador

Tabla 3.1 Asociación: Perfil – Tipo de usuario

El sistema también debe almacenar en la base de datos la información relacionada con cada operador en cuanto a su situación dentro de la organización del servicio. Deberá indicarse el puesto que ocupa dicho usuario, el grupo al que pertenece, la oficina que aloja dicho grupo y la entidad que engloba esta organización.

Para llevar a cabo esta tarea de manera estructurada, posibilitando el acceso a la información relativa a los diferentes niveles dentro de la organización, se ha planificado un diseño de la base de datos de forma que cada nivel esté representado por una tabla con su nombre.

Así, los niveles indicados dan lugar a las siguientes tablas, ordenadas de estrato inferior a superior en la jerarquía organizativa:

puesto

grupo

oficina

entidad

El diseño de esta estructura de tablas se basa en la asociación de cada entrada de la tabla de un determinado nivel con otra entrada de la tabla del nivel inmediatamente superior. Por medio de estas asociaciones se proporciona una estructura en la que cada *puesto* está asociado a un único *grupo*, cada *grupo* pertenece a una sola *oficina*, y las *oficinas* forman parte de una *entidad*.

De esta manera, al establecer un determinado puesto para un usuario, se le está situando dentro de la organización, llevando asociados implícitamente el grupo, la oficina, y la entidad a las que pertenece dicho puesto.

Son tablas cuyo contenido es estático, se establecen las entradas de los diferentes niveles al inicio, así como las relaciones de pertenencia entre ellas con respecto a entradas de niveles superiores.

Como a cada usuario operador se le puede asignar más de un puesto, y en la tabla *usuario* interesa que solo haya una entrada por cada usuario, para vincular a los usuarios con los puestos que se les pueden asociar se ha diseñado la siguiente tabla:

usuario-puesto.

Su cometido es registrar las asociaciones usuario-puesto, de forma que cada uno de los usuarios tendrá una entrada en la tabla *usuario-puesto* por cada puesto que tenga asociado.

Los puestos que puede ocupar un usuario de tipo operador se establecen a la hora de añadirlo al sistema, quedando definidas estas asociaciones. Después serán consultadas tras la autenticación del operador en la herramienta, de manera que éste pueda seleccionar el puesto que va a utilizar en esa sesión.

A la hora de realizar un registro, el operador selecciona el asunto relacionado con la llamada telefónica, y en caso de que se trate de una consulta de información, accede a las categorías y temas relacionados. Esta información se almacena en las siguientes tablas:

asunto

categoría

tema

La primera de ellas, *asunto*, es una tabla estática, ya que los dos tipos de asunto no van a ser modificados con el uso de la herramienta.

En cambio las tablas *categoría* y *tema* son de contenido dinámico, que los administradores de la aplicación van a poder modificar. Al añadir y eliminar categorías y temas de consulta del sistema, se insertarán e inhabilitarán las correspondientes entradas en estas tablas.

En la tabla *tema* se incluyen los campos necesarios para asociar cada tema introducido con la categoría a la que pertenece y con la URL que contiene la información sobre el tema en cuestión.

Por último, para almacenar el registro de los informes realizados se utilizan varias tablas en la base de datos.

En primer lugar se tiene la tabla *informe*, que almacena la información introducida por el operador en cada informe, además de los datos del operador que atiende la llamada.

Durante una llamada se pueden realizar varias consultas. Esto quiere decir que un mismo informe puede contener diferentes consultas. Para almacenar la información relacionada con cada una de ellas se utiliza la tabla *consulta.*

Se almacena por un lado el informe al que está asociada cada entrada, y por otro la información a la que se ha accedido en cada caso, introduciendo la categoría y el tema consultados.

3.3.1.3. *Componentes de estado y Capa de acceso a datos*

Una vez se han establecido la estructura y contenido de la base de datos, se procede a estudiar el diseño relativo a los diferentes componentes de estado y de la capa de acceso a datos.

- Componentes de estado.

Los objetos que constituyen estos componentes son la representación del contenido de la base de datos en elementos del dominio, de tipo *bean* [5], con sus correspondientes atributos, y métodos de tipo *get* y *set*. Todo el contenido de las tablas de la base de datos tiene su representación en estos elementos y en sus propiedades.

Además de los objetos mencionados, se definen otros dos componentes de estado.

El primero de ellos se utiliza para obtener la fecha y hora del sistema en el momento que sean solicitados. Mediante el uso de estos objetos, se hace posible el almacenamiento en la base de datos del instante de creación y modificación de cada entrada en las tablas.

Esto posibilita el posterior cálculo de las estadísticas, ya que permite consultar la fecha de creación de las entradas, y atenerse a los límites temporales fijados para cada cálculo.

También es necesario definir un último componente de utilidad para la presentación de las estadísticas. Contiene el resultado obtenido tras un cálculo estadístico, y los parámetros relativos a la operación llevada a cabo.

- Capa de acceso a datos.

Como se ha detallado previamente en el apartado dedicado al estado del arte, se denomina DAO (Data Access Object) u Objeto de Acceso a Datos [13] a un componente de software que suministra una interfaz común entre una aplicación y uno o más dispositivos de almacenamiento de datos.

Se ha estructurado esta capa de acceso a datos de manera que exista una clase DAO genérica, que defina las variables y métodos necesarios para llevar a cabo el acceso a la base de datos. De esta clase heredarán las clases DAO particulares, existiendo una clase DAO específica por cada clase que defina un componente de estado con representación en base de datos.

La clase DAO genérica, aunque define todos los componentes de la capa de acceso a datos, en su implementación es independiente del protocolo de acceso a la base de datos. Así, en esta clase se declaran todas las variables y métodos necesarios para acceder a la base de datos, pero solo se implementan aquellos métodos que son independientes del protocolo de acceso a la base de datos, y que serán comunes para los accesos llevados a cabo en todas las clases DAO particulares que heredan de ella.

Cada una de las clases DAO específicas contiene las variables y métodos que definen el acceso particular a la correspondiente tabla y al componente de estado a los que están asociados. Estas variables y métodos permiten "traducir" la información de la base de datos a objetos de estado del sistema, haciendo uso para este fin en su implementación del protocolo de acceso a la base de datos.

A continuación se ofrece una breve descripción y ubicación de los elementos definidos en la clase que implementa la representación genérica de un DAO. Estos elementos serán posteriormente implementados en dicha clase, o en cada clase DAO específica que hereda de ella.

- Las variables son cadenas de texto que corresponden a las distintas sentencias preparadas MySQL.

  Estas peticiones de información deben definirse de manera específica para cada tabla de la base de datos. Por ello, las variables son declaradas en la clase que define el DAO genérico, e instanciadas por los constructores de las clases DAO particulares.

Los métodos se pueden clasificar en tres grupos:

- Métodos de interacción entre los objetos de la aplicación y la información de la base de datos.

  Estos métodos se utilizan tanto para preparar sentencias de acceso a la base de datos, como para guardar información obtenida tras un acceso a la base de datos, utilizando para ello los componentes de estado de la aplicación.

  Se declaran en la clase que representa el DAO genérico y se instancian en las clases DAO específicas.

- Métodos de gestión de la conexión con la base de datos.

  Este conjunto de métodos se implementa en la propia clase que define el DAO genérico, proporcionando una forma sencilla de manejo de la conexión con la base de datos, que se utilizará independientemente del tipo de acceso que se quiera realizar.

- Métodos de acceso a la base de datos.

  Estos métodos harán uso de las variables, de los métodos de interacción y de los métodos de gestión de la conexión descritos previamente, para llevar a cabo los accesos a la base de datos.

  Este grupo de métodos se implementa en la clase que representa el DAO genérico, y proporciona las funciones básicas de acceso general a la base de datos.

En la *Figura 3.3* se muestra el esquema de las variables y métodos, indicando si se implementan en cada una de las clases DAO específicas, o bien en la propia clase abstracta que define el DAO genérico.

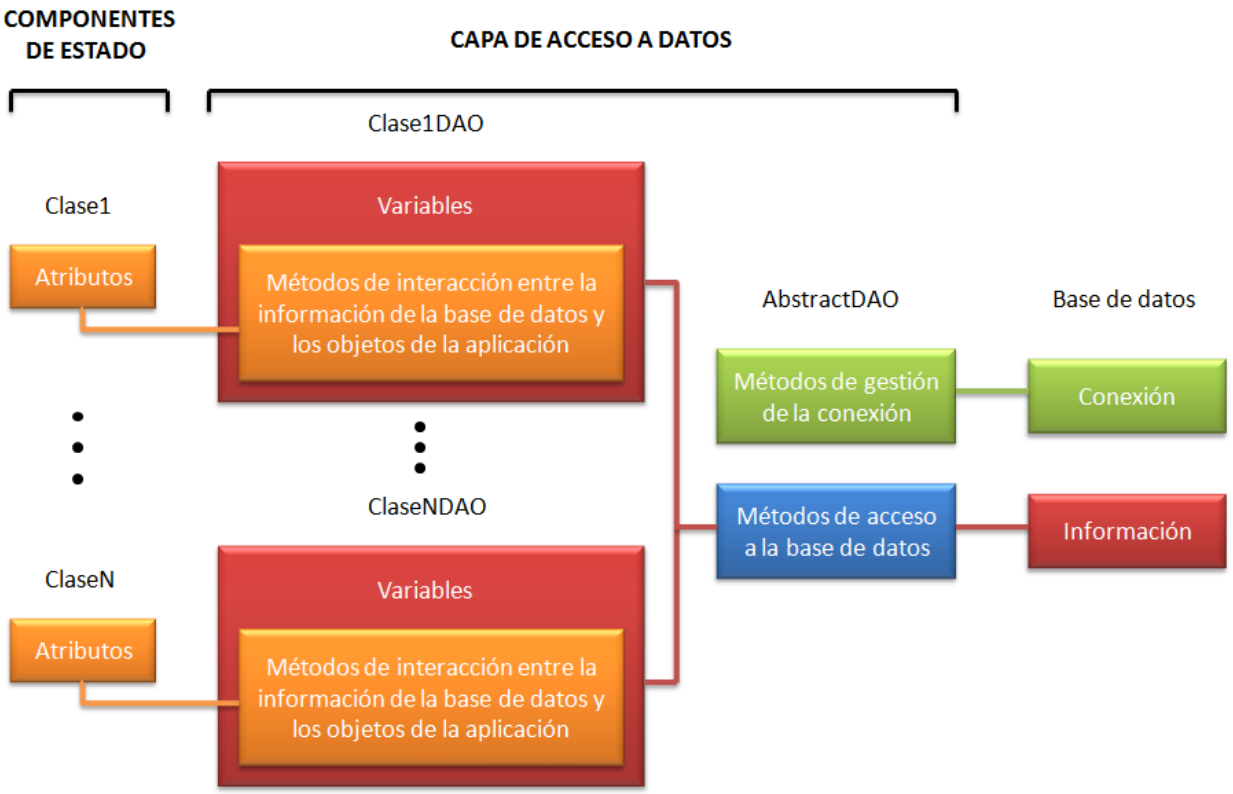


Figura 3.3 Esquema estado y datos del sistema

3.3.2. Gestión y Presentación

Los elementos que conforman el bloque de presentación de la aplicación son las páginas JSP, que proporcionan la representación de información que se muestra al usuario, permitiendo a su vez la solicitud de peticiones por parte de éste.

Las unidades de gestión que constituyen el bloque gestor son los servlets, que reciben las peticiones con carga de gestión de los usuarios, realizan las consultas requeridas a la base de datos y seleccionan la información que se devuelve al usuario tras su petición.

El bloque de presentación también puede recibir peticiones de los usuarios, que en este caso no conllevan prácticamente carga de gestión, sino que la finalidad básica de estas peticiones es la presentación de información al usuario.

Según este comportamiento, los posibles accesos al sistema se clasifican de la siguiente manera.

- Petición de gestión

  El cliente hace una petición destinada a un servlet.

  El servlet utiliza los objetos que definen el estado de la aplicación y gestiona los accesos necesarios a la base de datos, utilizando para ello los elementos de la capa de acceso a datos.

  Una vez realizadas las gestiones requeridas para completar la acción solicitada por el usuario, el servlet reenvía la petición a la página JSP correspondiente, que genera la presentación adecuada para mostrar al cliente el resultado de su petición a través del navegador.

- Petición de presentación

  La petición del cliente tiene como destino una página JSP. Estas peticiones de páginas JSP se realizan cuando la petición del cliente no requiere prácticamente gestión por parte de la aplicación, sino solamente la recuperación y presentación de otra página JSP, típicamente un formulario, o una página con enlaces a otras páginas JSP.

El cliente hace la petición de la página JSP, que ejecuta su código, accediendo en algunas ocasiones a la base de datos, y devuelve al usuario la presentación correspondiente.

Las páginas JSP y los servlets se van cediendo el control de la ejecución de la aplicación, en función de si se necesita presentar o recoger información del usuario, para lo que se usa un elemento del bloque de presentación, o de si se requiere una acción más compleja, que conlleve la utilización de un elemento del bloque gestor. Este es el motivo por el que el diseño de los módulos de presentación y gestión se ha realizado de manera conjunta.

A continuación se presentan las fases de la aplicación, y las necesidades de generación de elementos de estos bloques en cada caso.

❖ Autenticación.

En esta fase, inicio de la aplicación, se necesita en primer lugar un elemento de presentación para la "Autenticación", que permita al usuario introducir sus datos de acceso a la herramienta.

Estos datos requieren ser tratados por un elemento del bloque gestor de "Autenticación", que busca el usuario en la base de datos, y una vez autenticado, lo clasifica en función de su perfil y del número de puestos que tiene. A partir de esta clasificación, el elemento gestor decide cuál es el siguiente elemento de presentación al que debe reenviar la petición, de manera que tome el control de la aplicación.

La clasificación de los usuarios responde al siguiente esquema:

- Si se trata de un usuario con varios puestos asignados, tenga perfil de operador, o de administrador y operador, el siguiente elemento de presentación es el de "Selección de puesto"
- En el caso de un usuario con un puesto asignado, si es un operador, el elemento gestor cede el control al elemento de presentación de "Opción: Registro".
- Si el usuario tiene un puesto asignado, y tiene perfil de administrador o de administrador y operador, el elemento de presentación que toma el control de la aplicación es el de "Opciones".

La *Figura 3.4* representa los elementos de cada bloque que están involucrados en el proceso de autenticación.

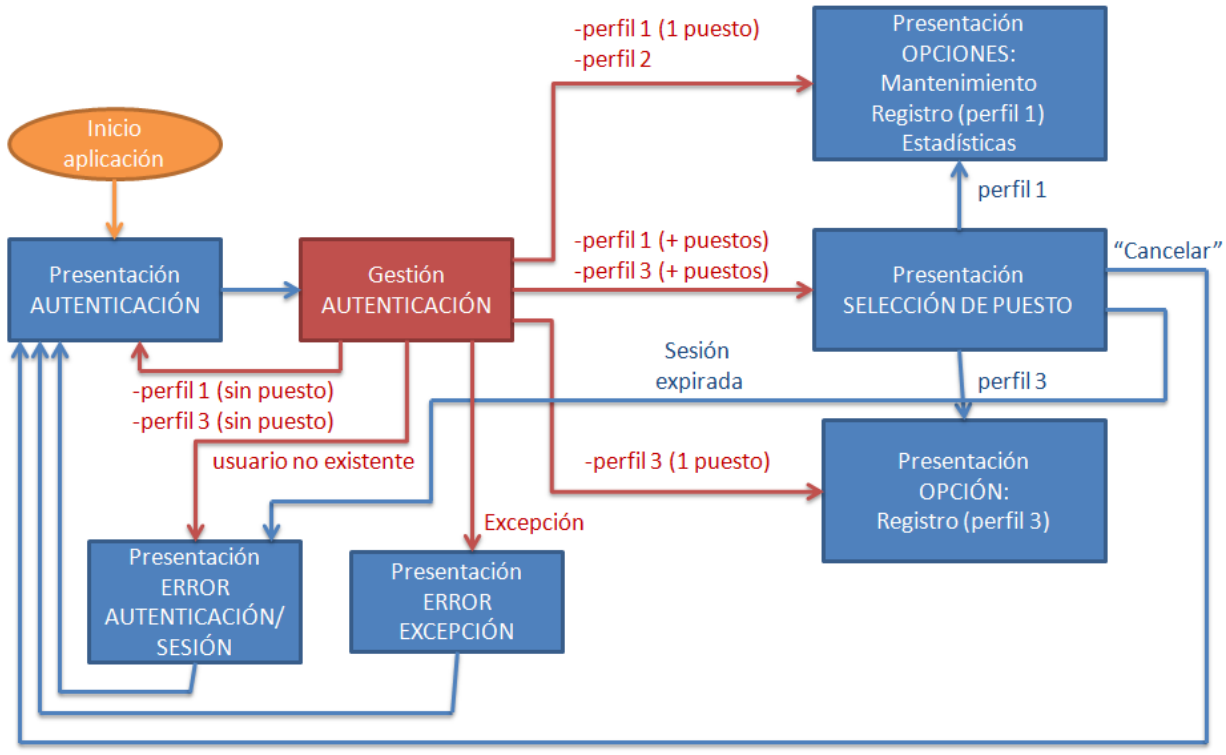


Figura 3.4 Gestión y Presentación. Autenticación – Selección de puesto

❖ Selección de puesto.

A través del elemento de presentación de "Selección de puesto" el usuario elige el puesto que va a ocupar en la sesión que está iniciando. A continuación, como puede verse en la *Figura 3.4*, en función del perfil de usuario, se requiere la presentación de uno de los siguientes elementos:

- En caso de que el usuario tenga perfil de administrador y operador, el siguiente elemento de presentación a ofrecer al usuario es el de "Opciones"
- Si por otro lado el perfil es de operador, el elemento de presentación que debe tomar el control de la aplicación es el de "Opción: Registro".

❖ Opciones de administración.

El elemento de presentación "Opciones" ofrece al usuario las posibles acciones que puede llevar a cabo como administrador de la herramienta, y es la primera página que se muestra a este tipo de usuario tras su correcta autenticación y selección de puesto, si procede.

Se trata de una página distribuidora, en la que el usuario seleccionará una de las opciones. Con esta selección, se cederá el control de la aplicación a un nuevo elemento del bloque de presentación, que mostrará la página JSP asociada a la acción solicitada. Las posibles opciones están representadas en las siguientes 3 figuras.

❖ Opciones de mantenimiento.

El mantenimiento de la herramienta consiste en gestionar las altas y bajas de diferentes elementos del sistema, que son las categorías, los temas y los operadores.

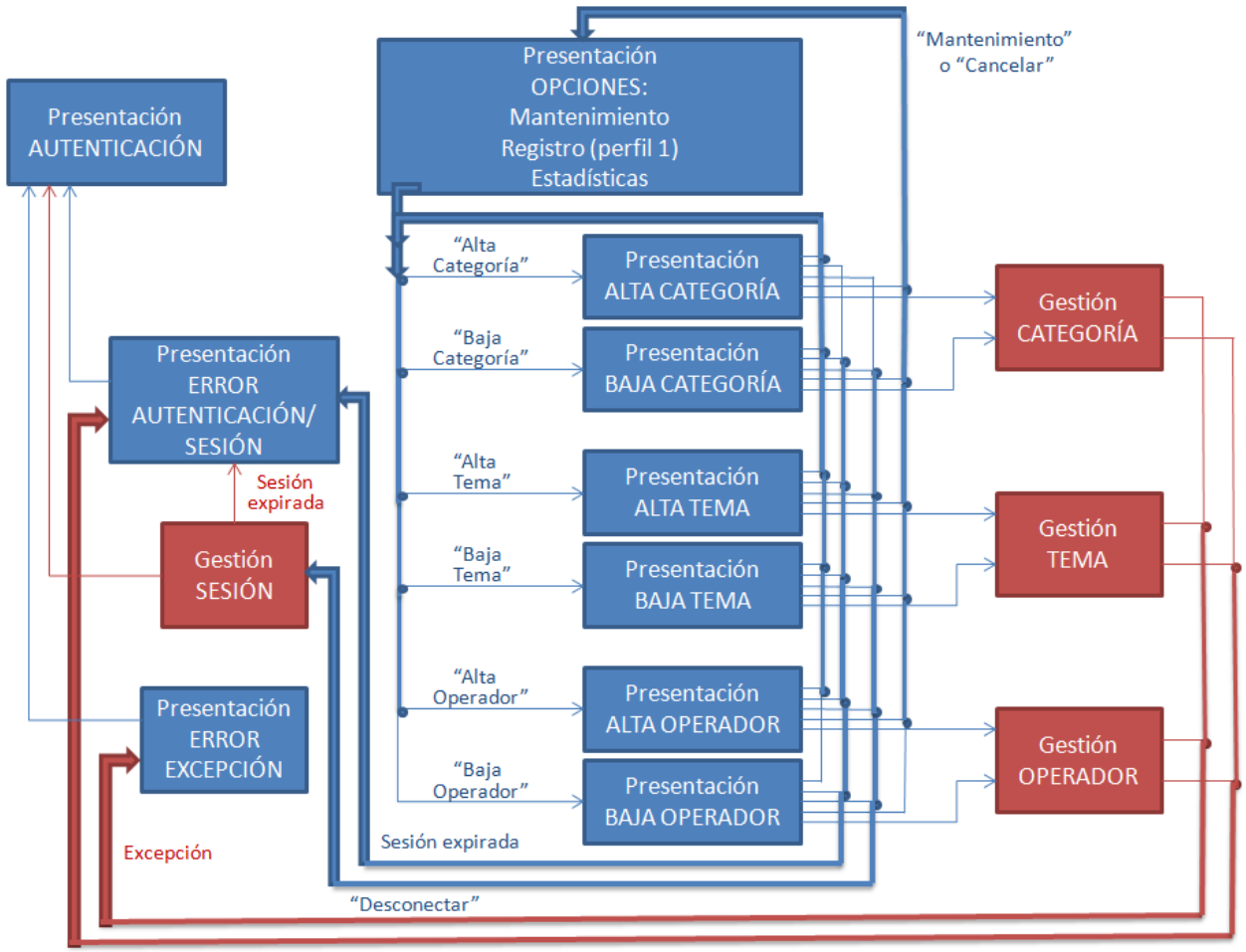


Figura 3.5 Gestión y Presentación. Mantenimiento

Cada una de estas acciones tiene asociado un elemento de presentación, que se invocará al ser seleccionada la opción correspondiente, como se puede apreciar en la *Figura 3.5*.

Para cada uno de estos tres elementos que pueden ser añadidos o eliminados del sistema se requiere de un bloque gestor, que lleve a cabo las acciones necesarias para completar el alta o la baja del elemento en cuestión.

- Estadísticas.

Para acceder a la presentación de las estadísticas ofrecidas por el sistema, basta con seleccionar en la barra de opciones de cualquiera de las páginas a las que accede el administrador, lo que cederá el control al elemento de presentación "Estadísticas".

Como se observa en la *Figura 3.6*, en esta página se produce una ramificación de caminos hacia los diferentes elementos de presentación relacionados con cada estadística del listado ofrecido.

También es necesario un elemento del bloque de gestión que recoja la información solicitada en cada estadística, la calcule, y devuelva su resultado en forma de listado, que será representado de nuevo por el mismo elemento de presentación de la estadística inicialmente escogida.

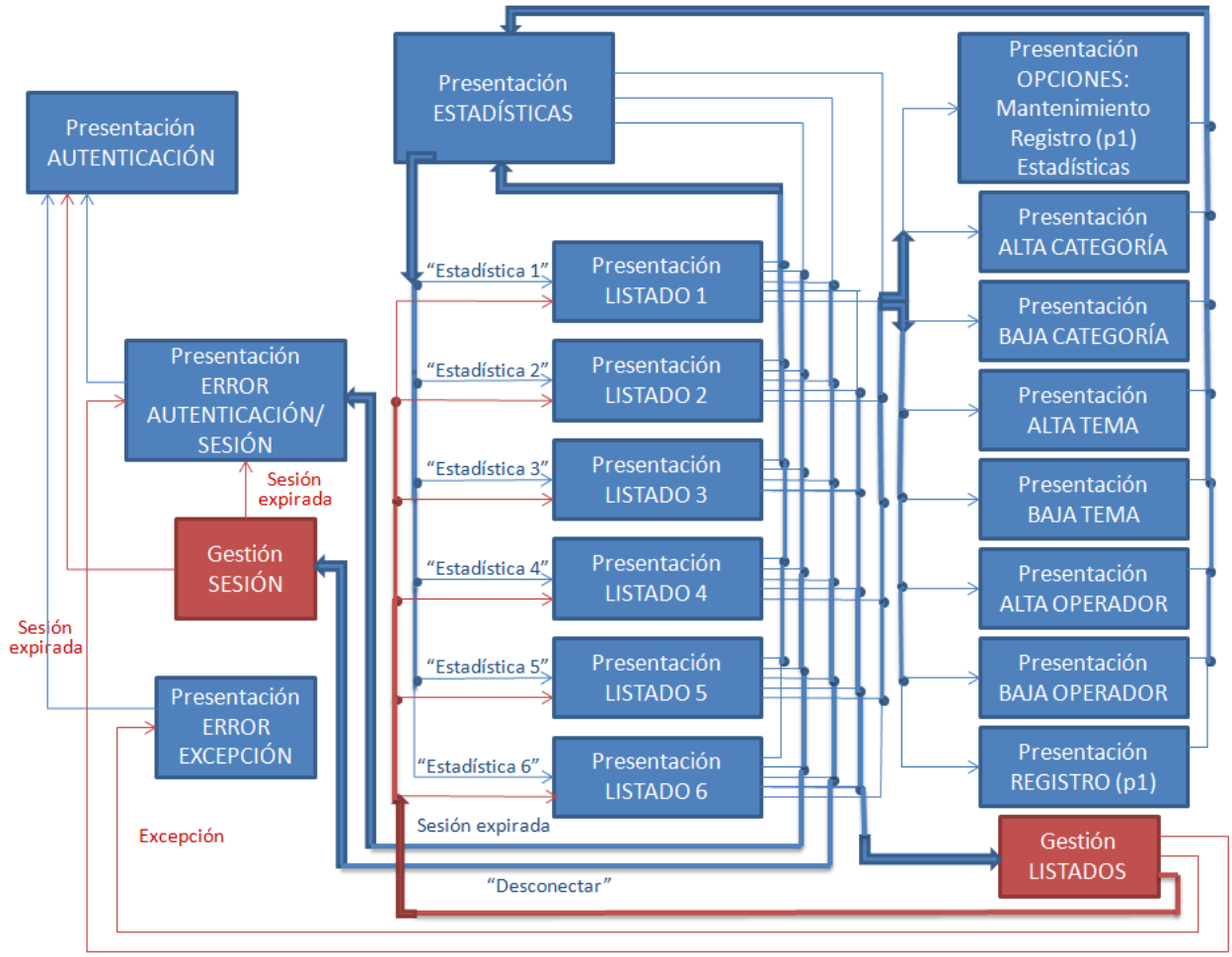


Figura 3.6 Gestión y Presentación. Estadísticas

❖ Opción de registro.

Esta opción se ofrece al usuario tanto en el elemento de presentación "Opciones", como en el elemento "Opción: Registro", como representa la *Figura 3.7*.

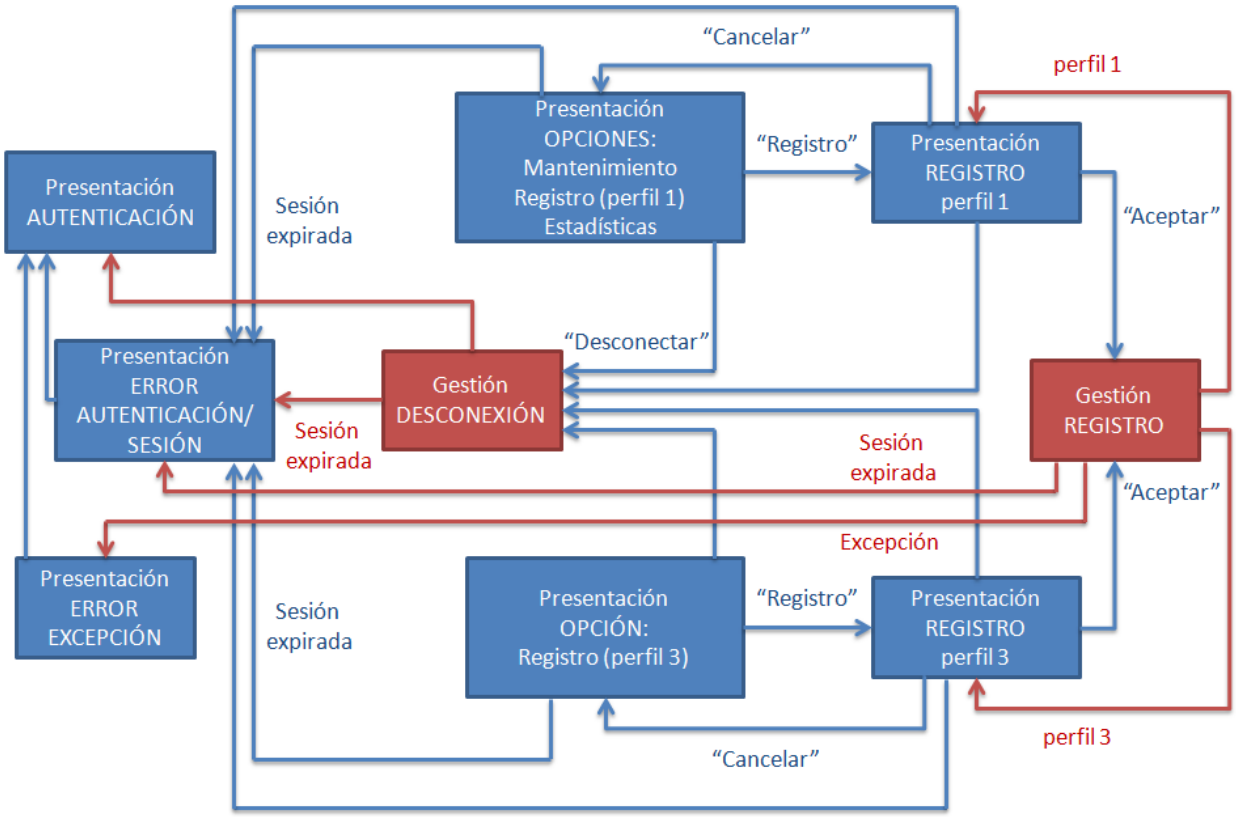


Figura 3.7 Gestión y Presentación. Registro

❖ Registro.

Cuando el usuario selecciona el botón de "Registro", se cede el control al elemento de presentación "Registro". Como se puede ver en la *Figura 3.7*, existen dos elementos de presentación diferentes, en función del perfil del usuario que quiera realizar este registro, que puede ser un operador corriente, o un operador con funciones también de administrador.

Para recoger los datos introducidos por el usuario en el registro y almacenar los correspondientes datos en la base de datos, se hace uso del elemento de "Registro" del bloque de gestión, que recibe y gestiona las peticiones de todos los operadores.

❖ Gestión de errores.

Como se ha podido observar en todas las figuras de esta sección, en las fases anteriores de la aplicación aparecen estos elementos de gestión de errores.

Si en el bloque de Gestión de la Autenticación no se encuentra un usuario que coincida con el introducido en el sistema, toma el control de la ejecución el bloque de Presentación "Error Autenticación/Sesión". También lo hace cuando desde cualquier otro bloque de Gestión o Presentación se detecta que ha expirado la sesión.

El otro bloque de gestión de errores está formado por el elemento Presentación "Error Excepción", al cual se le cede el control de la aplicación cuando se produce un error en un acceso a la base de datos.

En cada caso de los anteriores se presentará al usuario el mensaje de error adecuado, devolviendo el control de la ejecución al bloque de Presentación "Autenticación".

❖ Desconexión.

Esta fase de la aplicación, al igual que la anterior, se ha incluido en todas las figuras de esta sección.

La razón de que también aparezca en las fases anteriores es porque desde cualquier página de la aplicación, el usuario puede seleccionar la opción de desconexión del sistema. En este caso, toma el control el bloque de Gestión “Desconexión”, donde se invalida la sesión y se redirige al usuario al inicio de la aplicación, de forma que pueda iniciar una nueva sesión.

3.4. TECNOLOGÍA Y VERSIONES EMPLEADAS

A continuación se indican las versiones utilizadas de las diferentes tecnologías involucradas en el desarrollo del proyecto.

- Entorno de desarrollo integrado: Eclipse SDK Version: 3.2.2
- Base de datos: MySQL 5.0.37-community-nt
- Servidor web: Apache Tomcat v5.5 [Apache Tomcat v5.5]
- Compilador de Java: jdk1.6.0_12

4. IMPLEMENTACIÓN

Una vez descrito el diseño del sistema, en este apartado se procede a detallar el proceso de implementación de la aplicación. Para ello, se sigue el mismo orden en el esquema de bloques de la arquitectura del sistema que en la explicación del diseño de la estructura de la aplicación.

Por último, tras detallar la implementación de los bloques, se especifica la política de accesos a las diferentes páginas de la aplicación web en función del perfil del usuario que esté registrado.

4.1. ARQUITECTURA DEL SISTEMA

Siguiendo el orden indicado, se aborda en primer lugar la implementación de los módulos de estado y almacenamiento de datos, detallando también la interacción entre ellos, y a continuación los módulos de presentación y gestión.

4.1.1. Estado y Almacenamiento de información

4.1.1.1. *Contenido de la base de datos*

Según el diseño planteado, se han creado en la base de datos del sistema las tablas para implementar tanto la organización de los usuarios del sistema, como la información relacionada con las consultas de los operadores. A continuación se detalla el contenido de las tablas creadas.

Todas las tablas, a excepción de dos de ellas que se indicarán explícitamente, contienen los siguientes campos:

Nombre.

Se trata de una cadena de caracteres que permite definir un nombre para la entrada. En el caso de las tablas dinámicas el nombre se inserta en la tabla cuando la entrada es creada por un usuario de la aplicación, y para las tablas con contenido estático se establece el nombre durante la implementación de la tabla.

Identificador.

Este campo es un índice numérico que se asignará de manera automática comenzando en 1 en el caso de la primera entrada de una tabla y aumentando en una unidad para entradas sucesivas posteriores.

Esto se implementa en la creación de la tabla, especificando que el identificador es un entero sin signo, no nulo, con la capacidad de autoincremento.

Estado.

El estado está representado por un carácter. Para las entradas en uso de las tablas del sistema este carácter representa que están en estado activo. Solamente en caso de ser eliminada una entrada de la tabla, en lugar de eliminarse dicha entrada, se utiliza este campo para indicar mediante el carácter correspondiente que la entrada no está en uso.

Esta acción afecta a aquellas tablas cuyo contenido se puede ver modificado como resultado de las acciones llevadas a cabo por los administradores del sistema. Los administradores tienen la opción de dar de alta o de baja del sistema a los operadores que tienen acceso a la herramienta, así como las categorías y temas a consultar por dichos operadores.

Por ejemplo, para la entrada correspondiente a un operador recién añadido al sistema este parámetro indica que está activo y si se da de baja al operador, el estado pasa a indicar que está inactivo. Si se vuelve a dar de alta a este operador, en lugar de crearse una nueva entrada, se cambia este parámetro de estado inactivo a activo.

Fecha de la creación y fecha de la última modificación.

Estos parámetros se obtienen mediante el código Java de la aplicación, a través de un método que devuelve la fecha y hora actuales del sistema. Esta acción se lleva a cabo en las clases que acceden a la base de datos, y se realiza a la hora de insertar, eliminar o modificar el contenido de una entrada de una tabla. La fecha de la última modificación será diferente a la fecha de creación en aquellos casos en los que la entrada sea eliminada, pasando su estado a inactivo, o si es recuperada más adelante, estableciéndose la fecha y hora del último acceso.

Además de estos parámetros, algunas tablas contienen campos específicos exclusivos. A continuación se indican las tablas creadas y los campos implementados en cada caso, especificando para cada una de ellas el nombre de los campos y el tipo de elemento contenido en cada uno.

- *perfil*

  La tabla *perfil* contiene únicamente los campos comunes descritos previamente.

perfil	
PERF_IDPERFIL	int(10) unsigned
PERF_NOMBRE	varchar(45)
PERF_ESTADO	char(1)
PERF_FECHA_CREACION	datetime
PERF_FECHA_ULT_MODIF	datetime

Tabla 4.1 Elementos de la tabla perfil.

  La clave primaria o *primary key* [55] es el identificador perfil, “PERFIL_IDPERFIL”.

- *usuario*

  La aplicación almacena los datos de los usuarios que acceden al sistema en esta tabla.

  Además de los campos comunes, contiene los siguientes campos: el nombre de usuario y contraseña, datos personales como su nombre y apellidos y su NIF, y el perfil del usuario. Adicionalmente, se indica el nombre del usuario con perfil de administrador que lo ha añadido al sistema.

usuario	
USUA_IDUSUARIO	int(10) unsigned
USUA_LOGIN	varchar(45)
USUA_PASSWORD	varchar(45)
USUA_NOMBRES_APELLIDOS	varchar(80)
USUA_ESTADO	char(1)
USUA_FECHA_CREACION	datetime
USUA_FECHA_ULT_MODIF	datetime
USUA_IDPERFIL	int(10) unsigned
USUA_NIF	varchar(20)
USUA_LOGINUSUARIO	varchar(45)

Tabla 4.2 Elementos de la tabla usuario.

La clave primaria está compuesta por la combinación de las claves identificador de usuario "USUA_ISUSUARIO" y nombre de usuario "USUA_LOGIN".

- *usuario-puesto*

  En la creación de un usuario de tipo operador se indican los puestos que puede ocupar. Estas posibles asociaciones quedan registradas en la tabla *usuario-puesto*, pensada para tal efecto.

  Dichos vínculos se registran por medio de sus identificadores, generándose además un identificador de asociaciones. Este identificador es el único de los campos comunes que se incluye en la tabla *usuario-puesto*.

usuario-puesto	
UP_IDUSUARIO	int(10) unsigned
UP_IDPUESTO	int(10) unsigned
UP_IDUP	int(10) unsigned

Tabla 4.3 Elementos de la tabla usuario-puesto.

La clave primaria es el identificador de la asociación usuario-puesto, definido como "UP_IDUP".

Como se especifica en el diseño de la base de datos, se definen las siguientes tablas, representando cada una un nivel de la organización, al que pertenecen los operadores:

- *puesto*

puesto	
PUES_IDPUESTO	int(10) unsigned
PUES_NOMBRE	varchar(45)
PUES_ESTADO	char(1)
PUES_FECHA_CREACION	datetime
PUES_FECHA_ULT_MODIF	datetime
PUES_IDGRUPO	int(10) unsigned

Tabla 4.4 Elementos de la tabla puesto.

En la tabla *puesto*, la clave primaria es el identificador de puesto, "PUES_IDPUESTO". Se ha definido como clave externa o *foreign key* [21] el identificador de grupo, "PUES_IDGRUPO", que hace referencia a la clave primaria de la tabla *grupo*, definida a continuación.

- *grupo*

grupo	
GRUP_IDGRUPO	int(10) unsigned
GRUP_NOMBRE	varchar(45)
GRUP_ESTADO	char(1)
GRUP_FECHA_CREACION	datetime
GRUP_FECHA_ULT_MODIF	datetime
GRUP_IDOFICINA	int(10) unsigned

Tabla 4.5 Elementos de la tabla grupo.

En el caso de la tabla *grupo*, la clave primaria es el identificador de grupo, en este caso "GRUP_IDGRUPO". También se ha definido la clave externa "GRUP_IDOFICINA", el identificador de oficina. Éste hace referencia a la clave primaria de la tabla *oficina*, que se muestra en el siguiente punto.

- *oficina*

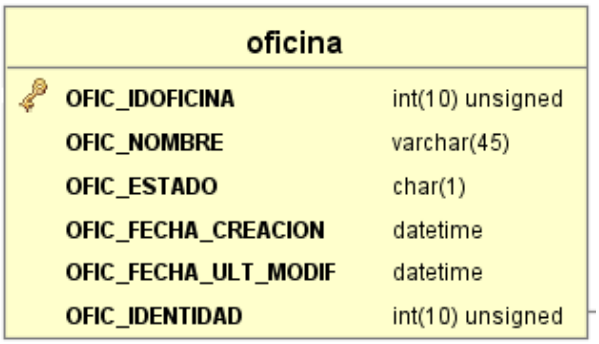


Tabla 4.6 Elementos de la tabla oficina.

La clave primaria de la tabla es el identificador de oficina, "OFIC_IDOFICINA" y la clave externa es el identificador de entidad, "OFI_IDENTIDAD", que hace referencia a la clave primaria de la tabla *grupo*, definida a continuación.

- *entidad*

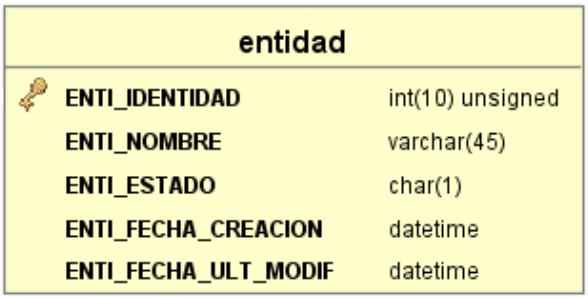


Tabla 4.7 Elementos de la tabla entidad.

En la tabla *entidad*, la clave primaria es el identificador de entidad, "ENTI_IDENTIDAD".

Según se ha diseñado esta estructura de las tablas *puesto*, *grupo*, *oficina* y *entidad*, la tabla de cada nivel contiene, además de los campos comunes a las otras tablas del sistema, un campo definido como clave externa, en el que se indica el nivel superior al que está asociada cada entrada. Con esto se consigue que asignando un puesto a un usuario, el grupo, oficina, y entidad a los que pertenece puedan ser consultados mediante consultas a las tablas de niveles superiores.

Este vínculo está presente en todos los niveles, hasta llegar al nivel superior, el de entidad.

Estas asociaciones de campos implementadas mediante estas claves externas quedan representadas en la *Figura 4.1*, donde se indican los parámetros asociados de una tabla y la siguiente.

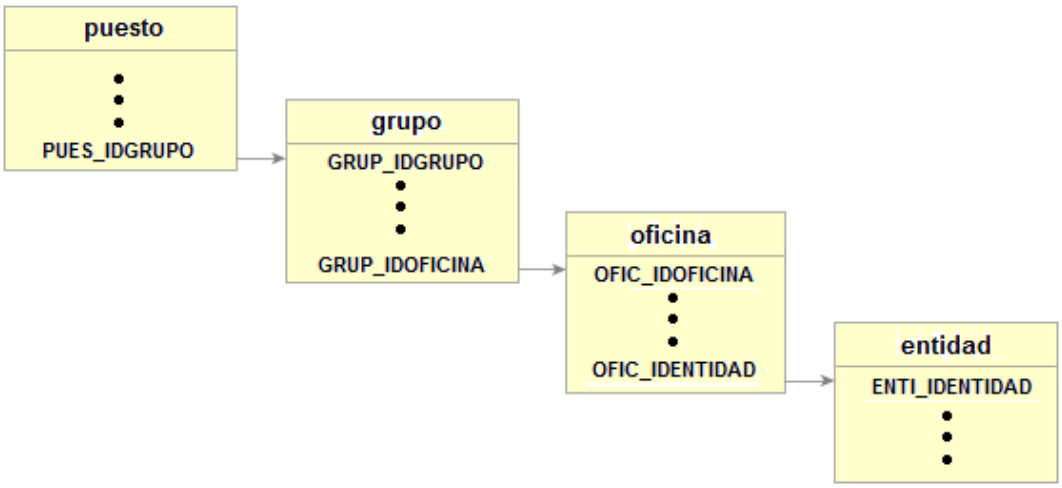


Figura 4.1 Asociación de campos mediante claves externas

Para almacenar en la base de datos la información relacionada con los tipos de asunto, categoría y tema, se crean las tablas correspondientes, y se rellenan con la información adecuada para cada caso.

- *asunto*

Esta tabla es estática, se establecen los dos tipos de asunto durante la implementación, y contiene únicamente los campos comunes.

asunto	
ASUN_IDASUNTO	int(10) unsigned
ASUN_NOMBRE	varchar(45)
ASUN_ESTADO	char(1)
ASUN_FECHA_CREACION	datetime
ASUN_FECHA_ULT_MODIF	datetime

Tabla 4.8 Elementos de la tabla asunto.

La clave primaria de la tabla *asunto* es el identificador de asunto, “ASUN_IDASUNTO”.

- *categoría*

Además de los campos comunes a las demás tablas, ésta contiene el nombre del usuario que crea la categoría.

categoria	
CATE_IDCATEGORIA	int(10) unsigned
CATE_NOMBRE	varchar(60)
CATE_ESTADO	char(1)
CATE_FECHA_CREACION	datetime
CATE_FECHA_ULT_MODIF	datetime
CATE_LOGINUSUARIO	varchar(45)

Tabla 4.9. Elementos de la tabla categoria.

En la tabla *categoria* se ha definido como clave primaria el identificador de categoría, "CATE_IDCATEGORIA".

- *tema*

Esta tabla contiene un campo para indicar la categoría a la que está asociado el tema.

Al igual que en el caso de las categorías, la tabla *tema* contiene el nombre del usuario que crea el tema.

Por último, además de los campos comunes a otras tablas, esta tabla contiene un campo para indicar la URL donde está alojada la información que se debe mostrar al seleccionar el tema.

tema	
TEMA_IDTEMA	int(10) unsigned
TEMA_NOMBRE	varchar(60)
TEMA_ESTADO	char(1)
TEMA_FECHA_CREACION	datetime
TEMA_FECHA_ULT_MODIF	datetime
TEMA_IDCATEGORIA	int(10) unsigned
TEMA_LOGINUSUARIO	varchar(45)
TEMA_URL	varchar(180)

Tabla 4.10 Elementos de la tabla tema.

La clave primaria de la tabla *tema* es el identificador de tema, definida como "TEMA_IDTEMA", y también se ha definido como clave externa el identificador de categoría, "TEMA_IDCATEGORIA", que hace referencia a la clave primaria de la tabla *categoria*, indicada en la *Tabla 4.9*. De esta manera, en el campo "TEMA_IDCATEGORIA" solo podrá haber valores contenidos en el campo "CATE_IDCATEGORIA" de la tabla *categoria*.

Según se ha diseñado la estructura de la base de datos, en las siguientes tablas se almacenan los datos relacionados con los informes llevados a cabo por los operadores.

- *informe*

Además de los parámetros comunes a las otras tablas, ésta almacena la información relacionada con el informe que realiza el operador al atender a una llamada.

La información contenida en esta tabla se detalla a continuación.

- Instante en el que se inicia el registro. Corresponde al momento en el que se abre la página de Registro. Este valor se muestra inicialmente en el campo "Hora inicio", hasta que se selecciona "Inicio llamada" y el contenido del campo es sustituido por ese otro valor.
- Duración de la llamada. Tiempo desde que se selecciona "Inicio llamada" hasta que se selecciona la opción "Enviar".
- Instante en el que se crea el informe. El informe se crea cuando se selecciona la opción "Enviar". Se calcula sumando al instante en el que se selecciona "Inicio llamada" la duración de la llamada.
- Identificador y puesto de usuario, que corresponden al identificador y al puesto del operador que está llevando a cabo el registro.
- Datos del ciudadano:
    - Nombre y apellidos.
    - NIF.

- Correo electrónico.
- Observaciones.

informe	
INFO_ID_INFORME	int
INFO_HORA_INICIO	datetime
INFO_TIEMPO	int
INFO_OBSERVACIONES	varchar(500)
INFO_ESTADO	char(1)
INFO_FECHA_CREACION	datetime
INFO_FECHA_MODIFICACION	datetime
INFO_ID_USUARIO	int
INFO_NOMBRE_APELLIDOS	varchar(80)
INFO_NIF	varchar(20)
INFO_EMAIL	varchar(80)
INFO_USUPUESTO	int(10) unsigned

Tabla 4.11. Elementos de la tabla informe.

La clave primaria de la tabla *informe* es el identificador de informe, “INFO_IDINFORME”. Se ha definido la clave externa “INFO_USUPUESTO”, que hace referencia a la clave primaria de la tabla *puesto*, mostrada en la *Tabla 4.4*. De esta forma se restringe contenido del campo “INFO_USUPUESTO” a valores que pertenezcan al campo “PUES_IDPUESTO” de la tabla *puesto*.

Además de la tabla *informe*, se utiliza la siguiente tabla para almacenar las consultas realizadas en cada informe.

- *consulta*

Esta tabla registra las asociaciones de cada consulta con el informe en el que se ha realizado. Por otro lado, se encarga también de almacenar los parámetros relativos a la consulta, representados también por medio de sus identificadores.

Los campos implementados se especifican a continuación.

- El identificador de la consulta y el del informe al que pertenece, generando la asociación previamente mencionada.
- El identificador del asunto al que hace referencia la consulta
- La categoría sobre la que se ha realizado la consulta. Está representada por su identificador, o por el valor 0 en caso de que el asunto sea una Sugerencia y no se requiera la selección de este campo.
- El tema que se ha consultado, que al igual que la categoría está representado por su identificador, o por el valor 0 en caso de que no haya sido consultado un tema. Esta situación se da en el caso de que el asunto sea una Sugerencia, o si es una Consulta de información en la que el usuario selecciona una categoría pero no un tema asociado a ella.

Como se puede apreciar, de los campos comunes a las demás tablas, la tabla *consulta* solamente necesita el identificador.

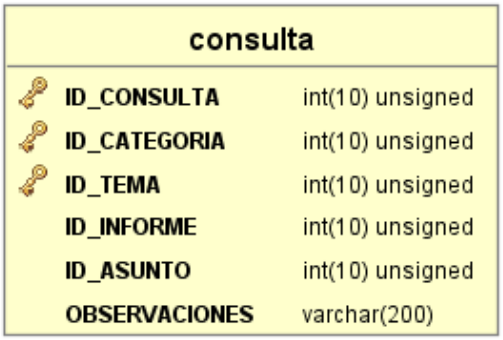


Tabla 4.12 Elementos de la tabla consulta.

La clave primaria de la tabla *consulta* está compuesta por la asociación de las claves identificador de consulta "ID_CONSULTA", identificador de categoría "ID_CATEGORIA" e identificador de tema "ID_TEMA".

4.1.1.2. *Componentes de estado y capa de acceso a datos*

– Componentes de estado.

En función del contenido de la base de datos, se definen las clases Java necesarias para representar los elementos del dominio, resultando las siguientes:

- `Asunto`
- `Categoria`
- `Consultas`
- `Puesto`
- `Informe`
- `Tema`
- `Usuario`
- `VistaObject`
- `DateTime`

Como se puede apreciar, no es necesario definir una clase Java por cada tabla contenida en la base de datos. A continuación, en la *Tabla 4.13*, se detallan los componentes de estado que representan el contenido de la base de datos, especificando las tablas que engloba cada una de las clases.

CLASE JAVA	TABLAS SQL
`Asunto`	`asunto`
`Categoria`	`categoria`
`Consultas`	`consulta`
`Puesto`	`entidad, grupo, oficina, puesto`
`Informe`	`informe`
`Tema`	`tema`
`Usuario`	`usuario, usuario-puesto, perfil`

Tabla 4.13 Representación: Componente de Estado – Tablas en Base de datos

Para las clases que representan directamente una tabla, las columnas de la tabla se traducen a las clases Java como los atributos de clase. Además, algunas de las tablas pasan directamente a conformar atributos de clases que las engloban.

Adicionalmente se definen dos últimas clases, que aportan a los componentes de estado las capacidades necesarias para la representación de los elementos del dominio:

- `VistaObject`
- `DateTime`

La clase `VistaObject` define un objeto que contiene los elementos necesarios para la representación de las estadísticas.

Por otro lado, `DateTime` proporciona las variables y métodos necesarios para definir la fecha y hora del sistema, en diferentes formatos, en cada momento. Estos datos se almacenan en el instante de creación y de modificación de las entradas en la base de datos, en los campos correspondientes destinados a registrar la fecha de creación y la fecha de la última modificación.

– Capa de acceso a datos.

Según el diseño planificado para la implementación del acceso a datos, la clase principal que representa un DAO genérico se define con el nombre de `AbstractDAO`. En ella se describen todos los componentes de esta capa de acceso a datos, declarando todas las variables y métodos necesarios para llevar a cabo dicho acceso.

Los métodos genéricos de acceso para cualquier DAO particular, independientes del protocolo de acceso a la base de datos, se implementan en esta clase `AbstractDAO` y los específicos de cada DAO concreto, que dependerán de este protocolo, son implementados por las clases `DAO`, que heredan de `AbstractDAO`.

Se define una clase DAO por cada componente de estado que tiene su representación en la base de datos del sistema.

COMPONENTE DE ESTADO	CLASE DAO
Asunto	AsuntoDAO
Categoria	CategoriaDAO
Consultas	ConsultaDAO
Puesto	PuestoDAO
Informe	InformeDAO
Tema	TemaDAO
Usuario	UsuarioDAO
VistaObject	VistaDAO

Tabla 4.14 Equivalencia: Componente de Estado – Clase DAO

La clase `VistaObject`no tiene representación en la base de datos. Es un componente de estado que se utiliza para la representación de las estadísticas, accediendo para ello a diferentes tablas (*informe* y *consulta*) en la base de datos.

A continuación se expone una descripción detallada de los elementos declarados en la clase `AbstractDAO`. En la *Figura 4.2* se detallan los componentes que van a ser descritos, representando en qué clase son instanciados. Las cajas de color rojo representan los componentes dependientes del protocolo. En el caso de las clases DAO, se trata de clases Java que contienen variables y métodos que utilizan código MySQL para realizar los accesos a la base de datos de la aplicación. El resto de cajas de color azul contienen clases Java, independientes del protocolo de acceso a la base de datos.

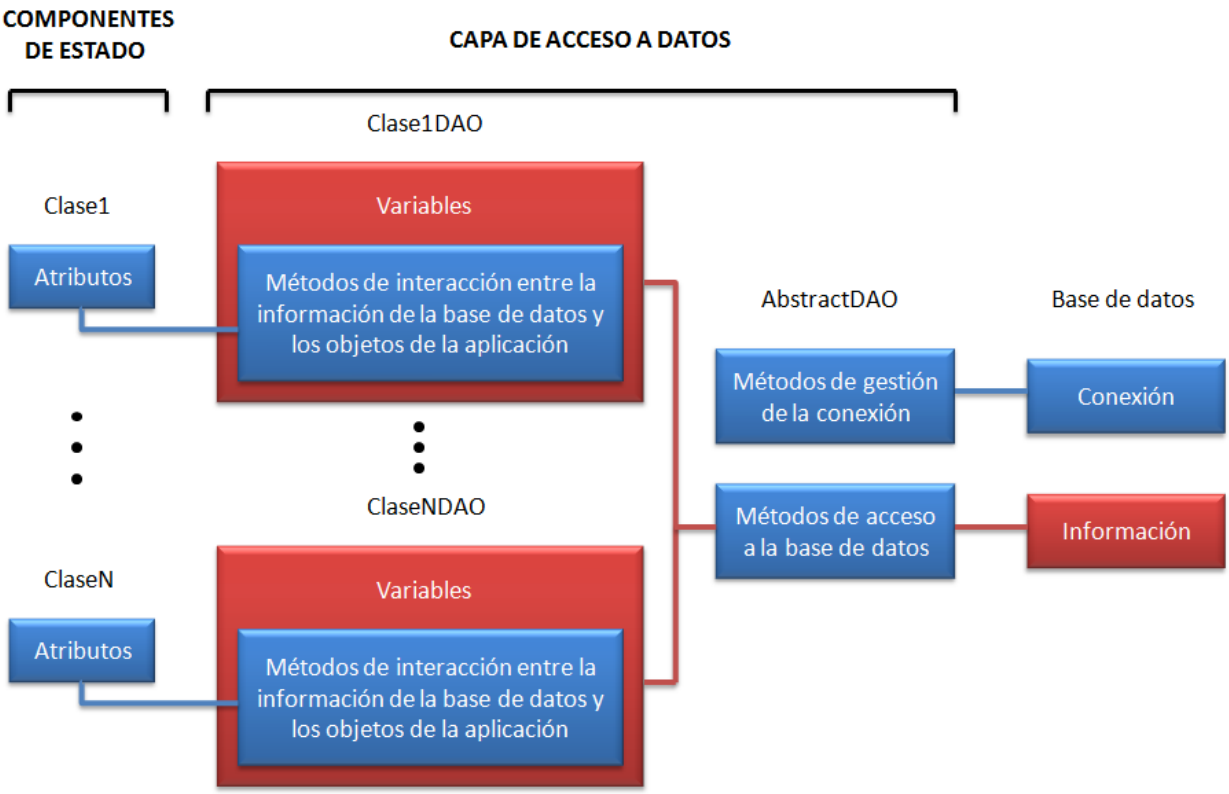


Figura 4.2 Esquema estado y datos del sistema.

Variables:

Las variables declaradas en la clase AbstractDAO son de tipo String, y cada una de las clases DAO las hereda y se encarga de instanciarlas, en función del contenido de la tabla correspondiente a cada clase en la base de datos.

```
String query_insertar;
String query_modificar;
String query_eliminar;
String query_cargar_objeto;
String query_cargar_busqueda;
String query_cargar_todos;
String query_buscar_componentes;
String query_cargar_all;
String query_obtener_id;
```

Estas cadenas de caracteres son sentencias SQL denominadas sentencias preparadas [59]. Proporcionan la posibilidad de establecer la sentencia una vez, y después ejecutarla muchas veces con parámetros diferentes.

Cada uno de estos Strings contiene una sentencia SQL, que además de directivas SQL, se compone de ciertas incógnitas que quedan pendientes de indicar, llamadas parámetros de sustitución y representadas por símbolos de interrogación. Cuando la sentencia vaya a ser ejecutada, es necesario especificar su valor.

Con ello se proporciona a la aplicación la posibilidad de disponer de unas estructuras de sentencia que pueden ser reutilizadas en los numerosos accesos a la base de datos que realicen el mismo tipo de acceso.

Estas sentencias mejoran el rendimiento de la aplicación, ya que solo es necesario analizar la sentencia una única vez. Cuando se prepara inicialmente la sentencia, MySQL la analiza para comprobar la sintaxis y establece la petición para ser ejecutada. Por tanto, todas las veces que se ejecute la sentencia, estas acciones no se volverán a llevar a cabo de nuevo. Este preanálisis aporta un incremento en la velocidad del acceso a la base de datos, sobre todo en el caso de aquellas sentencias más utilizadas.

Métodos de interacción entre los objetos de la aplicación y la información de la base de datos

Como se puede apreciar en la *Figura 4.2*, estos métodos sirven de herramienta a los métodos de acceso a la base de datos, cuyo comportamiento se detalla más adelante.

Se definen en cada una de las clases DAO, y hacen la función de "traductores" de información de las tablas en base de datos a los objetos pertenecientes a los componentes de estado del sistema.

Dentro de esta categoría, los métodos se pueden dividir en dos tipos:

- Métodos de preparación de la petición

  Preparan sentencias MySQL para el acceso a la base de datos, en base a los componentes de estado del sistema.

```
void prepararSentenciaInsercion(PreparedStatement pstm,Object obj)

void prepararSentenciaModificacion(PreparedStatement pstm,Object key)

void prepararSentenciaEliminacion(PreparedStatement pstm,Object key)

void prepararSentenciaCarga(PreparedStatement pstm, Object key)

void prepararSentenciaCarga(PreparedStatement pstm, Object key1, Object
key2)

void prepararSentenciaBuscarId(PreparedStatement pstm, Object obj)
```

  El parámetro que se recibe de tipo `PreparedStatement` [41] representa la sentencia SQL precompilada.

  Como se detalla más adelante, en la fase de Preparación de la sentencia precompilada en el acceso a la base de datos, las sentencias precompiladas de tipo `PreparedStatement` se crean a partir de las variables de la clase previamente descritas, las sentencias preparadas SQL, acarreando las incógnitas provenientes de dichas sentencias.

  Para establecer el valor de estas incógnitas, los métodos de preparación de la petición utilizan los atributos de los objetos recibidos como segundo parámetro, componentes de estado de la aplicación.

- Métodos de carga de información

  Almacenan en los componentes de estado del sistema la información obtenida de las consultas a la base de datos.

  Estos métodos son de utilidad para los métodos de acceso a la base de datos, concretamente para los métodos de búsqueda.

```
Object cargarAll(ResultSet rs)

Object cargar(ResultSet rs)

Object cargarTodos(ResultSet rs)

Object cargarsinComponentes(ResultSet rs)

Object cargarTodosObjetos(ResultSet rs)

int cargarId(ResultSet rs)
```

  El resultado de una búsqueda en la base de datos se almacena en un objeto que implementa la interfaz `ResultSet` [42].

Dicho objeto consiste en una tabla que representa un conjunto de resultados de la base de datos. Este objeto `ResultSet` mantiene un cursor, que apunta a la fila de datos en la que actualmente se encuentra. Inicialmente apunta a la primera fila y proporciona los mecanismos necesarios para recorrer la tabla hasta la última fila.

Los métodos de carga reciben por parámetro uno de estos objetos que implementa la interfaz `ResultSet`, obtenido tras la consulta a la base de datos.

La interfaz `ResultSet` procura los métodos `getter` para obtener los valores de las columnas de la tabla correspondientes a la fila en la que en ese momento se encuentre posicionado el cursor. El driver de JDBC convierte los datos obtenidos a los tipos de datos básicos de Java, en función del método `getter` utilizado.

El método de carga, a partir de los datos obtenidos de la base de datos, crea una instancia del componente de estado correspondiente a la tabla consultada.

Métodos de gestión de la conexión:

A continuación se detalla el comportamiento de cada uno de los métodos implementados para llevar a cabo esta funcionalidad.

- `Connection obtenerConexion()`

Este método obtiene un objeto que implementa la interfaz `java.sql.Connection` con el que acceder a la base de datos. Para ello sigue los siguientes pasos.

Carga el driver "`com.mysql.jdbc.Driver`" mediante la declaración habitual usada para conectores jdbc (`Class.forName("com.mysql.jdbc.Driver"))`.

No es necesario crear un ejemplar de un driver y registrarlo con el `DriverManager`, ya que la llamada a `Class.forName` lo hace automáticamente.

Una vez cargado el driver, es posible establecer una conexión con un controlador de base de datos.

Para obtener la conexión, se invoca al método `DriverManager.getConnection()`, que seleccionará el driver de los registrados que pueda manejar la conexión descrita en los parámetros que recibe el método y se devolverá el objeto que implementa la interfaz `java.sql.Connection`:

```
Connection con = DriverManager.getConnection(
"jdbc:mysql://localhost/cantabriaPFC", "root", "root" );
```

Al utilizar la base de datos de MySQL, la URL empieza por: `jdbc:mysql:`, y va seguida por la localización de la base de datos: `//localhost/cantabriaPFC`. Los siguientes parámetros son el nombre y la contraseña utilizados para entrar en el controlador de la base de datos.

La conexión devuelta por el método `DriverManager.getConnection` es una conexión abierta que se puede utilizar para crear sentencias JDBC que pasen nuestras sentencias SQL al controlador de la base de datos.

Cuando se crea una conexión, está en modo auto-entrega o *autocommit* [26]. Esto significa que cada sentencia SQL individual es tratada como una transacción y será automáticamente entregada justo después de ser ejecutada.

La forma de permitir que dos o más sentencias sean agrupadas en una transacción es desactivar el modo auto-entrega.

Esto se realiza mediante el siguiente código, donde `con` es una conexión activa:

```
con.setAutoCommit(false);
```

Una vez que se ha desactivado la auto-entrega, no se entregará ninguna sentencia SQL hasta que se llame explícitamente al método commit. Todas las sentencias ejecutadas después de la anterior llamada al método commit serán incluidas en la transacción actual y serán entregadas juntas como una unidad.

- `void desconectar(Connection con)`

Este método sirve para finalizar la conexión con la base de datos.

Antes de eliminar la conexión, se activa de nuevo el modo auto-commit. De esta forma, se evitan conflictos con otros usuarios:

```
con.setAutoCommit(true);
```

Para cerrar la conexión con la base de datos se utiliza la sentencia:

```
con.close();
```

- `void registrarTransaccion(Connection con)`

Al estar desactivado el modo Auto-entrega, es necesario registrar manualmente el fin de una transacción. Para ello, como se ha indicado previamente, se ejecuta la sentencia:

```
con.commit();
```

Estos métodos de gestión de la conexión van a proporcionar a los métodos de acceso a la base de datos una forma rápida y eficiente de trabajar con las conexiones.

Métodos de acceso a la base de datos:

Dentro de este grupo de métodos se puede hacer una distinción en dos subgrupos, en función del tipo de acceso a la base de datos:

A. Métodos de búsqueda.
   Son aquellos métodos que devuelven información sobre una consulta en la base de datos.
B. Métodos de manipulación.
   Este conjunto de métodos tiene como función modificar el contenido de la base de datos.

Para realizar los accesos a la base de datos, los métodos de ambos tipos siguen el procedimiento descrito a continuación:

1. Creación de la conexión.

En primer lugar se crea la conexión, utilizando el método de gestión de la conexión anteriormente descrito `obtenerConexion()`.

2. Preparación de la sentencia precompilada.

```
PreparedStatement pstm = con.prepareStatement(query_insertar);
```

El objeto `Connection` crea, mediante la llamada al método `prepareStatement` [54], el objeto de tipo `PreparedStatement`.

El `String` que se pasa por parámetro es una de las variables de la clase, utilizadas para el acceso a la base de datos, y contiene una sentencia SQL preparada, a partir de la cual se genera el objeto `PreparedStatement`.

En el caso particular del ejemplo la acción es una inserción, y por ello el `String` es: `query_insertar`.

Estas sentencias preparadas se definen en cada clase DAO, y para cada método de acceso a la base de datos se requiere una sentencia concreta, según se detalla en la *Tabla 4.15*. La clase DAO da valor a las sentencias preparadas según sus necesidades, en función de las acciones que se vayan a realizar en base de datos con las tablas asociadas a cada una de ellas.

Método de acceso a la BD	Sentencia preparada
`int buscarId(Object obj)`	`query_obtener_id`
`Object buscarObjeto(Object key1, Object key2)`	`query_cargar_objeto`
`ArrayList buscarComponentes(Object key1)`	`query_buscar_componentes`
`ArrayList buscarTodos()`	`query_cargar_todos`
`ArrayList buscarTodossinComponentes()`	`query_cargar_todos`
`ArrayList buscarTodosObjetos()`	`query_cargar_todos`
`ArrayList buscarAll()`	`query_cargar_all`
`int insertar(Connection con, Object obj)`	`query_insertar`
`boolean eliminar(Connection con, Object key)`	`query_eliminar`
`boolean modificar(Connection con, Object key)`	`query_modificar`

Tabla 4.15 Equivalencia: Método de acceso – Sentencia preparada

El objeto `PreparedStatement` creado representa la sentencia SQL precompilada. Se denomina sentencia precompilada porque contiene las incógnitas provenientes de la sentencia SQL preparada, los parámetros de entrada cuyos valores no se determinan a la hora de crear la sentencia sino antes de ser ésta ejecutada.

3. Preparación de la petición.

Para completar la sentencia SQL precompilada se utilizan los métodos de preparación de la petición descritos previamente.

Estos métodos, definidos en cada una de las clases DAO, establecen el valor de los parámetros de entrada del objeto `PreparedStatement`, utilizando para ello los atributos del objeto que se pasa por parámetro.

La *Tabla 4.16* indica el método de preparación de la petición establecido para a cada método de acceso a la base de datos que lo requiere.

Método de acceso a la BD	Método de preparación de la petición
`int buscarId(Object obj)`	`prepararSentenciaBuscarId`
`Object buscarObjeto(Object key1, Object key2)`	`prepararSentenciaCarga`
`ArrayList buscarComponentes(Object key1)`	`prepararSentenciaCarga`
`int insertar(Connection con, Object obj)`	`prepararSentenciaInsercion`
`boolean eliminar(Connection con, Object key)`	`prepararSentenciaEliminacion`
`boolean modificar(Connection con, Object key)`	`prepararSentenciaModificacion`

Tabla 4.16 Equivalencia: Método de acceso – Método de preparación de la petición

Los métodos de acceso a la base de datos que no solicitan esta preparación de la sentencia, ya que la sentencia que utilizan no contiene incógnitas, se muestran en la *Tabla 4.17.*

Método de acceso a la BD	Sentencia preparada completa
`ArrayList buscarTodos()`	`query_cargar_todos`
`ArrayList buscarTodossinComponentes()`	`query_cargar_todos`
`ArrayList buscarTodosObjetos()`	`query_cargar_todos`
`ArrayList buscarAll()`	`query_cargar_all`

Tabla 4.17 Equivalencia: Método de acceso – Sentencia preparada completa

Al igual que sucede con las sentencias preparadas, cada clase DAO, en base a las acciones que se vayan a realizar con la tabla asociada en la base de datos a cada una de ellas, implementa los métodos de preparación da la petición que se requieran.

4. Ejecución de la petición.

Para ejecutar la sentencia SQL contenida en el objeto `PreparedStatement` [41] se utiliza uno de los siguientes métodos de la interfaz `PreparedStatement`, en función del tipo de método:

A. Métodos de búsqueda.
   Utilizan el método `boolean execute()`, empleado para ejecutar cualquier tipo de sentencia SQL.
B. Métodos de manipulación.
   Utilizan el método `int executeUpdate()`. Este método se utiliza para ejecutar el tipo de sentencias que usan estos métodos de manipulación: `INSERT`, `UPDATE` o `DELETE`.

Tras la ejecución de la petición, el siguiente paso diferirá en función del tipo de método:

5.A. Obtención del resultado de la consulta (Métodos de búsqueda).

Para obtener el resultado de la búsqueda, se utilizan los métodos de carga, diseñados para levar a cabo esta función.

Es necesario implementar un método de carga casi para cada método de búsqueda, ya que se requiere particularizar la carga en función de la búsqueda realizada. En la *Tabla 4.18* se indica el método de carga implementado para cada método de búsqueda.

Método de búsqueda	Método de carga
`int buscarId(Object obj)`	`int cargarId(ResultSet rs)`
`Object buscarObjeto(Object key1, Object key2)`	`Object cargar(ResultSet rs)`
`ArrayList buscarComponentes(Object key1)`	`Object cargarTodos(ResultSet rs)`
`ArrayList buscarTodos()`	`Object cargar(ResultSet rs)`
`ArrayList buscarTodossinComponentes()`	`Object cargarsinComponentes(ResultSet rs)`
`ArrayList buscarTodosObjetos()`	`Object cargarTodosObjetos(ResultSet rs)`
`ArrayList buscarAll()`	`Object cargarAll(ResultSet rs)`

Tabla 4.18 Equivalencia: Método de búsqueda – Método de carga

Los métodos de carga se definen en las clases DAO que vayan a realizar la búsqueda asociada a dicho método, cargando en los atributos del objeto que se devuelve la información contenida en la tabla consultada en base de datos.

El objeto que implementa la interfaz `ResultSet` almacena el resultado de la consulta, y se obtiene mediante el método de la interfaz `PreparedStatement`:

```
Resultset getResultSet()
```

Como se ha avanzado en el apartado relativo a los métodos de carga de información, en la llamada al método de carga se pasa por parámetro un objeto que implementa la interfaz `ResultSet`, obtenido durante la ejecución de la consulta a la base de datos previamente realizada.

El método de carga, descrito previamente, devuelve una instancia del componente de estado que corresponda, creada a partir de los datos obtenidos de la base de datos.

Por ejemplo, una llamada a un método de carga puede ser:

```
Object obj = cargar(pstm.getResultSet());
```

Con esta llamada, en el método de búsqueda se obtiene el objeto creado por el método de carga, y será el objeto que devuelva como resultado.

5.B. Registro del fin de la transacción (Métodos de manipulación).

Se fuerza la ejecución y entrega de las sentencias SQL previas mediante el método `registrarTransaccion(con)`. Es necesario registrar el fin de la transacción en este tipo de métodos, para actualizar el contenido de la base de datos.

6. Finalización de la conexión.

Se finaliza la conexión con la base de datos utilizando el método `desconectar(con)`.

Cada método de búsqueda se ha implementado como una unidad que lleva a cabo en su totalidad los pasos anteriormente descritos.

En el caso de los métodos de manipulación, cada uno de ellos, en lugar de ejecutarse en un bloque, se ha dividido en una pareja de métodos, realizando entre los dos la funcionalidad completa de la acción que implementan, siguiendo los pasos que se han definido.

El primero de los métodos de la pareja tiene una función de gestión, encargándose de proporcionar el entorno requerido para llevar a cabo la acción de manipulación de la base de datos sobre el objeto que recibe por parámetro. Crea la conexión necesaria para el acceso a la base de datos, y tras el acceso, registra la transacción y cierra la conexión.

El segundo método es el que lleva a cabo la acción en la base de datos. Recibe para ello por parámetro el objeto del primer método, y también recibe de éste la conexión que va a utilizar para realizar el acceso.

Ambos métodos devuelven un entero o booleano, que define el éxito o fracaso de la acción, en función del resultado obtenido del segundo método.

Estas parejas de métodos se encargan de gestionar y llevar a cabo la inserción, eliminación y modificación en la base de datos del objeto que reciben por parámetro.

- `int insertar(Object obj)`
- `int insertar(Connection con, Object obj)`

- `boolean eliminar(Object key)`
- `boolean eliminar(Connection con, Object key)`

- `boolean modificar(Object key)`
- `boolean modificar(Connection con, Object key)`

Gracias a la arquitectura de clases que se ha diseñado para los componentes de la capa de acceso a datos, se proporciona un acceso global a la base de datos mediante la clase `AbstractDAO`, utilizando unos métodos de acceso comunes, y estableciendo en cada caso los parámetros adecuados en función de la clase `DAO` que esté en uso.

Como se ha mencionado previamente, cada clase DAO da valor a los parámetros e implementa los métodos, en función de las acciones que se van a realizar en la base de datos con la tabla que corresponde al componente de estado que tiene asociado.

A continuación se indican las acciones en base de datos que necesitará realizar cada componente de estado, por las que se han implementado los métodos correspondientes en cada clase DAO.

- Asunto

  La única acción que tiene sentido llevar a cabo con los tipos de asunto es su consulta, ya que se trata de una tabla cuyo contenido no va a ser modificado durante el uso de la aplicación.

- Categoría y Tema

  Las categorías y temas en base de datos se pueden insertar, modificar, eliminar y consultar, por lo que las clases `CategoriaDAO` y `TemaDAO` implementan los métodos necesarios para que se puedan llevar a cabo estas acciones, además de las búsquedas en la base de datos necesarias en cada caso.

- Informe y Consulta

  Los informes y las consultas se almacenan en la base de datos en las tablas destinada a este propósito, y las acciones que se pueden llevar a cabo son la inserción, eliminación y consulta, por lo que las clases `InformeDAO` y `ConsultaDAO` implementan los siguientes métodos.

  La clase `InformeDAO`, además de los métodos de acceso a la base de datos declara un método no heredado de `AbstractDAO`, `int obtenerLastId()`, que obtiene el identificador del último informe y se utiliza para almacenarlo en la tabla `consulta`, indicando así el informe al que están asociadas las consultas realizadas durante cada informe.

- Puesto

  Los puestos están establecidos en la tabla `puesto`, y no son modificados durante el uso de la herramienta, son consultados, para lo que la clase `PuestoDAO` proporciona los elementos necesarios. Además implementa también las variables y métodos que permiten llevar a cabo determinadas inserciones y modificaciones, que se realizarán sobre la tabla `usuario-puesto`.

- Usuario

  Desarrolla y da valor a todos los métodos y atributos heredados de la clase padre `AbstractDAO`, algunos de los cuales han sido implementados solo en esta clase. Por ejemplo el método `int buscarId(Object obj)`, que obtiene el identificador de un determinado usuario, indicado por parámetro.

- VistaObject

  Esta clase no tiene una tabla asociada en base de datos. `VistaObject` se ha implementado con la finalidad de disponer de un tipo de objeto que contenga los elementos relacionados con la obtención de estadísticas.

  Para llevar a cabo este cometido, `VistaDAO` no necesita implementar los métodos heredados de la clase `AbstractDAO`. Los métodos de la clase `VistaDAO` acceden a diferentes tablas en la base de datos, en función de la estadística que se desea calcular.

En la *Tabla 4.19* se indican los métodos de acceso a la base de datos que utilizará cada componente de estado. En función de los métodos que necesita un componente para acceder a la base de datos, la

clase DAO asociada a dicho componente implementará las propiedades y funciones utilizadas por estos métodos.

Método de acceso a BD\ Componente de Estado	Asunto	Categoría	Consulta	Puesto	Informe	Tema	Usuario
buscarId	NO	NO	NO	NO	NO	NO	SI
buscarObjeto	NO	NO	NO	NO	NO	NO	SI
buscarComponentes	NO	SI	SI	NO	SI	NO	SI
buscarTodos()	SI	SI	SI	SI	SI	SI	SI
buscarTodossinComponentes()	NO	NO	NO	NO	NO	NO	SI
buscarTodosObjetos()	NO	NO	NO	NO	NO	NO	SI
buscarAll()	NO	SI	NO	NO	NO	SI	SI
insertar	NO	SI	SI	SI	SI	SI	SI
eliminar	NO	SI	SI	SI	SI	SI	SI
modificar	NO	SI	NO	NO	NO	SI	SI

Tabla 4.19 Métodos de acceso a base de datos utilizados por cada componente de estado

Los métodos de búsqueda se implementan según conviene a cada clase DAO, teniendo en cuenta las consultas necesarias de este tipo a la base de datos, y los parámetros requeridos.

Por ejemplo, el método `ArrayList buscarComponentes(Object key1)` se implementa en varias de las clases DAO, en cada una de ellas para realizar una búsqueda diferente. En el caso de la clase `CategoriaDAO`, se utiliza este método para buscar en la tabla `tema` aquellos temas activos cuya categoría asociada coincida con la categoría que se ha pasado por parámetro, obteniéndose el nombre e identificador de cada uno de estos temas. En cambio, en la clase `UsuarioDAO`, este método extrae de la base de datos las siguientes características de un determinado usuario, que se pasa por parámetro:

- el identificador y el nombre del puesto asociado, de la tabla `puesto`.
- el nombre de grupo asociado, de la tabla `grupo`.
- el nombre de entidad asociada, de la tabla `entidad`.
- el nombre de oficina asociada, de la tabla `oficina`.

En cuanto a los métodos de manipulación, tendrán un comportamiento muy similar en cada clase DAO, siempre teniendo en cuenta las particularidades de cada componente de estado y tabla asociada.

Como se ha mencionado, la clase `VistaDAO` no implementa los métodos heredados de `AbstractDAO`.

A continuación se detallan los métodos que incluye esta clase, que servirán para realizar diferentes accesos a la base de datos y obtener así la información necesaria para el cálculo de las estadísticas proporcionadas por el sistema.

En todos los casos se pasa a los métodos los siguientes parámetros:

- `int idpuesto`
- `String fechaDesdeaux`
- `String fechaHastaaux`

A partir de estos parámetros se realizan las búsquedas en base de datos en las tablas que almacenan la información relacionada con los informes y consultas realizados (*informe* y *consulta*), teniendo en cuenta los informes que se hayan efectuado correctamente, en el rango de tiempos establecido.

Las consultas SQL realizadas para el cálculo de los parámetros correspondientes a cada caso utilizan funciones de agrupamiento o *aggregate functions* [22], como son "`COUNT()`" y "`AVG()`".

En cuanto al rango temporal relativo a la realización de las consultas, se tienen en cuenta los informes que han sido iniciados entre las 00:00h del día `fechaDesdeaux` y las 23:59h del día `fechaHastaaux`.

Además, si el administrador ha escogido un puesto, se acota la búsqueda anterior, teniendo en cuenta solamente las acciones llevadas a cabo en dicho puesto. En caso de que el administrador haya escogido obtener el resultado de todos los puestos, se obtienen todas las ocurrencias del rango previamente indicado.

Los métodos que componen la case `VistaDAO` son los siguientes:

- `void listarAsuntos(int idpuesto, String fechaDesdeaux, String fechaHastaaux)`

  Este método guarda en el atributo de la clase una lista con los asuntos consultados, almacenando por cada asunto su identificador, su nombre, y el número de veces que ha sido consultado.

- `void listarCategoriasporAsunto(int idpuesto, int idasunto, String fechaDesdeaux, String fechaHastaaux)`

  El método `listarCategoriasporAsunto` guarda en el atributo de la clase el conjunto de las categorías consultadas, englobando cada una de ellas su identificador, su nombre, el número de veces que ha sido consultada y el asunto correspondiente.

- `void listarTemasporCategorias(int idpuesto, int idasunto, int idcategoria, String fechaDesdeaux,String fechaHastaaux)`

  Este método se utiliza para guardar en el atributo de la clase la lista de los temas consultados. Se almacena por cada tema su identificador, su nombre, y el número de veces que ha sido consultado.

- `void listarTemasTiempoMedio(int idpuesto, String fechaDesdeaux, String fechaHastaaux)`

  El método `listarTemasTiempoMedio` guarda en el atributo de la clase una lista con todos los temas del sistema, almacenando por cada uno su identificador y la media del tiempo en segundos que lleva a los operadores realizar su consulta.

- `void listarOperariosTiempoMedio(int idpuesto, String fechaDesdeaux, String fechaHastaaux)`

  Se utiliza este método para guardar en el atributo de la clase una lista con los operadores del sistema que han realizado algún informe en el periodo de tiempo correspondiente y en el puesto indicado. Se almacena por cada operador su nombre y la media del tiempo en segundos que le lleva realizar las consultas.

- `void listarDiasTiempoMedio(int idpuesto, String fechaDesdeaux, String fechaHastaaux)`

  Este método guarda en el atributo de la clase una lista con los días que abarca el periodo determinado por las dos fechas proporcionadas, ambas incluidas. Se almacena por cada día la fecha correspondiente y la media del tiempo en segundos que ha llevado realizar las consultas ese día.

- `void listarFranjasHorariasTiempoMedio(String rangohorario, int idpuesto, String fechaDesdeaux, String fechaHastaaux)`

  El método `listarFranjasHorariasTiempoMedio` guarda en el atributo de clase una lista con las franjas horarias contenidas en un día, en intervalos de 60 minutos. Almacena por cada franja horaria su etiqueta identificadora y la media del tiempo en segundos que ha llevado realizar las consultas durante ese tiempo.

- `listarDiaSemanaTiempoMedio(int idpuesto, String fechaDesdeaux, String fechaHastaaux)`

  Mediante este método se guarda en el atributo de la clase una lista con los días de la semana, almacenando por cada día la etiqueta correspondiente y la media del tiempo en segundos que ha llevado realizar las consultas ese día de la semana.

4.1.2. Gestión y Presentación

A continuación se describen brevemente las páginas JSP que componen el bloque de presentación de la aplicación, y los servlets cuyo conjunto forma el bloque de gestión.

En el caso de las páginas JSP se detallan el contenido mostrado, en función del estado de la aplicación, almacenado en los atributos de la petición o de la sesión, y los parámetros a través de los cuales el usuario interacciona con la aplicación.

Se ha empleado código JavaScript [39] en la capa de presentación, para implementar la visualización de elementos como por ejemplo la carga previa de las imágenes o para llevar a cabo las validaciones en el lado del cliente de los diferentes formularios de la aplicación. También se emplea JavaScript para mostrar el calendario de selección de fechas mostrado en los formularios, para lo que se ha utilizado la librería CalendarXP [6].

Como es habitual en este tipo de aplicaciones, la apariencia se gestiona a través de las hojas de estilo (*CSS*) [24]. En el caso de esta aplicación, se han utilizado dos hojas de estilo, "estilos.css" y "menu_estilos.css".

En cuanto a los servlets, obtienen la información recibida de las páginas JSP y gestionan las acciones requeridas por la aplicación, utilizando para ello la capa de acceso a datos y los componentes de estado, para almacenar el estado de la aplicación en los atributos correspondientes. Finalmente, devuelven el control de la ejecución a una página JSP, que, tras consultar el estado, genera la presentación para mostrar al usuario.

Para ambos componentes se indica desde qué puntos de la ejecución se puede acceder a cada JSP o servlet, y a qué otros componentes de la aplicación es posible dirigirse desde ellos, indicándolo mediante el gráfico correspondiente en cada caso.

Con ello se detalla paso a paso el flujo de la aplicación, comenzando desde su inicio, y teniendo en cuenta las diferentes posibilidades y caminos que se pueden tomar en el transcurso de la ejecución de la herramienta.

Estos elementos se pueden clasificar en grupos, puesto que pertenecen a las diferentes fases de la aplicación definidas en el diseño.

❖ Autenticación.

login.jsp

La URL de inicio de la aplicación accede directamente a esta página. Se trata de la pantalla de autenticación de usuarios para el acceso a la herramienta.

Contiene un formulario en el que se introducen dos campos, el nombre de usuario y la contraseña.

Una vez introducidos estos campos, se envía el formulario al pulsar el botón "Aceptar" al servlet `ServletLogin`.

Como se puede observar el la *Figura 4.3*, además de acceder a esta página de autenticación al iniciarse la aplicación, también se invoca a login.jsp en los siguientes puntos de la aplicación:

- Si en `ServletLogin` el usuario que es de tipo operador (es decir, usuario operador o usuario administrador y operador) no tiene asignado ningún puesto, se devuelve el control a `login.jsp`. En el servlet se modifica el estado, almacenando en el correspondiente atributo de

la petición que se ha producido este fallo, de manera que en la página de autenticación `login.jsp`, al consultar dicho atributo, se muestra un mensaje de alerta indicando este error.

- Cuando ha habido un error en la autenticación o ha expirado la sesión, la página `errorlogin.jsp` invoca a `login.jsp`.

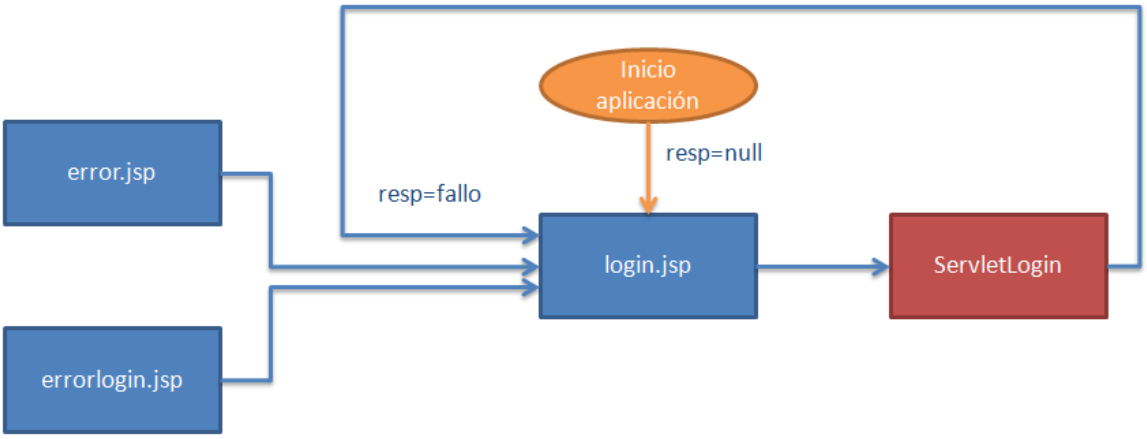


Figura 4.3 Flujo de entrada y salida a login.jsp

ServletLogin.java

La función de este servlet es gestionar el acceso inicial a la aplicación, en función del tipo de usuario que está accediendo y su puesto.

En primer lugar, obtiene los parámetros introducidos por el usuario en el formulario de la página de autenticación, y comprueba en la base de datos de la aplicación que dicho usuario está registrado en la herramienta. En función del resultado de esta comprobación, se llevan a cabo unas u otras acciones, según se indica a continuación, y se ilustra en la *Figura 4.4*.

A. Usuario registrado

En de que esté registrado, se almacena este usuario como atributo de sesión [47], de manera que pueda ser utilizado por todos los componentes de la aplicación, y se inicia la gestión del acceso a la aplicación, teniendo en cuenta los aspectos que se detallan a continuación.

En primer lugar se determina el tipo de usuario que ha accedido a la aplicación, que en función de su perfil tendrá acceso a unas u otras funcionalidades de la herramienta.

Usuario administrador y operador.

Si se trata de un usuario administrador y operador, es necesario conocer el puesto de dicho usuario. Por ello, se consulta el número de puestos asociados a este usuario.

- Si no tiene ningún puesto asociado, esto indica que se ha producido un error. En este caso, se reenvía la petición a `login.jsp`, indicando el fallo mediante el parámetro correspondiente.
- Si el usuario tiene un solo puesto asociado, ya puede acceder a la aplicación. Teniendo en cuenta su perfil, se realiza la petición a `mantenimiento.jsp`.
- Los usuarios con más de un puesto asociado deben seleccionar el puesto que van a utilizar para realizar este acceso a la aplicación. Con este fin se reenvía la petición a `puestos.jsp`.

Usuario operador.

El procedimiento para gestionar el acceso de este tipo de usuario es muy similar al caso del usuario administrador-operador, ya que también puede tener uno o varios puestos.

Tras consultar el número de puestos, se sigue el mismo esquema que se ha descrito en el caso previo, con una única diferencia.

- Si el usuario tiene un solo puesto asociado, accede a la página que corresponde a su perfil mediante el reenvío de la petición a `menuRegistro.jsp`.

Usuario administrador.

Este tipo de usuario no necesita gestión de puestos, por lo que directamente se puede reenviar la petición a `mantenimiento.jsp`.

B. Usuario no registrado

Si los parámetros obtenidos del formulario proveniente de la página de autenticación no coinciden con ningún usuario registrado en la herramienta, se reenvía la petición a la página `errorlogin.jsp`, indicando mediante el atributo correspondiente que los datos de autenticación son incorrectos.

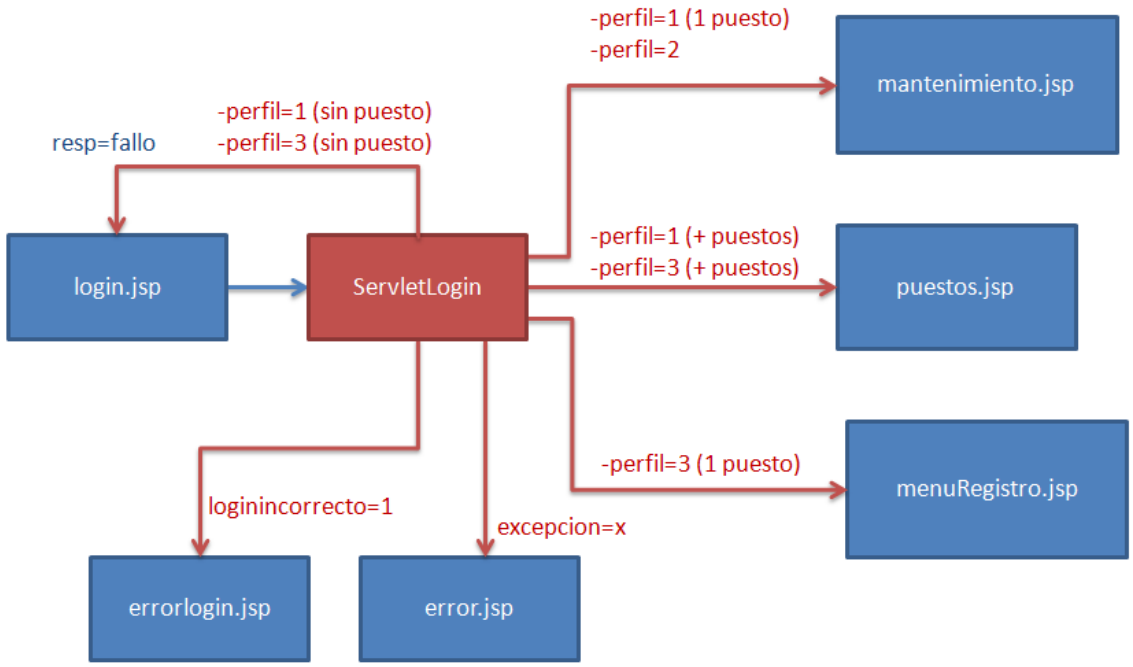


Figura 4.4 Flujo de entrada y salida a ServletLogin.java

❖ Gestión de errores

errorlogin.jsp

Según se ilustra en la *Figura 4.5*, se accede a esta página cuando se ha producido un error de los siguientes tipos.

- Si los parámetros del formulario que recibe el servlet `ServletLogin` de la página de autenticación no coinciden con ningún usuario registrado en la base de datos de la aplicación, dicho servlet reenvía la petición a esta página JSP.
- En caso de que la sesión haya expirado, por haberse superado el límite de tiempo sin haber sido utilizada, se invoca esta página desde todas las demás páginas JSP, y los servlets, exceptuando `ServletLogin`, ya que en este servlet es donde se establece la sesión.

El límite de tiempo en el que una sesión de tipo `HttpSession` [52] permanece activa sin ser utilizada está definido por defecto en 30 minutos. A partir de ese momento, la sesión expira automáticamente.

Existe la posibilidad de modificar dicho límite. Una forma sencilla consiste en establecer en el archivo `web.xml` el valor de dicho parámetro, en minutos, como se muestra en las siguientes líneas de código:

```
<session-config>
<session-timeout>30</session-timeout>
</session-config>
```

En el caso que aquí ocupa se ha decidido mantener el límite por defecto, ya que 30 minutos se considera un periodo más que razonable para un usuario sin utilizar la aplicación.

Seguidamente se indica que ha habido un error en la operación y en cada caso se detalla la descripción del error.

Una vez el usuario pulsa el botón “Aceptar” a continuación del texto informativo anterior, se devuelve el control a la página `login.jsp`, ofreciéndose de nuevo al usuario la autenticación.

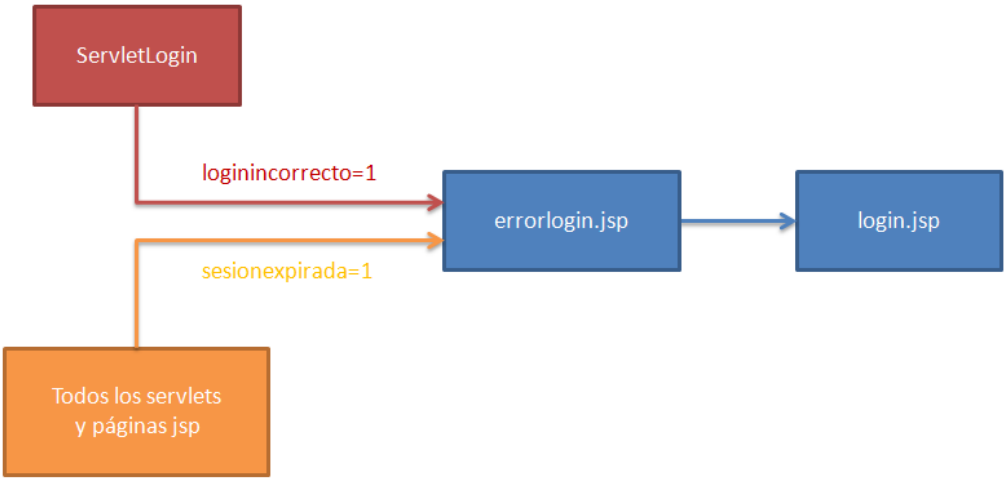


Figura 4.5 Flujo de entrada y salida a errorlogin.java

error.jsp

La aplicación redirige el control a esta página cuando se produce un error en el acceso a la base de datos desde el punto de la aplicación en el que se esté realizando dicho acceso.

Esta página muestra al usuario un mensaje, informándole de que se ha producido dicho error en el acceso a la base de datos.

Al igual que en el caso de `errorlogin.jsp`, como se puede observar en la *Figura 4.6*, cuando se selecciona el botón “Aceptar” que aparece tras el texto anterior, la página `login.jsp` vuelve a tomar el control de la ejecución, ofreciendo al usuario la página de autenticación.

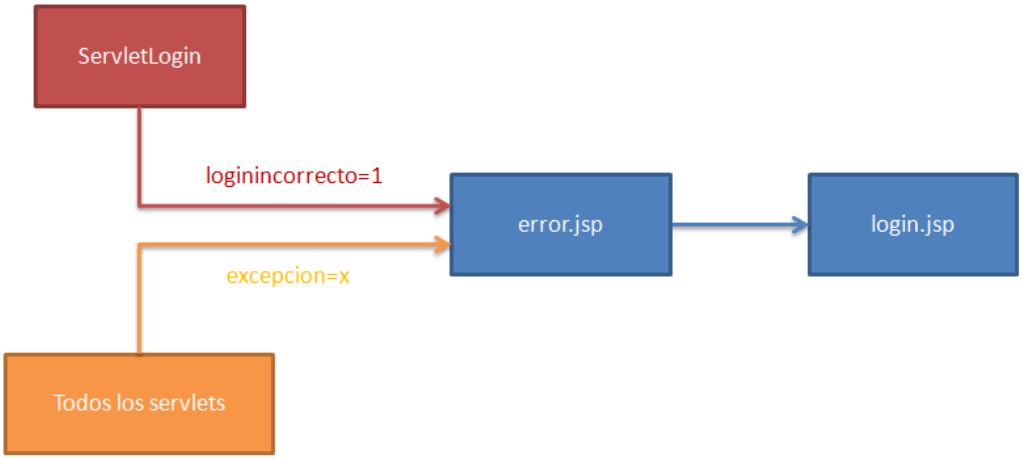


Figura 4.6 Flujo de entrada y salida a error.java

❖ Selección de puesto.

puestos.jsp

Esta página JSP se muestra tras la autenticación a los usuarios con perfil administrador-operador y operador, que pueden tener más de un puesto, para que determinen el puesto que quieren utilizar de entre los que les corresponden.

Se obtiene el atributo usuario de la sesión, y se ofrece un menú desplegable con los puestos que tiene asignados, obtenidos de la base de datos.

Finalmente, se envía el formulario con el puesto seleccionado a la siguiente página JSP, que depende del perfil del usuario autenticado.

Como se puede observar en la *Figura 4.7*, si se trata de un usuario administrador y operador, se envía el formulario a `mantenimiento.jsp`, y en caso de que sea un usuario operador, a `menuRegistro.jsp`.

En caso de que esta operación sea cancelada, la aplicación devuelve el control a la página de autenticación.

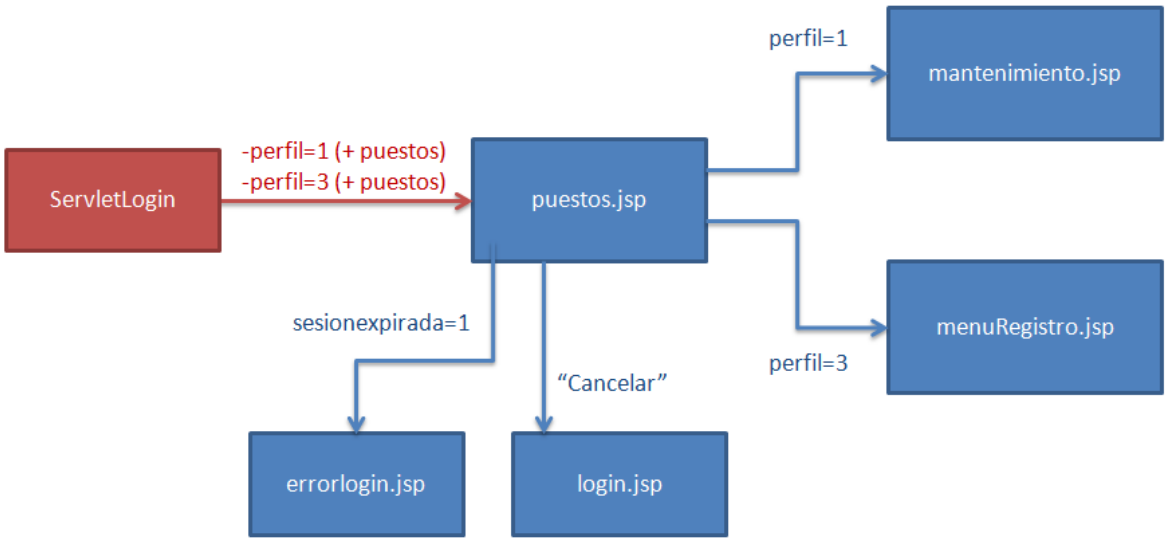


Figura 4.7 Flujo de entrada y salida a puestos.jsp

❖ Opción de registro.

menuRegistro.jsp

A esta página JSP solo acceden usuarios con perfil de operador. En ella se establece el puesto para la sesión. Los accesos a `menuRegistro.jsp` se ilustran en la *Figura 4.8*.

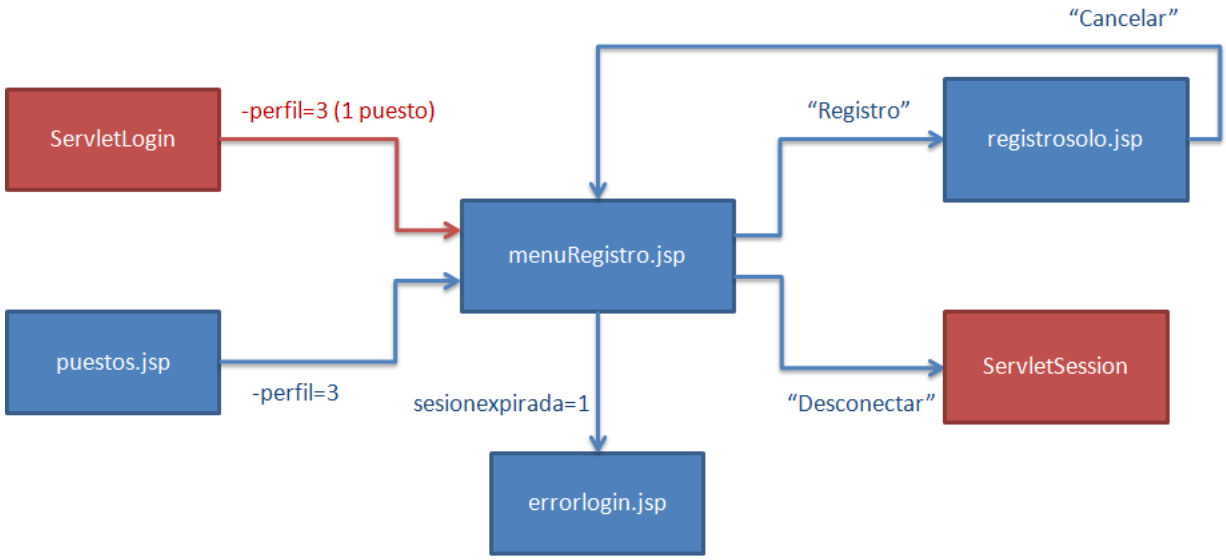


Figura 4.8 Flujo de entrada y salida a menuRegistro.jsp

Se accede a esta página desde varios puntos de la ejecución de la aplicación:

- Los operadores con un solo puesto acceden tras la autenticación, desde `ServletLogin`.
- Si un operador tiene más de un puesto, tendrá que seleccionarlo previamente a través de la página `puestos.jsp`, accediendo después a esta página.
- En caso de que se cancele el registro de un usuario con perfil de operador, se vuelve a esta página.

Antes de establecer el puesto de la sesión, es necesario obtenerlo, de la siguiente manera:

- Si ya hay puesto de sesión establecido, se obtiene dicho puesto.
- Si por el contrario no existe aún el puesto en la sesión actual, se tienen en cuenta dos posibles casos:
  - Si el usuario tiene un solo puesto, se obtiene de la base de datos.
  - En caso de que el usuario tenga más de un puesto, se obtiene el puesto seleccionado para esta sesión del parámetro correspondiente del formulario proveniente de `puestos.jsp`.

Una vez se ha establecido el puesto de la sesión, en caso de que no existiera ya, el operador ya puede comenzar un registro. Accederá a la opción de registro seleccionando el botón correspondiente, delegando en la página `registrosolo.jsp` el control de la ejecución de la aplicación.

❖ Registro.

registrosolo.jsp

Esta página JSP muestra el formulario de registro que utilizan los operadores para introducir la información relacionada con las llamadas telefónicas atendidas.

Por tanto, acceden a esta página JSP los usuarios con perfil de operador. Aquellos con perfil administrador-operador, que también pueden realizar registros de llamadas, acceden a una página similar, `registro.jsp`, que se detalla a continuación de ésta.

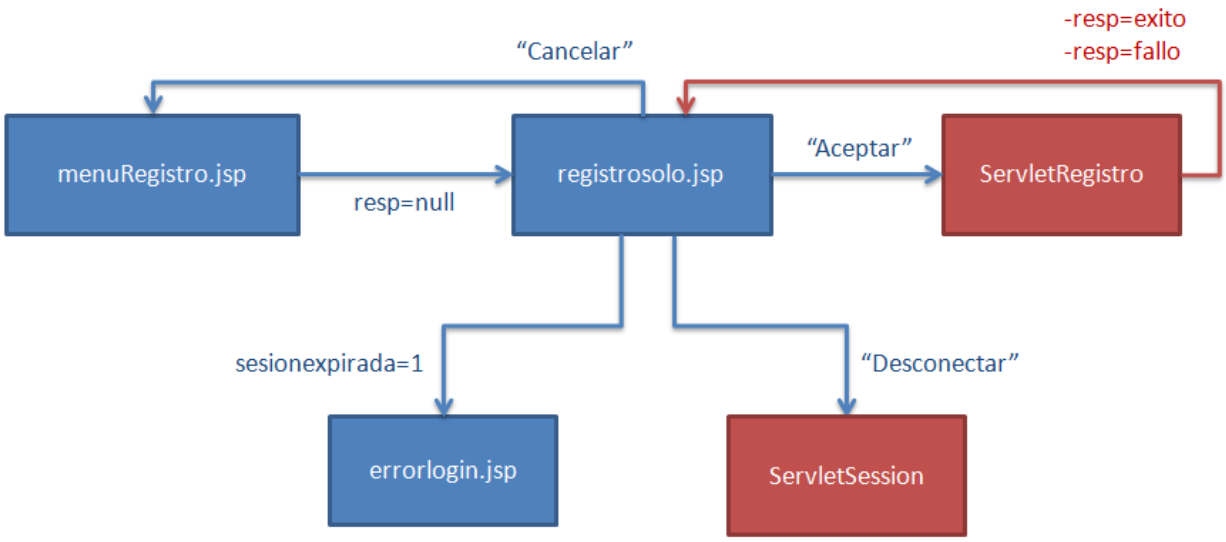


Figura 4.9 Flujo de entrada y salida a registrosolo.jsp

Como se indica en la *Figura 4.9*, a esta página `registrosolo.jsp` se puede acceder en dos circunstancias:

- En caso de que el operador no haya realizado aún ningún registro en la actual sesión, o bien haya cancelado un registro previo, al seleccionar la opción de Registro en la página `menuRegistro.jsp` se accede a esta página.
- Si el operador ha realizado previamente un registro, una vez gestionado en el servlet `ServletRegistro`, se vuelve a ofrecer automáticamente un nuevo registro al operador, accediendo para ello a esta página. En este caso, se indica al inicio mediante un mensaje de alerta el resultado de éxito o fracaso del registro anterior, en función del valor del correspondiente parámetro de la petición.

Para los dos casos anteriores, la página ejecuta el mismo código, que se explica a continuación.

En primer lugar se obtiene el usuario y puesto de éste, ambos atributos de la sesión. Con esta información, se accede en la base de datos a las propiedades del usuario y el puesto, para por un lado mostrarlas al operador que está realizando el registro, a modo de identificación personal en la sección "Datos del operador", y por otro para enviar estos datos como parte del formulario del registro.

En esta página JSP se implementan los métodos que llevan el control del temporizador que contabiliza el tiempo que dura el registro. En la sección "Datos horario" se muestran los datos relacionados con la fecha y hora de inicio del registro y el temporizador muestra el tiempo que ha transcurrido desde el inicio de la llamada.

Los campos del formulario que se muestran a continuación componen la sección "Datos ciudadano" y son introducidos por el operador de manera opcional, en caso de que sean proporcionados por el usuario que está realizando la consulta telefónica.

Para implementar las siguientes secciones, "Asunto" y "Consultas", se obtienen de la base de datos los asuntos y todas las categorías y temas que forman parte del contenido que puede ser consultado por el operador con el fin de responder al ciudadano.

Los asuntos que se recuperan y muestran en el menú desplegable correspondiente son dos: Consulta de Información y Sugerencias. El operador debe seleccionar el tipo de asunto relativo a la llamada, y en función de dicha selección, rellena los siguientes campos del formulario.

Si se selecciona Sugerencias en dicho menú, se habilita únicamente el campo Observaciones para poder introducir el texto relacionado con la queja o sugerencia recibida. En este caso, ya estaría completado el registro.

Si por el contrario el operador selecciona Consulta de Información en el menú del asunto, además del campo Observaciones se habilita el menú desplegable de la categoría en la sección Consultas.

En este menú se muestran las categorías que se pueden consultar. Una vez elegida una de ellas, se ofrece al usuario la posibilidad de asignar un tema concreto dentro del ámbito de la categoría seleccionada. Si accede, se habilita entonces el menú desplegable de los temas, que muestra los temas asociados a la categoría previamente indicada.

Tras seleccionar el tema en el menú desplegable, se abre en el navegador por defecto la url almacenada en la base de datos en relación con el tema, en la que está contenida la información que el operador necesita consultar.

Esta operación de consulta de información se puede realizar hasta un máximo de diez veces por registro, siendo recopiladas las consultas realizadas durante el registro en la tabla que se muestra al final del formulario.

Tanto si se especifica un tema como si solamente se indica la categoría de la consulta, se añade a la tabla una nueva entrada, almacenándose el tipo de asunto, la categoría y el tema, en caso de que se haya especificado uno, de cada consulta.

Toda la información de este formulario, incluido el contenido de la tabla, se envía al servlet `ServletRegistro` al seleccionar el botón Aceptar al final del formulario.

En caso de cancelarse la operación, se invoca la página previa `menuRegistro.jsp`.

registro.jsp

Al igual que `registrosolo.jsp`, esta página muestra el formulario de registro, en este caso el utilizado por un usuario administrador-operador para introducir los datos obtenidos durante una llamada atendida.

El comportamiento y contenido de ambas páginas son básicamente los mismos, con algunas diferencias que se detallan a continuación.

En la *Figura 4.10* se puede observar que el esquema de los flujos de entrada salida es el mismo para ambas páginas JSP.

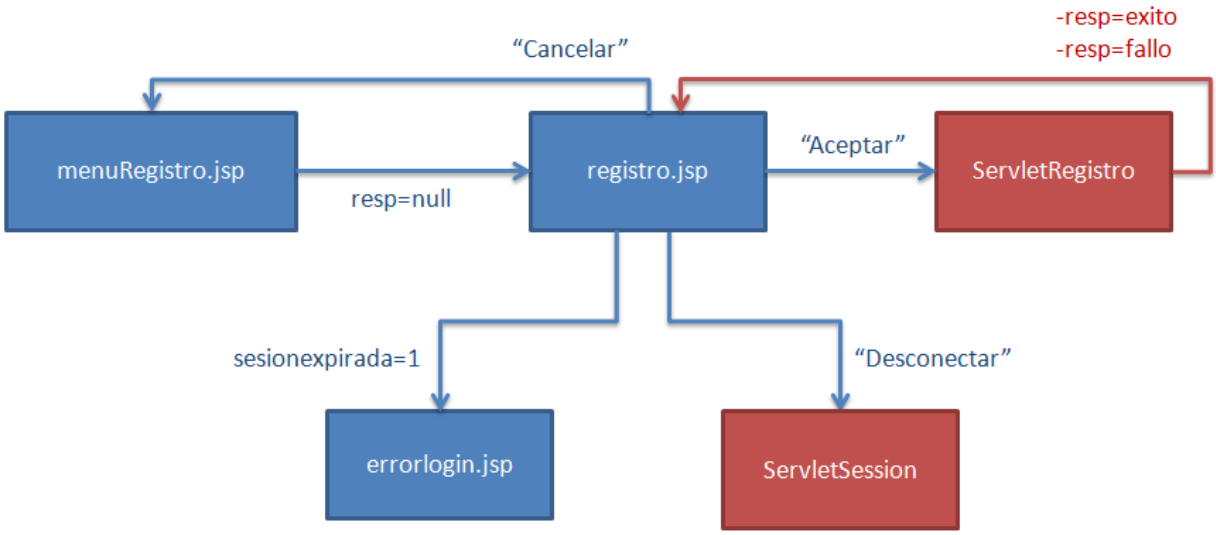


Figura 4.10 Flujo de entrada y salida a registro.jsp

En la cabecera de esta página JSP, al tratarse de un usuario administrador operador, además de la opción de registro se muestran también las opciones de administración:

- Mantenimiento
- Registro
- Estadísticas

A la página `registro.jsp` se puede acceder también en las mismas dos circunstancias que a `registrosolo.jsp`, con la excepción de que el usuario selecciona la opción de Registro en `mantenimiento.jsp`, en lugar de en `menuRegistro.jsp`.

Del mismo modo, si se cancela la operación, la página que toma el control de la ejecución es `mantenimiento.jsp` en lugar de `menuRegistro.jsp`.

Además en esta página JSP se establece el valor de un parámetro oculto del formulario que se envía al servlet `ServletRegistro`, que indica que el formulario proviene de la página de `registro.jsp` y no `registrosolo.jsp`.

ServletRegistro.java

Este servlet se encarga de generar el informe correspondiente a un registro llevado a cabo por un operador. Para ello se encarga de recopilar la información obtenida del formulario de registro, y a partir de estos datos crea el informe, que finalmente almacena en la base de datos.

A este servlet se accede, como se ilustra en la *Figura 4.11*, desde la página de registro. Si el operador tiene perfil de usuario de administrador-operador, provendrá de `registro.jsp`, y si el perfil es de operador, accederá desde `registrosolo.jsp`.

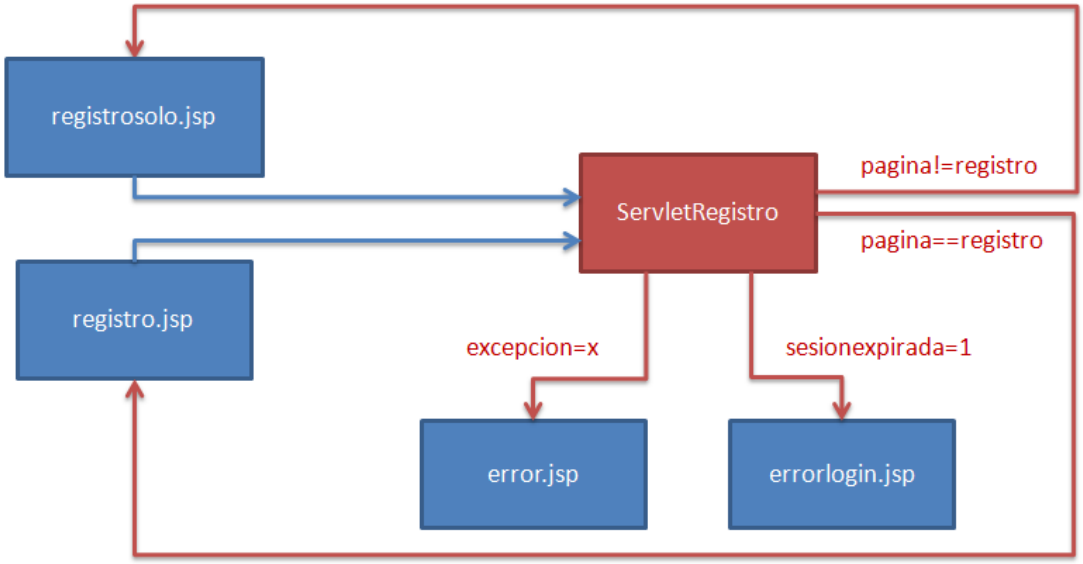


Figura 4.11 Flujo de entrada y salida a ServletRegistro.jsp

El servlet comienza creando un nuevo informe, y va dando valor a los atributos de dicho objeto a partir de los parámetros obtenidos de la petición proveniente del registro.

Para añadir consultas al registro, se comprueba si había entradas en la tabla de consultas del registro. En ese caso, se añaden también al informe.

Finalmente, se almacena en la base de datos tanto el informe completo como cada una de las consultas realizadas. El resultado exitoso o fallido de la operación de inserción en la base de datos se almacena en el atributo de la petición correspondiente, de manera que en la página JSP que se ejecuta a continuación pueda ser consultado.

De la misma manera que se accede a este servlet desde registro.jsp o desde registrosolo.jsp, el servlet, una vez finalizado su cometido, devuelve el control de la ejecución a la página JSP que lo llamó, ofreciéndole al operador un nuevo registro que completar. Para determinar a qué página debe encaminar la ejecución, consulta el parámetro correspondiente del formulario recibido, que contiene la información sobre la página de la que se provenía.

❖ Opciones de administración.

mantenimiento.jsp

A esta página JSP acceden usuarios con perfil de administrador y de administrador-operador. El acceso es también posible para estos dos tipos de usuario, como se puede observar en la *Figura 4.12*, a todas las opciones del mantenimiento:

- `alta_categoria.jsp`
- `baja_categoria.jsp`
- `alta_tema.jsp`
- `baja_tema.jsp`
- `alta_operador.jsp`
- `baja_operador.jsp`

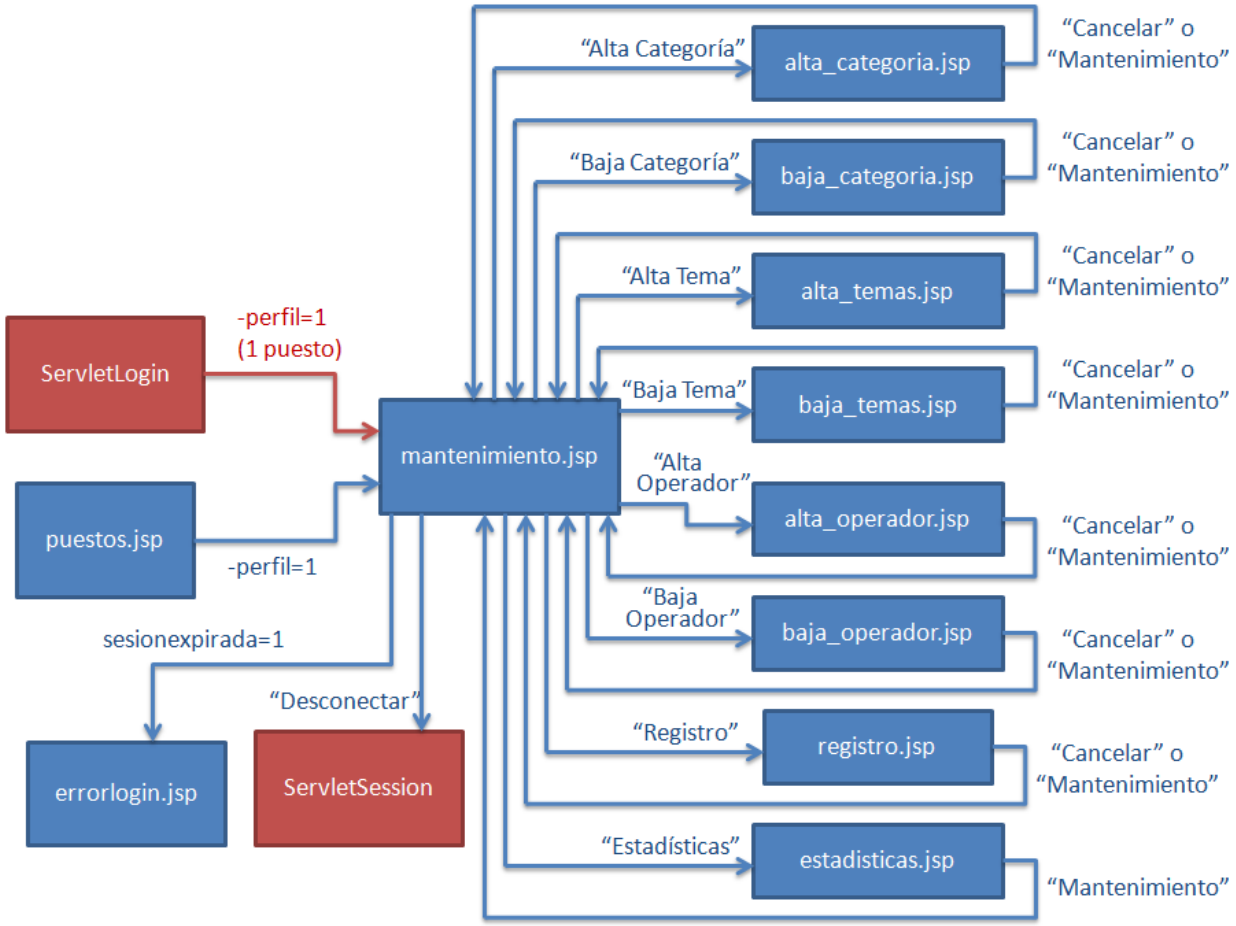


Figura 4.12 Flujo de entrada y salida a mantenimiento.jsp

Al igual que `menuRegistro.jsp`, se encarga de establecer el puesto del usuario para la sesión. En `menuRegistro.jsp` se trataban los usuarios con perfil de operador, y en este caso se lleva a cabo esta acción para los usuarios con perfil de administrador-operador.

Los usuarios con perfil de administrador que acceden a esta página no necesitan realizar esta acción, ya que no tienen varios puestos. Para ellos, `mantenimiento.jsp` hace simplemente la función de página de inicio tras la autenticación.

Es posible acceder a `mantenimiento.jsp` desde varios puntos de la ejecución de la aplicación:

- Los usuarios con perfil de administrador y aquellos usuarios con perfil de operador-administrador que tienen asignado un solo puesto acceden tras la autenticación, desde el servlet `ServletLogin`.

- Los usuarios de tipo operador-administrador que tienen más de un puesto, deben seleccionarlo tras la autenticación a través de la página `puestos.jsp`, accediendo después a `mantenimiento.jsp`.
- Desde la página de registro y todas las páginas pertenecientes al mantenimiento, en caso de cancelarse la operación correspondiente, se vuelve a esta página de inicio.
- Desde la página de registro y todas las páginas pertenecientes al mantenimiento y a las estadísticas, al seleccionar la opción de mantenimiento, se accede a esta página.

Para obtener el puesto asociado al usuario de tipo operador-administrador se siguen los mismos pasos que en `menuRegistro.jsp`.

Tras obtener el puesto del usuario, se estable como puesto de la sesión, en caso de que no estuviera ya establecido.

A partir de este punto, el usuario ya puede acceder a las diferentes funcionalidades que le ofrece la aplicación, en función de su perfil. Seleccionando el botón asociado a la opción deseada se accederá a la página JSP que corresponda a dicha opción.

❖ Opciones de mantenimiento.

alta_categoria.jsp

Desde esta página, el usuario con funciones de administrador tiene la opción de añadir nuevas categorías a la lista de aquellas que pueden ser consultadas por un usuario operador durante el registro de una llamada en la que se realiza una consulta de información.

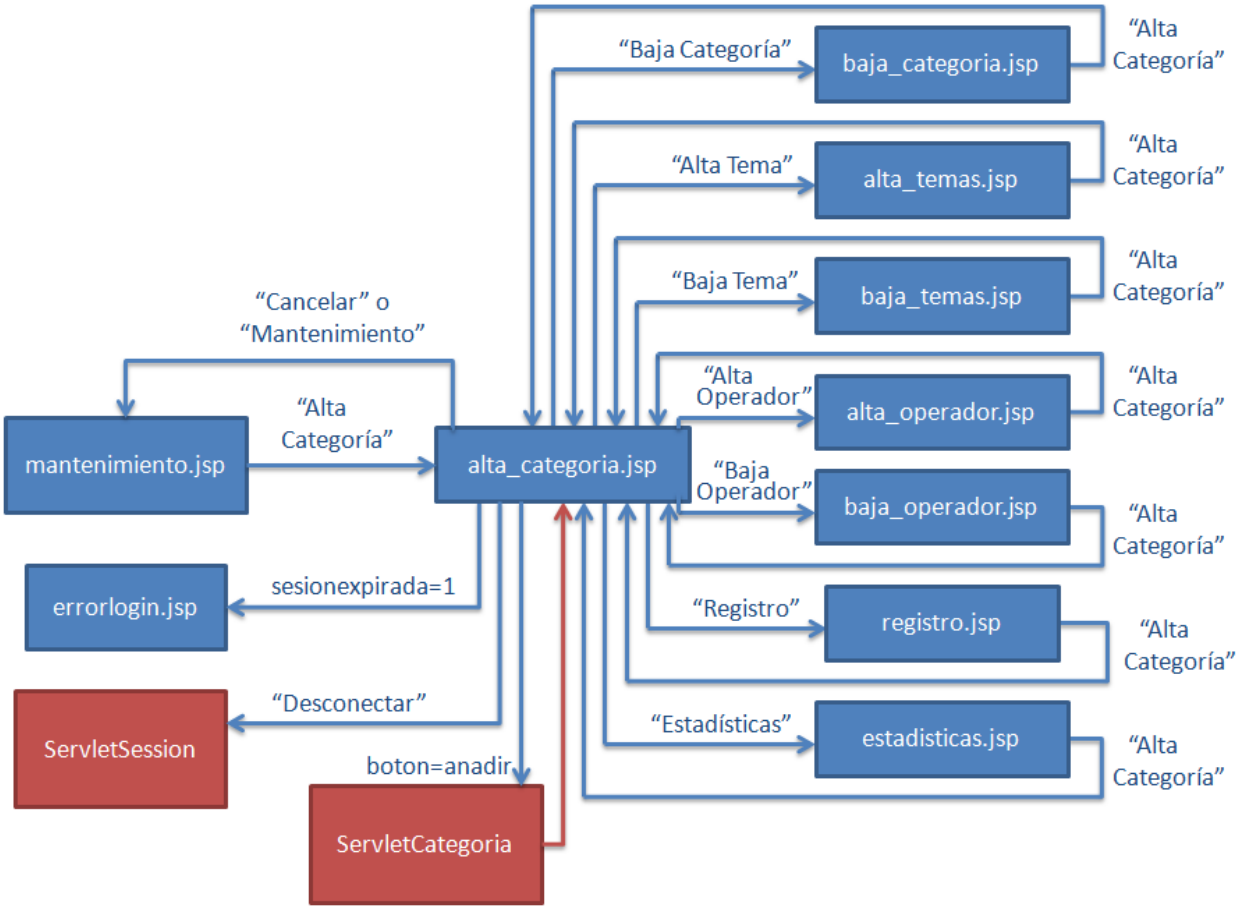


Figura 4.13 Flujo de entrada y salida a alta_categoria.jsp

Como se puede observar en la *Figura 4.13*, se puede acceder a esta página JSP desde los siguientes puntos de ejecución de la aplicación:

- Desde la página de registro y todas las páginas pertenecientes al mantenimiento y a las estadísticas, al seleccionar la opción de "Alta categoría", se accede a esta página, iniciando un nuevo proceso de alta de categoría.
- Si el administrador ha intentado previamente llevar a cabo el alta de una categoría, una vez gestionada dicha acción en el servlet `ServletCategoria`, se vuelve a acceder a esta página.
- Además hay un caso concreto en el que desde `alta_categoria.jsp` se invoca esta misma página para que se cargue de nuevo el contenido correspondiente al anterior proceso de alta. Como se detalla a continuación, esto sucede al procesar la respuesta del servlet `ServletCategoria`, cuando ya existe la categoría que se ha intentado añadir y el usuario vuelve a intentarlo de nuevo.

En función del punto del proceso de alta en el que se ha accedido a esta página, se realizan diferentes acciones. Este flujo se ilustra en la *Figura 4.14*.

- Cuando se está iniciando un nuevo proceso de alta, se ofrece un formulario en el que el administrador puede introducir la categoría que desea añadir.

  Dicho formulario se envía al servlet `ServletCategoria`, indicando, además de la categoría a añadir, la acción que se va a llevar a cabo.
- Si el paso previo es el servlet `ServletCategoria`, esto indica que no es el primer acceso a esta página. El formulario de alta ya ha sido enviado, y se actúa en función del resultado de la acción solicitada. El valor de dicho resultado está contenido en el estado de la aplicación, habiendo sido almacenado por `ServletCategoria` como atributo de la petición.

  A. Si se ha añadido correctamente la categoría, se muestra un mensaje indicando el éxito de la operación.

  B. Si ha sucedido algún problema al intentar insertar la categoría en la base de datos, se informa mediante un mensaje de que se ha producido un fallo al intentar dar de alta la categoría.

  C. En caso de que la categoría ya exista en el sistema, el administrador es informado de ello y se le ofrece la posibilidad de intentarlo de nuevo. Si el usuario accede, se solicita que se vuelva a cargar la página `alta_categoria.jsp`, para iniciar un nuevo proceso de alta.

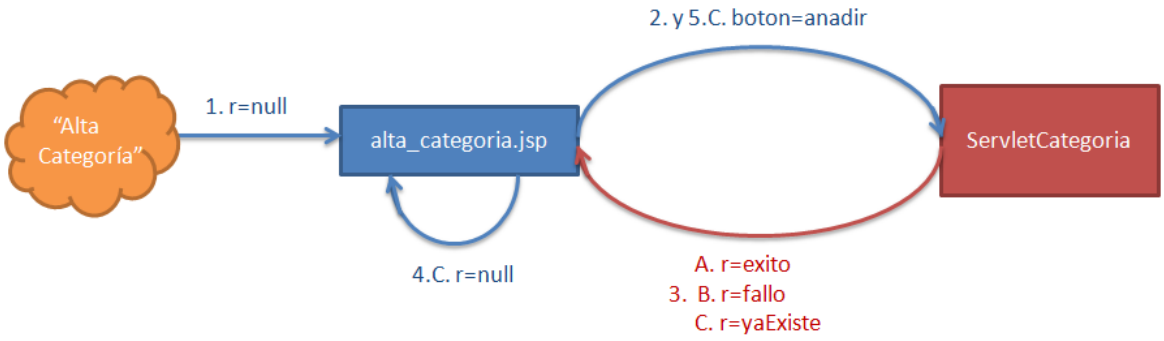


Figura 4.14 Flujo entre alta_categoria.jsp y ServletCategoria.java

baja_categoria.jsp

En esta página se ofrece la posibilidad de eliminar categorías existentes en el sistema. Dichas categorías pueden tener temas asociados, lo que supone un aspecto a tener en cuenta a la hora de eliminar una categoría.

El flujo de entrada y salida a `baja_categoria.jsp` se representa en la *Figura 4.15*.

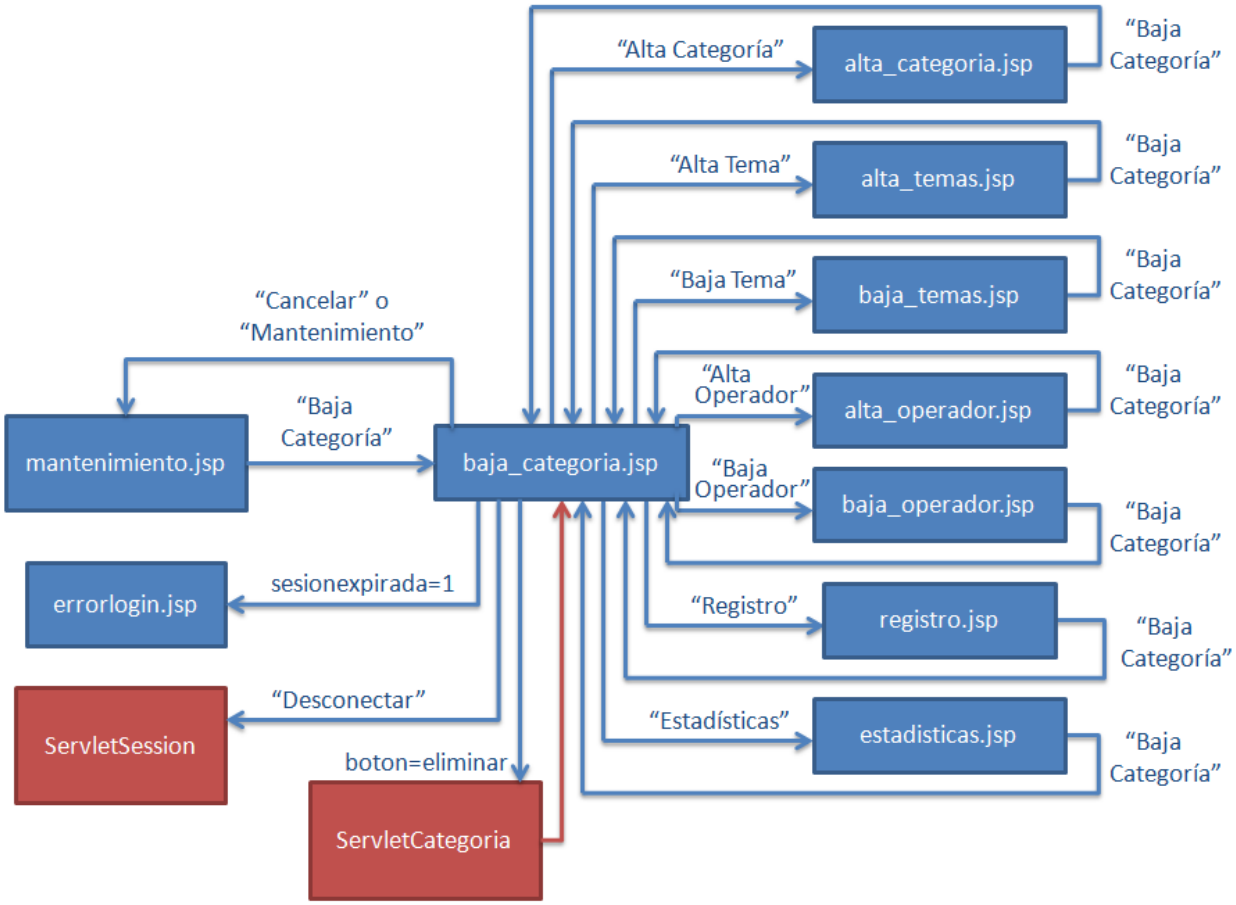


Figura 4.15 Flujo de entrada y salida a baja_categoria.jsp

Para acceder a esta página JSP la ejecución de la aplicación debe encontrarse en uno de los siguientes puntos:

- Al igual que en el caso de `alta_categoria.jsp`, desde la página de registro y todas las páginas pertenecientes al mantenimiento y a las estadísticas, si se selecciona la opción de Baja categoría, se accede a esta página, iniciándose un nuevo proceso de baja de categoría.
- Si el proceso de baja ya ha sido iniciado y el administrador ha seleccionado una categoría a dar de baja, el servlet `ServletCategoria` devuelve el control de la ejecución a esta página. En el transcurso de la gestión de baja por parte del servlet `ServletCategoria` esta acción puede realizarse más de una vez, concretamente en el caso en el que la categoría tenga temas asociados, como se explica a continuación.

La página `baja_categoria.jsp` mostrará el contenido correspondiente, teniendo en cuenta el punto en el que se encuentre la ejecución del proceso de baja ilustrado en la *Figura 4.16*:

- Si es la primera vez que se accede a esta página, se está iniciando un nuevo proceso de baja. En este caso se ofrece un menú desplegable con las categorías pertenecientes al sistema, para que el administrador pueda escoger aquella que desea eliminar.

  La categoría escogida se envía al servlet `ServletCategoria` en un formulario, que también incluye la acción que se desea realizar, la eliminación de la categoría.

- En caso de que se provenga de `ServletCategoria`, no se tratará del primer acceso a esta página. La categoría a eliminar ya se ha enviado al servlet, y éste almacena el resultado de la operación de baja en el atributo correspondiente.

  A. Si la categoría seleccionada tiene temas asociados, se informa sobre ello al administrador, dándole la posibilidad de decidir si desea eliminar o no la categoría junto con los temas asociados a ésta.

En caso de acceder el administrador a eliminarlos, se vuelve a enviar al servlet `ServletCategoria` el formulario con la categoría escogida, y la confirmación de la eliminación de ésta y sus temas.

B. En caso de haberse llevado a cabo correctamente la eliminación de la categoría, o en su caso de la categoría y los temas asociados, esta página muestra un mensaje en el que informa sobre el éxito de la baja.

C. Si por el contrario no se ha podido eliminar la categoría, o algún tema asociado en caso de que proceda, de la base de datos, se indica que se ha producido un error en el proceso de baja de la categoría.

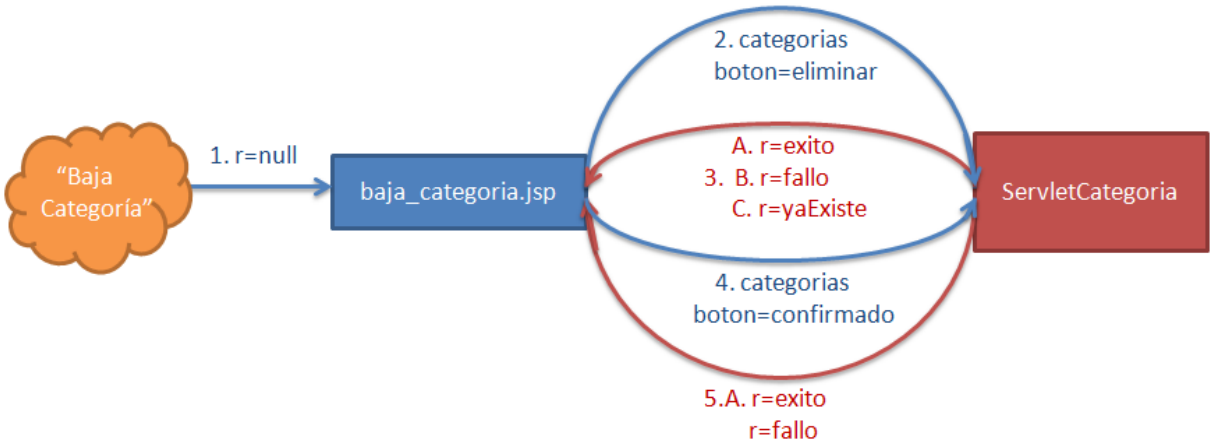


Figura 4.16 Flujo entre baja_categoria.jsp y ServletCategoria.java

ServletCategoria.java

Este servlet gestiona las peticiones de altas y bajas de las categorías del sistema. Se accede al servlet desde las páginas que solicitan dichas altas y bajas: `alta_categoria.jsp` y `baja_categoria.jsp`, como se indica en la *Figura 4.17*.

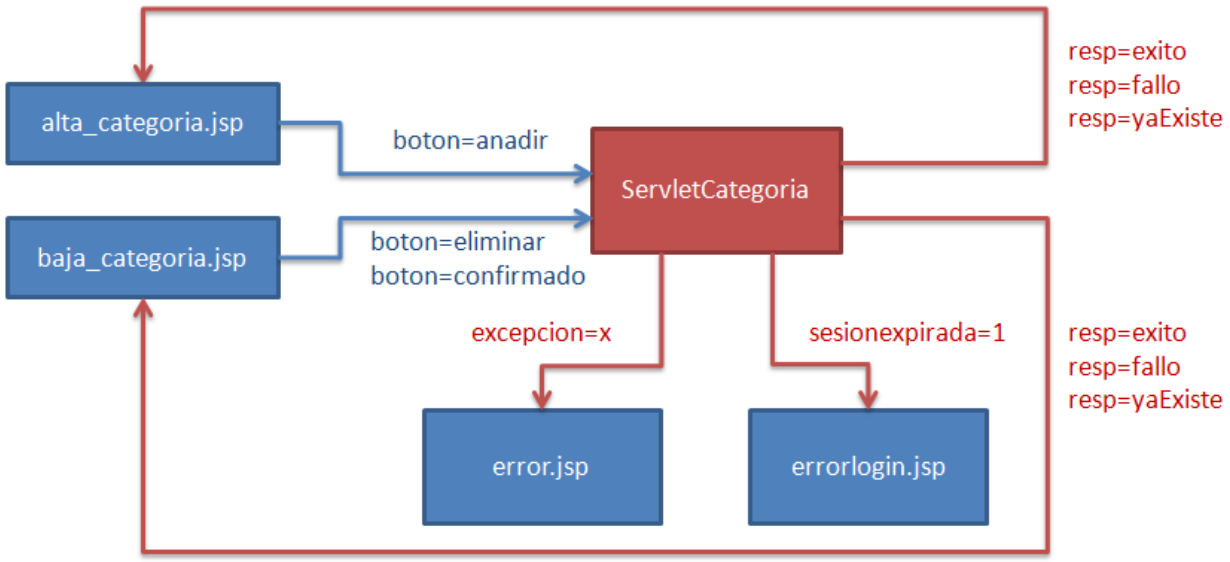


Figura 4.17 Flujo de entrada y salida a ServletCategoria.java

Cuando se recibe una de estas peticiones, en primer lugar se determina si el administrador ha solicitado el alta o la baja de una categoría, en función del correspondiente parámetro del formulario.

Alta de categoría

En este caso lo primero es obtener del formulario el parámetro correspondiente al nombre de la categoría a añadir, y a partir de él, se crea una nueva categoría que lleve este nombre.

Es necesario comprobar que el nombre no pertenezca a ninguna categoría existente en el sistema. Para ello se obtienen las categorías almacenadas en la base de datos y se comparan sus nombres con el nuevo nombre a añadir. A partir de esta comparación pueden darse los siguientes casos:

- Si la categoría ya existe, se consulta su estado para comprobar si está activa o inactiva, ya que en caso de estar inactiva, se contabilizará como añadida al cambiar su estado a activa.
    - En caso de estar inactiva, se procede a actualizar su estado a activa.
        - Si la operación de acceso a base de datos se realiza correctamente, el resultado de la acción será exitoso.
        - En caso contrario el resultado será fallido.
    - Si la categoría existe y está activa, el resultado de la operación almacena este estado, indicando que ya existe esta categoría.
- En caso de que la categoría no exista previamente, se intenta añadir a la base de datos del sistema.
    - Si se añade correctamente, el resultado de la operación de alta es exitoso.
    - Si no se puede añadir la categoría a la base de datos, el resultado de la acción es fallido.

Tras gestionar el proceso de alta de la categoría solicitada y almacenar el resultado de la operación en el atributo correspondiente, el servlet devuelve el control de la ejecución de nuevo a la página `alta_categoria.jsp`.

Baja de categoría

En el caso de que se esté gestionando la baja de una categoría, hay dos posibles situaciones que llevan a `baja_categoria.jsp` a acudir a este servlet.

La primera de ellas es que se haya iniciado el proceso de baja, habiéndose seleccionado la categoría a eliminar. La segunda posibilidad es que, teniendo la categoría elegida temas a consultar asociados, el administrador haya confirmado que desea eliminar tanto la categoría como los temas del sistema.

En ambos casos, por un lado se recupera de la base de datos el listado de categorías del sistema, y por otro se obtiene del formulario de baja el parámetro que contiene el valor de la categoría que se ha seleccionado.

A. Categoría a eliminar seleccionada.

En primer lugar, se procede a determinar si la categoría seleccionada tiene temas asociados. Para ello se consulta la base de datos del sistema y se actúa en función del resultado obtenido de dicha consulta.

- Si hay temas asociados a la categoría elegida, es necesario consultar al administrador si desea eliminarlos antes de llevar a cabo la baja. El resultado de la operación almacena este estado, indicando que existen temas asociados a la categoría recibida.

  Dicho resultado junto con la categoría objetivo son almacenados en los atributos correspondientes, formando parte del estado de la aplicación. Dicho estado será consultado por `baja_categoria.jsp`.
- Si la categoría no tiene ningún tema asociado, se procede a la eliminación de ésta de la base de datos.
    - Si se elimina correctamente, el resultado de la operación de baja es exitoso.
    - Si se produce algún problema en el proceso de eliminación de la categoría de la base de datos, se indica el fallo en el resultado de la operación.

El resultado de la eliminación se almacena en el correspondiente atributo, que será consultado por `baja_categoria.jsp`.

B. Eliminación de categoría y temas confirmada.

En este caso el servlet recibe la confirmación de la eliminación de la categoría y sus temas asociados. Además de esta confirmación, obtiene del formulario de `baja_categoria.jsp` el parámetro que contiene el valor de la categoría que se desea eliminar.

Recupera de la base de datos los temas asociados a la categoría obtenida y procede a la eliminación de la categoría y sus temas.

- Si es posible eliminarlos correctamente de la base de datos, se indica en el resultado de la operación que la baja se ha realizado con éxito.
- En caso de que no sea posible eliminar correctamente alguno de los elementos, el resultado de la operación es fallido.

En los dos casos se almacena el resultado de la operación en el correspondiente atributo de la petición.

alta_temas.jsp

En esta página se permite dar de alta nuevos temas que podrán ser accedidos por los operadores que deseen realizar una consulta de información. Cada uno de los temas del sistema pertenece a una categoría, por lo que a la hora de añadir un tema se selecciona la categoría a la que se quiere asociar.

En la *Figura 4.18* se muestra el esquema de los componentes web que acceden a `alta_temas.jsp` y a los que accede dicha página JSP. Dichos accesos son detallados a continuación.

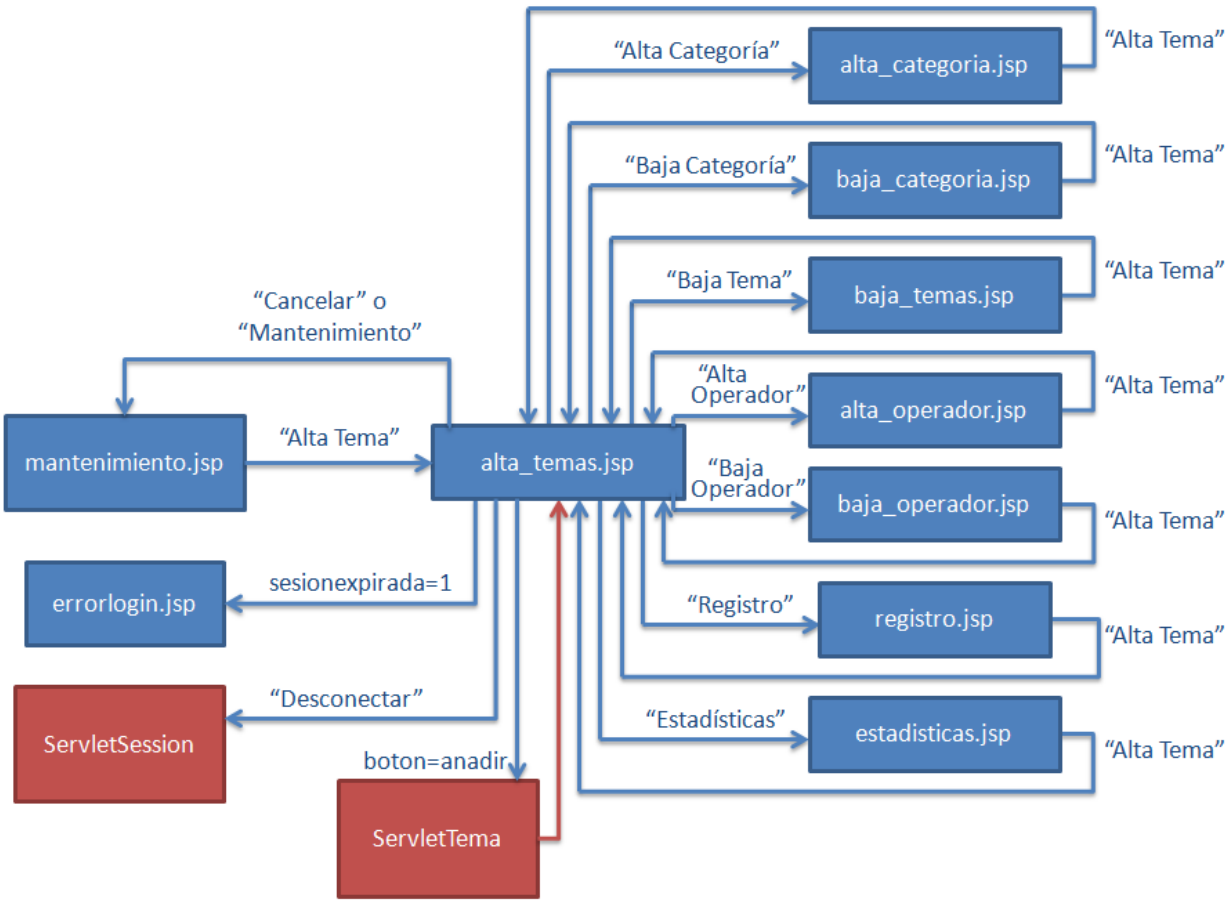


Figura 4.18 Flujo de entrada y salida a alta_temas.jsp

El comportamiento de esta página es muy similar al de la página `alta_categoria.jsp`, como se puede comprobar a continuación.

El acceso a `alta_temas.jsp` es posible si la ejecución de la aplicación se encuentra en una de las siguientes situaciones:

- En la página de registro y todas las páginas de mantenimiento y las de estadísticas, si se pulsa la opción de "Alta tema" se accede a esta página. En este punto se procede a iniciar el proceso de alta de un nuevo tema.
- Si ya se ha iniciado un proceso de alta, y el administrador ha introducido el tema que desea añadir, esta operación se gestiona en el servlet `ServletTema`, que vuelve a acceder a esta página tras finalizar la operación.
- Por último, existe un caso en el que se accede a `alta_temas.jsp` desde esta misma página. Este acceso se realiza para volver a cargar el contenido correspondiente al anterior proceso de alta, cuando ya existe el tema que se ha intentado añadir y el usuario decide intentarlo de nuevo.

El contenido mostrado por esta página dependerá de la situación de las anteriores de la que se provenga, como se puede observar en la *Figura 4.19*.

- En el inicio de un nuevo proceso de alta de un tema, el administrador debe rellenar un formulario indicando el tema que desea añadir, la URL en la que está almacenada la información que el usuario operador consultará y la categoría a la que pertenece el tema en cuestión.

  Finalmente, este formulario es enviado al servlet `ServletTema`, donde además de indicarse los parámetros anteriores, se indica al servlet qué acción se desea realizar, que será el alta de un nuevo tema.

- En caso de provenir del servlet `ServletTema`, no se tratará del primer acceso a esta página. En este caso, el alta del tema ya se ha solicitado y procesado en este servlet. El comportamiento será diferente, en función del resultado que haya sido almacenado por el servlet en el estado de la aplicación:

  A. Si el resultado es exitoso significa que se ha añadido correctamente el tema, y la página muestra un mensaje informando del éxito del alta.

  B. En caso contrario, si el resultado indica que se ha producido un fallo al intentar añadir el tema en la base de datos, la página muestra un mensaje en el que se indica que ha sucedido un error en el proceso de alta del tema.

  C. Si el tema ya existe en el sistema se informa al administrador de la situación y se le ofrece la posibilidad de volver a añadir otro tema. Si el usuario acepta, se carga de nuevo la página `alta_temas.jsp` con los datos de la solicitud previa, para que el administrador los modifique y se inicie un nuevo proceso de alta.

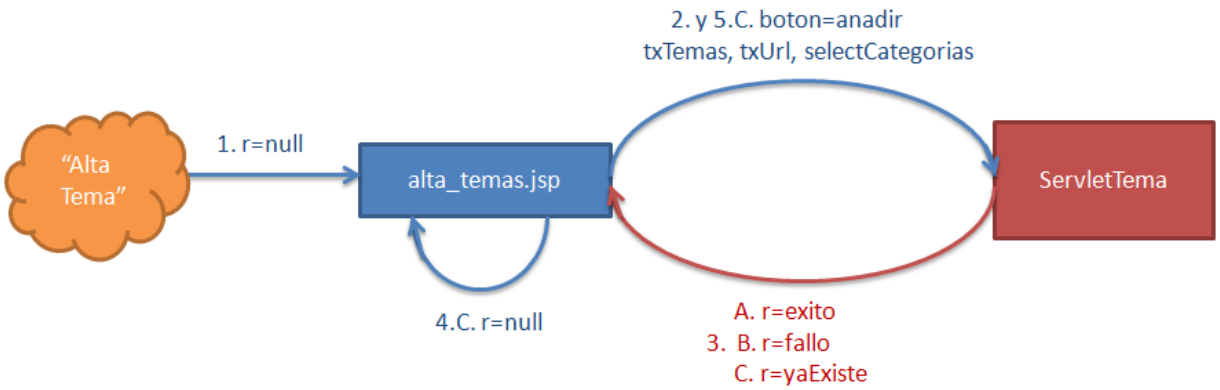


Figura 4.19 Flujo entre alta_temas.jsp y ServletTema.java

baja_temas.jsp

Desde esta página los administradores del sistema pueden dar de baja un tema, eliminándolo de la base de datos, de manera que ya no pueda ser consultado por los operadores.

A continuación se indican los puntos de la aplicación desde los que se puede acceder a `baja_temas.jsp`, y a los que accede este JSP. El esquema se ilustra en la *Figura 4.20*.

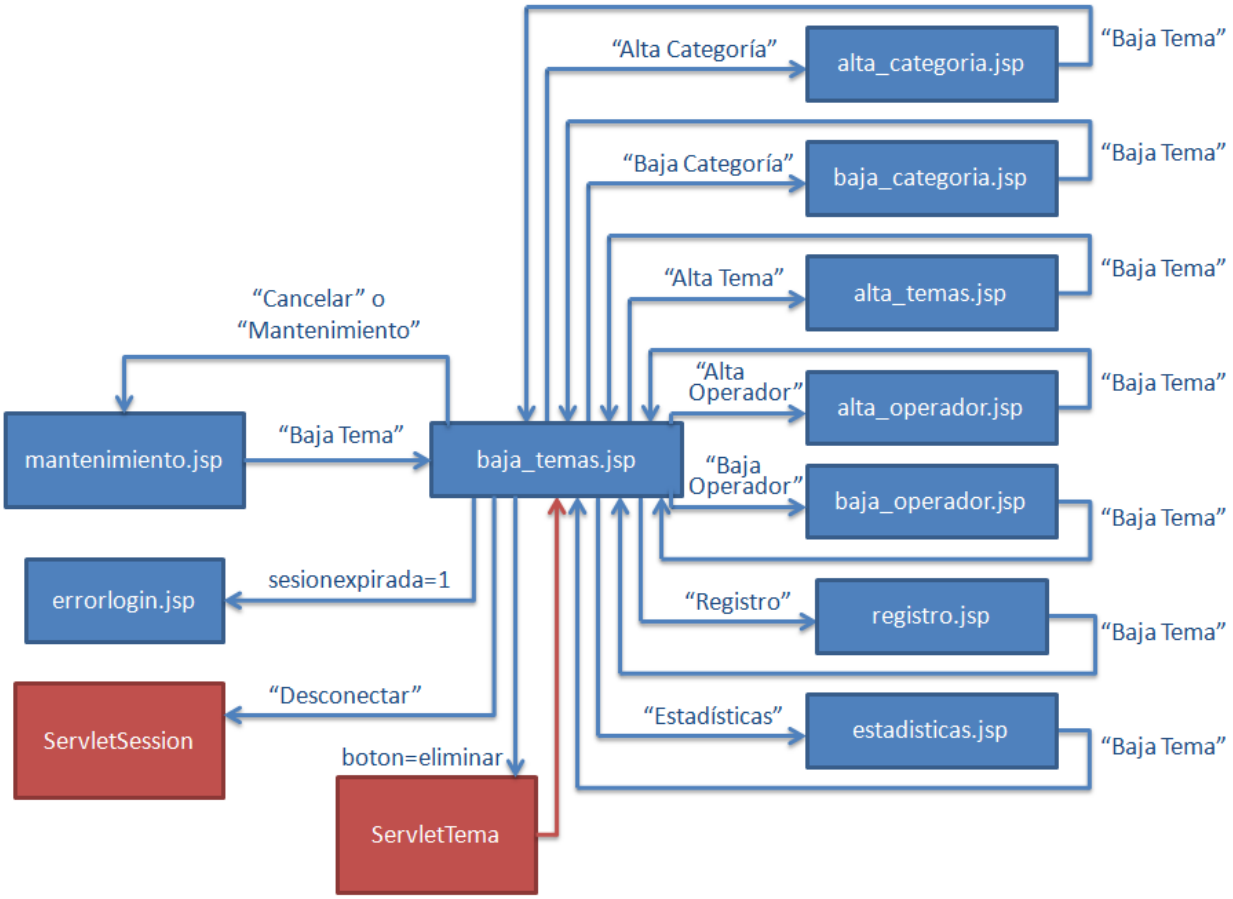


Figura 4.20 Flujo de entrada y salida a baja_temas.jsp

Es posible acceder a esta página JSP desde los siguientes puntos de la ejecución de la aplicación:

- Como es común para este tipo de páginas, si en la página de registro o en cualquiera de las páginas de mantenimiento o estadísticas se selecciona la opción de "Baja tema", se accede directamente a esta página. De esta forma se inicia un nuevo proceso de baja de un tema.
- Si por el contrario el proceso de baja ya ha sido iniciado, el administrador habrá seleccionado el tema a eliminar. El servlet `ServletTema`, tras gestionar la baja, devuelve el control de la ejecución a esta página.

La página `baja_temas.jsp`, según sea accedida desde una u otra de las opciones anteriores, mostrará el contenido correspondiente, según se muestra en la *Figura 4.21* y se indica a continuación:

- Si el administrador ha solicitado la baja de un tema, es la primera vez que se accede a esta página. Al iniciarse un nuevo proceso de baja de un tema, en primer lugar el administrador debe seleccionar la categoría de la que quiere eliminar el tema.

  Para ello se le ofrece un menú desplegable que muestra las categorías del sistema. Cuando el administrador selecciona una categoría de este menú, se cargan en el siguiente menú desplegable los temas asociados a la categoría escogida. Esta acción se realiza cada vez que se cambia la selección de la categoría.

Finalmente, al aceptar la operación, se envía dicho formulario al servlet `ServletTema`, incluyendo la acción que se desea llevar a cabo.

- Si el acceso a esta página se realiza desde `ServletTema`, no se trata del primer acceso a `baja_temas.jsp`. En este punto, ya se ha seleccionado el tema que se quiere eliminar y se ha enviado a `ServletTema`, obteniéndose el resultado del estado de la aplicación, mediante al atributo correspondiente al que se ha dado valor en `ServletTema`.

  A. Si el resultado es exitoso, se muestra un mensaje informativo en el que se indica que llevado a cabo correctamente la baja del tema.

  B. En caso de que haya algún problema al eliminar el tema de la base de datos, se informa al administrador de que se ha producido un error en el proceso de baja del tema.

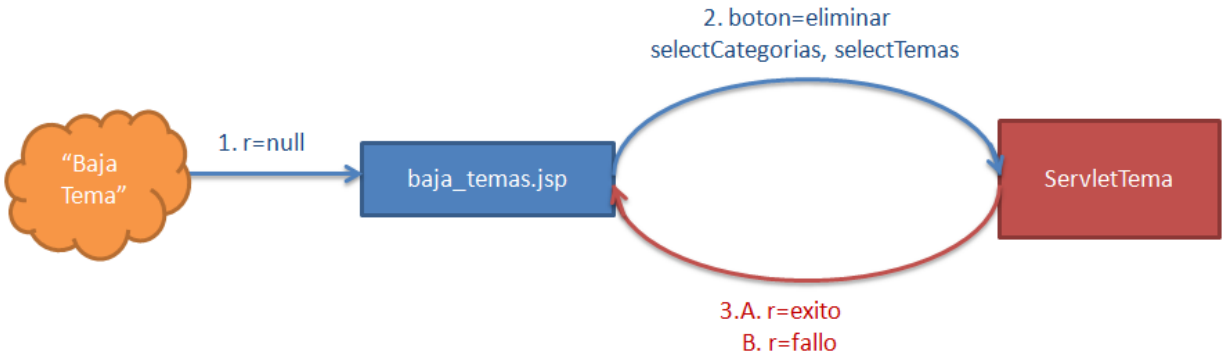


Figura 4.21 Flujo entre baja_temas.jsp y ServletTema.java

ServletTema.java

En este servlet se gestionan las solicitudes de altas y bajas de los temas pertenecientes a la aplicación. Por tanto, tienen acceso a `SevletTema` las páginas JSP que realizan estas peticiones: `alta_temas.jsp` y `baja_temas.jsp`, como se ilustra en la *Figura 4.22*.

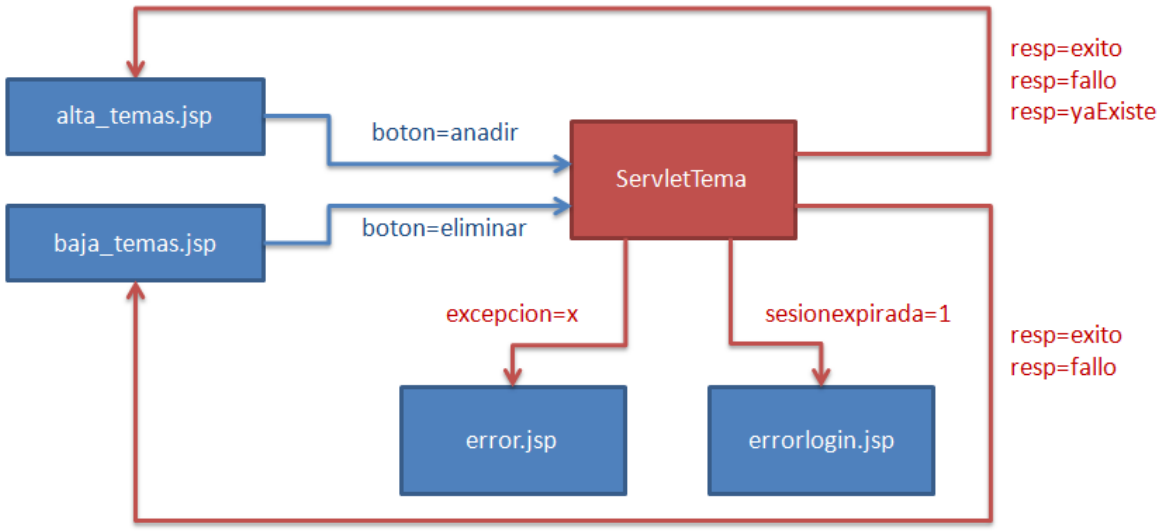


Figura 4.22 Flujo de entrada y salida a ServletTema.java

El servlet, al recibir una solicitud, identifica si se trata de una petición de alta o de baja, mediante el parámetro del formulario que indica dicha acción. Además, obtiene también el parámetro que indica la categoría a la que pertenece el tema, ya que tanto al añadir un tema como al eliminarlo, se incluye este parámetro en el formulario de la petición realizada al servlet.

Alta de tema

Se recuperan las categorías del sistema y se identifica aquella que ha sido seleccionada en el formulario de la página JSP previa. También se obtienen los restantes parámetros recibidos en el formulario proveniente `alta_temas.jsp`: el nombre del tema a añadir y la url asociada a este tema.

A continuación se crea un nuevo tema a añadir, y se da valor a sus atributos a partir de la información obtenida del formulario.

Antes de insertar el tema, se comprueba que no exista en el sistema un tema con el mismo nombre, comparándolo con todos los temas dados de alta en la aplicación.

- En caso de existir, se consulta el estado del tema, para comprobar si está activo o inactivo.
    - En caso de estar inactivo podrá añadirse, cambiando su estado a activo. Además se actualizan los atributos del tema existente con los del tema nuevo creado.
        - Si es posible realizar correctamente la actualización del tema en la base de datos, se indicará que ha sido satisfactorio el resultado de la operación.
        - Si hay algún problema en este acceso a la base de datos, se indicará que el resultado ha sido erróneo.
    - En cambio, si está ya activo no será posible insertar un tema con el mismo nombre. En este caso, el resultado de la operación indica que ya existe el tema escogido.
- Si no se encuentra en la base de datos un tema con el mismo nombre, se procede a añadirlo al sistema.
    - El resultado de la operación es exitoso si la inserción en la base de datos se realiza correctamente.
    - En caso de que el tema no se pueda insertar correctamente en la base de datos, el resultado de la operación es fallido.

Una vez finalizado el proceso de alta del nuevo tema, se almacena el resultado de la operación en el atributo correspondiente de la petición, y se devuelve el control de la ejecución a la página `alta_temas.jsp`.

Baja de tema

La baja de un tema es un proceso sencillo de gestionar. Desde `baja_temas.jsp` se acude a este servlet para realizar dicha gestión, indicando mediante el parámetro de del formulario adecuado cuál es el tema que se quiere dar de baja.

Además de obtener esta información, el servlet recupera de la base de datos el listado de temas del sistema, e identifica cuál de estos temas es el que se desea eliminar.

- Si es posible eliminar correctamente el tema seleccionado, se almacena en el resultado que la operación de baja del tema ha sido exitosa.
- Si por el contrario surge algún problema en la eliminación del tema de la base de datos, el resultado de la operación almacena que no se ha realizado correctamente.

Finalmente, dicho resultado se almacena en el atributo correspondiente, y se invoca la página `baja_temas.jsp`.

alta_operador.jsp

Esta página posibilita al administrador el proceso de alta de un operador.

El proceso de alta de un operador es muy similar al que se sigue en el caso del alta de una categoría o un tema, por lo que el comportamiento de esta página sigue el mismo esquema que el de las páginas mencionadas.

El esquema de accesos se muestra en la *Figura 4.23*, como se puede observar a continuación.

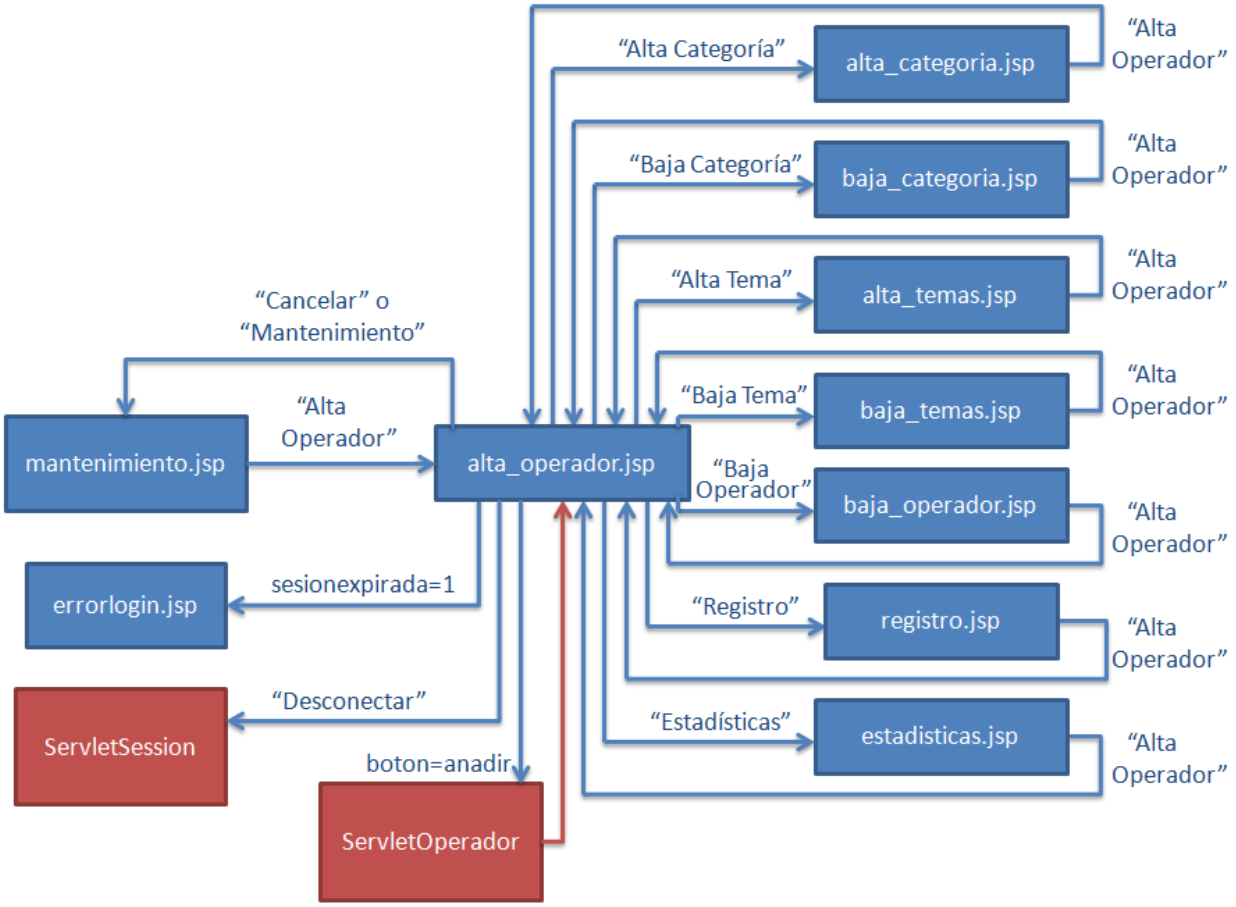


Figura 4.23 Parámetros de entrada y salida a alta_operador.jsp

Se puede acceder a esta página desde los siguientes puntos de ejecución de la aplicación:

- Estando en la página correspondiente el registro y desde todas las páginas de mantenimiento y las de estadísticas, al seleccionar la opción "Alta operador", el administrador accede a `alta_operador.jsp`, en categoría de inicio del proceso de alta de un nuevo operador.
- En el caso de que ya se haya iniciado el proceso de alta del operador, y se haya rellenado el formulario con los datos del operador a añadir, el servlet `ServletOperador` devuelve el control de la ejecución a esta página tras haber gestionado la operación de alta.
- Al igual que sucede en las páginas JSP de alta cuyo funcionamiento ha sido previamente explicado, también es posible el acceso a `alta_operador.jsp` desde esta misma página. Cuando ya existe en el sistema el operador que se ha intentado añadir y el administrador decide intentarlo de nuevo, se accede al paso anterior, cargando el contenido correspondiente al paso previo del alta, de forma que el formulario contiene en sus campos los datos introducidos inicialmente por el administrador. Este acceso también equivale a un nuevo comienzo del proceso de alta de un operador.

En función del punto de ejecución de los anteriores de donde se acceda a la página, se mostrará el contenido descrito a continuación e ilustrado en la *Figura 4.24*:

- Si se trata del inicio de un proceso de alta, se muestra el formulario a rellenar por el administrador, conteniendo los siguientes campos relativos al operador al que se quiere dar de alta en el sistema:
  - Nombre
  - NIF
  - Usuario

- Contraseña
- Puestos

Con respecto al último parámetro, los puestos que puede tener asociados el operador se ofrecen en una lista al administrador para que pueda seleccionar los que convengan.

Una vez se ha rellenado este formulario, se envía al servlet `ServletOperador`, indicando mediante un parámetro oculto que se trata de una operación de alta de un operador.

- Si no se está comenzando un proceso de alta, no se trata del primer acceso a esta página, sino que se proviene del servlet `ServletOperador`.

  En función del valor del atributo que almacena el resultado de la operación llevada a cabo por dicho servlet, se realiza una de las siguientes acciones.

  A. Si el resultado indica que la operación se ha llevado a cabo correctamente, se muestra un mensaje informativo al administrador, indicándole que el alta del operador ha sido realizada.

  B. Si por el contrario el resultado indica que no se ha realizado correctamente la acción, el mensaje mostrado indica que se ha producido un fallo en el proceso de alta.

  C. Por último, en caso de que el resultado indique que ya existe un operador con el mismo nombre en el sistema, se muestra un mensaje en el que se ofrece al administrador la posibilidad de volver a añadir otro operador. Si éste está de acuerdo, se carga de nuevo la página `alta_operador.jsp`, en cuyo formulario aparecen los datos introducidos previamente, de forma que el administrador los pueda modificar e iniciar de nuevo el proceso de alta del operador.

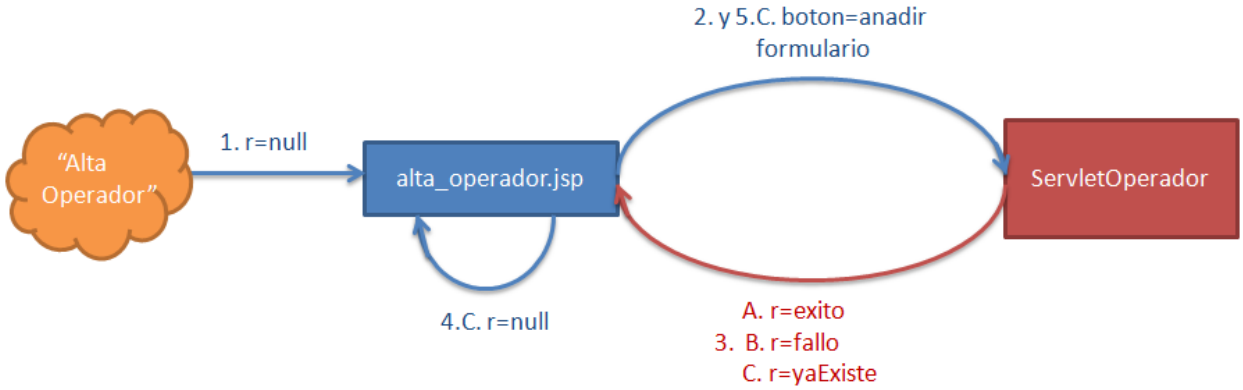


Figura 4.24 Flujo entre alta_operador.jsp y ServletOperador.java

baja_operador.jsp

En esta página se ofrece al administrador la posibilidad de dar de baja un operador del sistema.

Los puntos de la aplicación desde donde se proviene y donde se puede acceder desde este JSP se ilustran en la *Figura 4.25*.

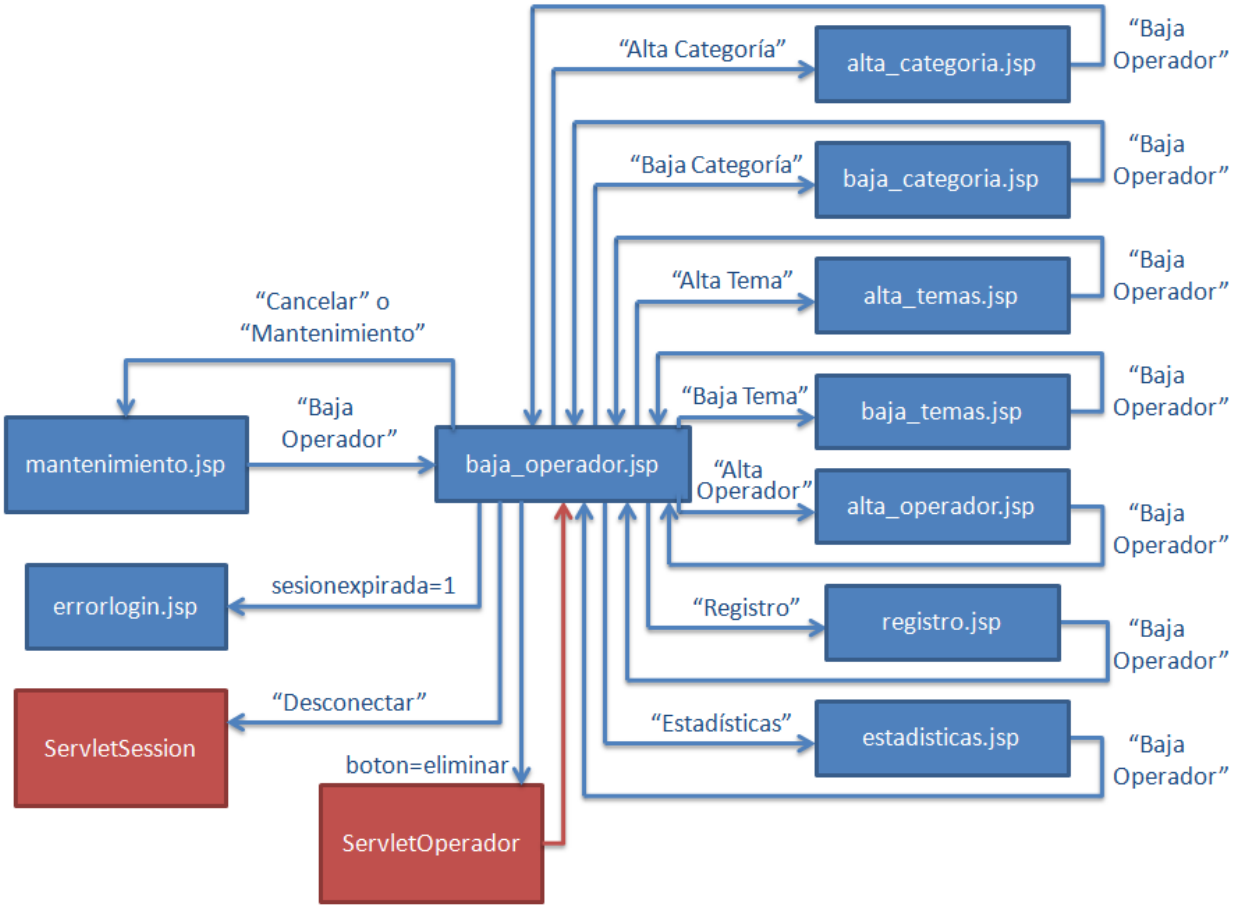


Figura 4.25 Flujo de entrada y salida a baja_operador.jsp

Se puede acceder a la página `baja_operador.jsp` encontrándose la aplicación en uno de los siguientes puntos de ejecución:

- Al igual que sucede con las demás páginas de mantenimiento, si desde la página de registro o desde las páginas de mantenimiento o de estadísticas se pulsa la opción de "Baja operador", se accede por primera vez a esta página, iniciándose un nuevo proceso de baja de un operador.
- También es posible acceder a esta página desde el servlet `ServletOperador`. En este caso el proceso de baja ya ha sido iniciado y gestionado por el servlet, que tras finalizar sus tareas, devuelve el control a esta página.

Esta página recupera de la base de datos de la aplicación todos los usuarios de tipo operador que han sido dados de alta en el sistema.

Según se acceda por una u otra vía, el comportamiento de la página `baja_operador.jsp` variará, mostrando el contenido que se detalla a continuación.

- Si se trata del primer acceso a esta página en el proceso de baja de un operador, se muestra al administrador un menú desplegable que contiene los usuarios de tipo operador pertenecientes al sistema, previamente recuperados. Cuando el administrador selecciona uno de los nombres en este menú, se muestran en el siguiente parámetro del formulario, a modo informativo, los puestos asociados a este operador.

  Una vez que el administrador acepta la operación de eliminación del operador seleccionado, se traspasa el control de la ejecución al servlet `servletOperador`, enviándose el formulario anterior, que también incluye un parámetro oculto indicando que la acción a realizar es baja del operador.

- En caso de que no sea el primer acceso a `baja_operador.jsp` durante la baja de un operador, se provendrá de `ServletOperador`, que almacena el resultado de la operación que ha llevado a cabo en el estado de la aplicación, en este caso la eliminación del operador indicado.

  A. Si dicho resultado es satisfactorio, esta página muestra un mensaje al administrador informándole de que la operación se ha llevado a cabo con éxito.

  B. Si se ha producido algún fallo durante la eliminación del operador de la base de datos del sistema, se muestra un mensaje indicando al administrador que no se ha podido realizar la operación requerida.

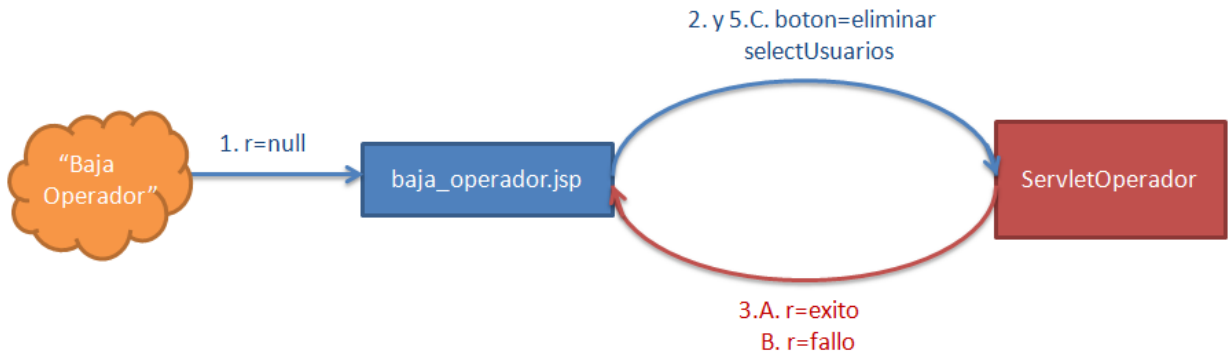


Figura 4.26 Flujo entre baja_operador.jsp y ServletOperador.java

ServletOperador.java

Las peticiones de alta y baja de operadores en el sistema, recibidas desde las páginas `alta_operador.jsp` y `baja_operador.jsp`, son gestionadas por este servlet, como se puede observar el la *Figura 4.27*.

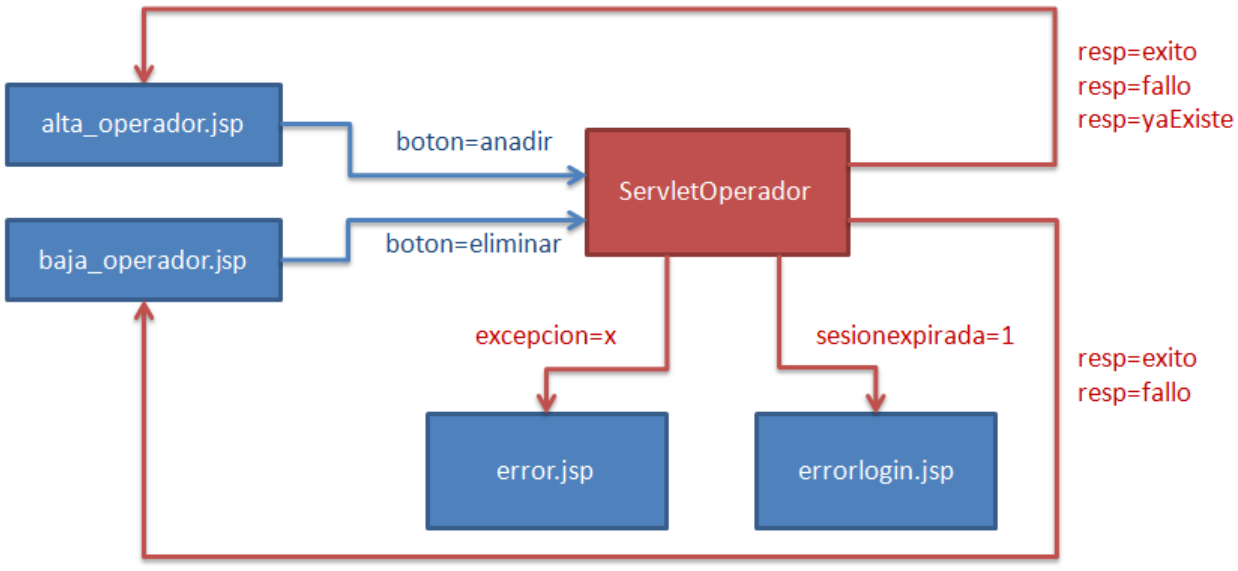


Figura 4.27 Flujo de entrada y salida a ServletOperador.java

En estas páginas JSP se indica en la petición a `ServletOperador.java` en cada caso si se trata de una solicitud de alta o de baja de un usuario, mediante el correspondiente parámetro oculto del formulario.

Alta de operador

En primer lugar, se obtienen del formulario de alta los parámetros que el administrador ha introducido en la página `alta_operador.jsp`: el nombre del operador, su NIF, el nombre de usuario y contraseña y los puestos que va a tener asociados.

A partir de estos datos, se crea un nuevo objeto de tipo "Usuario", que será insertado en la base de datos, siempre y cuando no exista un usuario activo en el sistema con el mismo nombre de usuario. Para realizar esta comprobación, se efectúa una búsqueda entre todos los usuarios del sistema, comparando uno a uno sus nombres de usuario con el nombre de usuario del operador a añadir.

- Si se encuentra un usuario con este nombre de usuario, al igual que con las categorías y los temas, se obtiene su estado.
    - Si el estado del usuario existente es inactivo, para añadir el nuevo usuario al sistema es necesario realizar las siguientes modificaciones en la base de datos. Se trata de tres pasos, que consisten en cambiar el estado de inactivo a activo, actualizar los parámetros del usuario previo con los atributos del usuario que se está añadiendo, y por último, almacenar los puestos del usuario que se está insertando en la tabla correspondiente en la base de datos.
        - Si estas acciones se han podido llevar a cabo satisfactoriamente, el resultado del alta del nuevo operador será exitoso.
        - Si por el contrario se produce algún problema en los accesos mencionados a la base de datos, el resultado indicará que se ha producido un fallo en la operación de alta.
    - En caso de que el usuario existente tenga estado activo, no será posible añadir el nuevo usuario al sistema, reflejando en el resultado de esta búsqueda que el usuario que se desea añadir ya existe en el sistema.
- Si tras la búsqueda no se detecta la existencia de ningún usuario en la base de datos con el mismo nombre de usuario, se incorpora al sistema el nuevo operador. Para ello, se inserta en la base de datos el usuario creado, y se añaden también los puestos asociados a éste.
    - Si es posible realizar dichas acciones en las correspondientes tablas de la base de datos correctamente, el resultado del alta del operador será satisfactorio.
    - El resultado es fallido si surge alguna incidencia en los accesos a la base de datos, o no es posible llevar a cabo las acciones requeridas.

Tras finalizar el proceso de alta del operador solicitado, `ServletOperador` almacena el valor del resultado en el atributo de la petición correspondiente, y devuelve el control de la ejecución a la página `alta_operador.jsp`.

Baja de operador

Para llevar a cabo la baja de un operador se obtiene, a través del parámetro correspondiente del formulario recibido de `baja_operador.jsp`, el usuario de tipo operador que se desea eliminar del sistema.

Por otro lado, se recupera de la base de datos el grupo de usuarios de tipo operador, y se realiza una búsqueda en este conjunto del usuario a suprimir. Una vez encontrado, se procede a la eliminación del operador.

- Si dicho proceso se realiza correctamente, la operación de baja del operador concluye con éxito, reflejándose en el resultado.
- Si no es posible eliminar el operador correctamente de la base de datos, el resultado de la operación de baja indicara que se ha producido un fallo en el proceso de baja.

Concluida la operación de baja, el servlet almacena el resultado del proceso de baja en el atributo correspondiente, e invoca la página `baja_operador.jsp`, que retoma el control de la ejecución de la aplicación.

❖ Estadísticas.

estadisticas.jsp

Esta página muestra al administrador el listado de las posibles estadísticas que el sistema permite calcular, y proporciona el acceso a cada una de ellas.

El flujo de entrada y salida a `estadisticas.jsp` se muestra en la *Figura 4.28*.

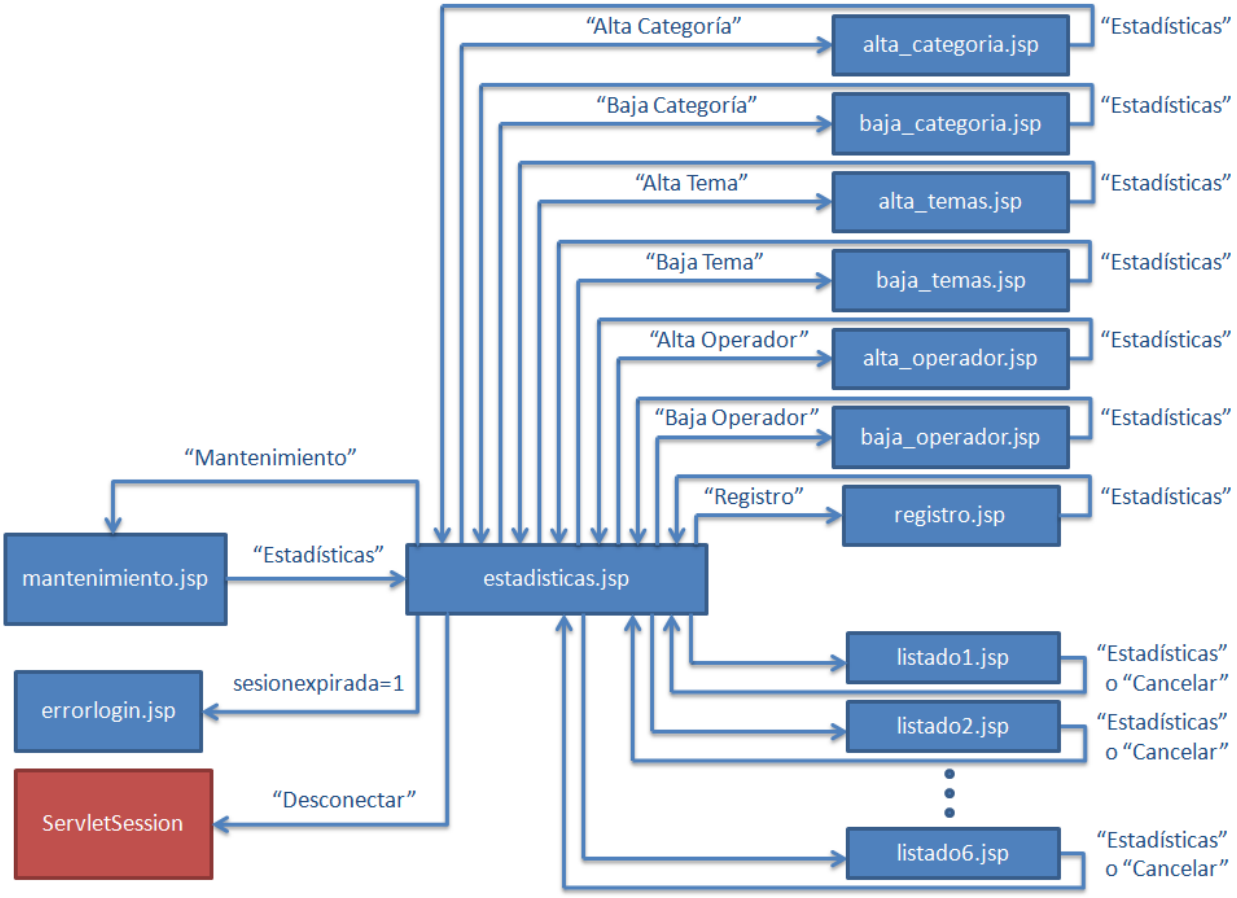


Figura 4.28 Flujo de entrada y salida a estadisticas.jsp

Es posible acceder a esta página desde los siguientes puntos de la aplicación:

- Desde la página de registro y todas las páginas pertenecientes al mantenimiento y a las estadísticas, al seleccionar la opción de Estadísticas, se accede a esta página.
- Desde las páginas pertenecientes al ámbito de las estadísticas, si el usuario cancela la operación, se vuelve a mostrar esta página.

Desde `estadisticas.jsp` se accede a las páginas JSP establecidas en la *Tabla 4.20*, en función de la estadística seleccionada por el usuario:

Página JSP	Estadística
`listado1.jsp`	Asuntos, categorías, y temas más consultados
`listado2.jsp`	Tiempos medios por temas
`listado3.jsp`	Tiempos medios por operario
`listado4.jsp`	Tiempos medios por días
`listado5.jsp`	Tiempos medios por franjas horarias
`listado6.jsp`	Tiempos medios por días de la semana

Tabla 4.20 Equivalencia Página JSP – Estadística relacionada

listadoN.jsp

Estas páginas ofrecen al usuario con funciones de administrador la posibilidad de consultar la estadística asociada a cada listado. En la *Figura 4.29* se ilustran los accesos relacionados con cada `listadoN.jsp`.

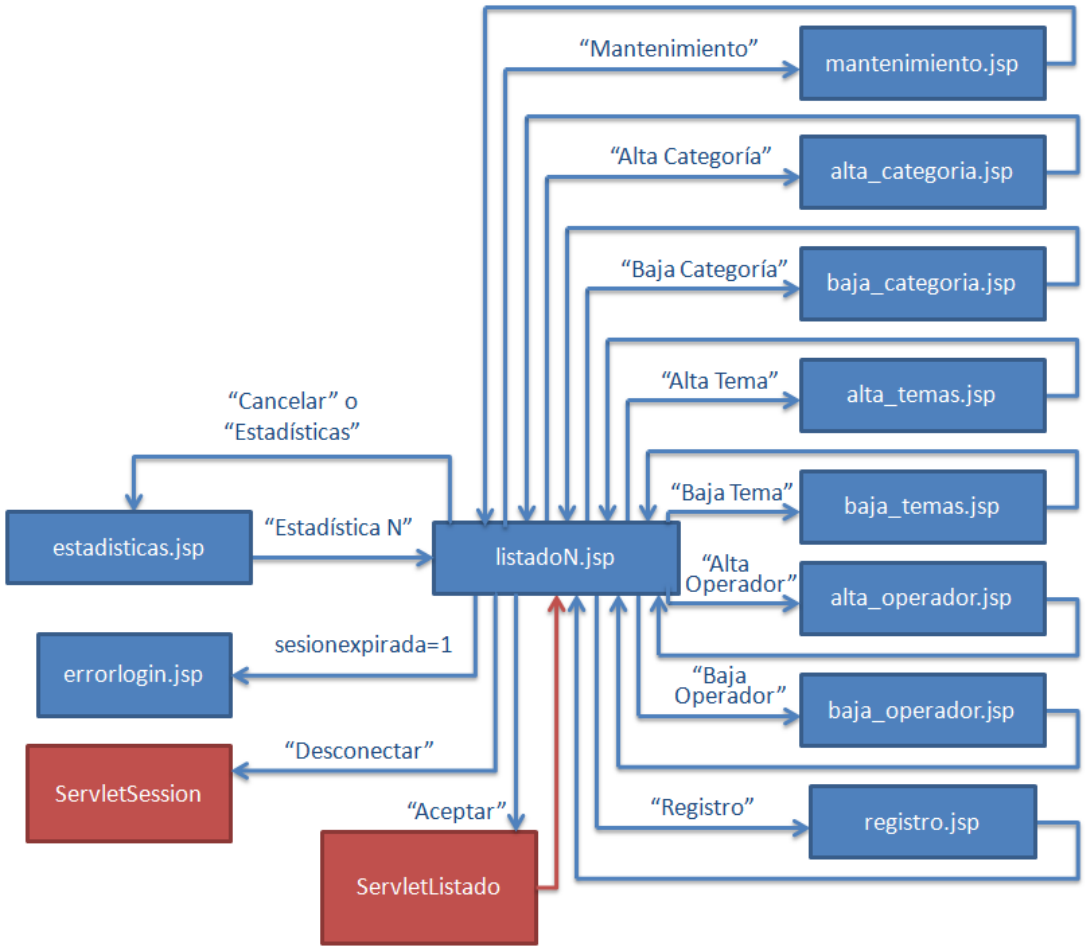


Figura 4.29 Flujo de entrada y salida a listadoN.jsp

Se accede a estas páginas JSP desde dos posibles escenarios en la aplicación:

- Desde la página de estadísticas, al seleccionar cada una de los posibles opciones.
- Si el administrador ya ha iniciado el proceso de consulta de la estadística, tras ser gestionado el cálculo en el servlet `ServletListado`, se vuelve a acceder a esta página.

- Por último, existe el caso en el que desde la propia página `listadoN.jsp` se hace una petición a ella misma. Esto sucede cuando tras mostrarse el resultado obtenido del servlet, el usuario selecciona la opción "Volver". En ese caso, se muestra de nuevo la página de inicio del listado, con el contenido de la consulta previamente realizada cargado en los campos correspondientes.

Las clases `listadoN.jsp` se han implementado siguiendo el mismo esquema, que se describe a continuación.

Se maneja un atributo de estado del sistema cuya función consiste en indicar si se trata del primer acceso a la página, o si por el contrario ya se ha accedido, determinando el contenido a mostrar.

La primera vez que se accede a una página `listadoN.jsp`, se muestra al usuario administrador una página con un formulario, en la que debe indicar los siguientes parámetros:

- Fecha de inicio para el cálculo estadístico.
- Fecha de fin para el cálculo estadístico.
- Puesto que se desea monitorizar.

Esta información se envía al servlet `ServletListado`, además de un parámetro oculto que indica el listado del que se proviene. El servlet la procesa, gestiona los correspondientes accesos a la base de datos y el cálculo de la estadística que corresponde en cada caso, y almacena el resultado de estas acciones en forma de listado, como atributo de la petición, reenviando de nuevo la petición a la página `listadoN.jsp` que lo llamó.

En este segundo acceso a la página JSP, que se detecta mediante el atributo de estado previamente mencionado, en primer lugar se muestra la información relativa al usuario administrador que ha solicitado el cálculo y el puesto que se está monitorizando, junto con los datos sobre su localización dentro de la organización.

Por otro lado se obtiene el listado almacenado por el servlet en el correspondiente atributo y se representa este resultado al usuario en forma de tabla.

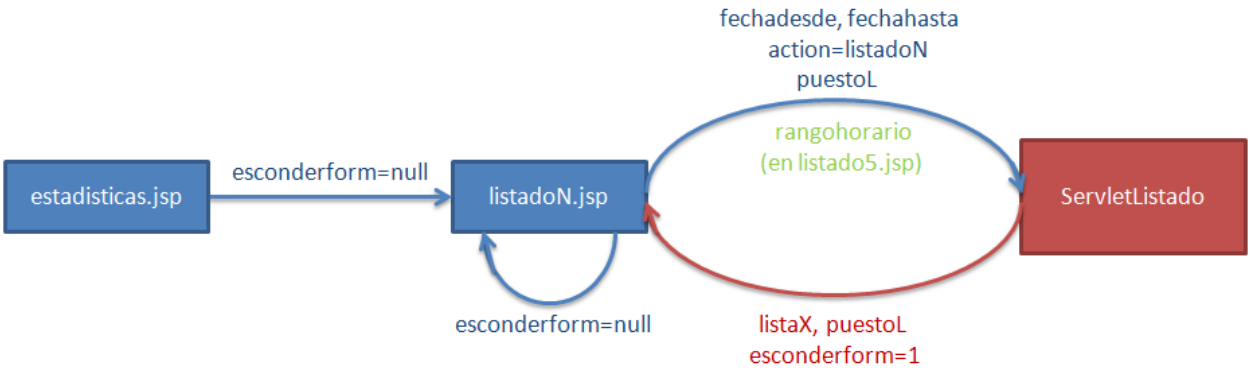


Figura 4.30 Flujo entre listadoN.jsp y ServletListado.java

A continuación se detallan el contenido mostrado por cada estadística, y las particularidades de algunos de los casos.

listado1.jsp: Asuntos, categorías, y temas más consultados

Esta página permite obtener información sobre los asuntos, las categorías y los temas consultados y el número de veces que se ha realizado cada consulta. Este contenido se va obteniendo paso a paso, tras varias consultas al servlet `ServletCategoria`, una por cada segmento de información.

Esta página JSP conlleva más complejidad que las demás, ya que además del esquema descrito anteriormente, permite más de una consulta a `ServletCategoria`. A diferencia del resto de listados, proporciona más de dos posibles vistas para el usuario.

En la *Figura 4.31* se ilustran los diferentes accesos que se pueden realizar desde `listado1.jsp` al servlet `ServletListado`, en el orden que indican los pasos de la *Figura 4.31*.

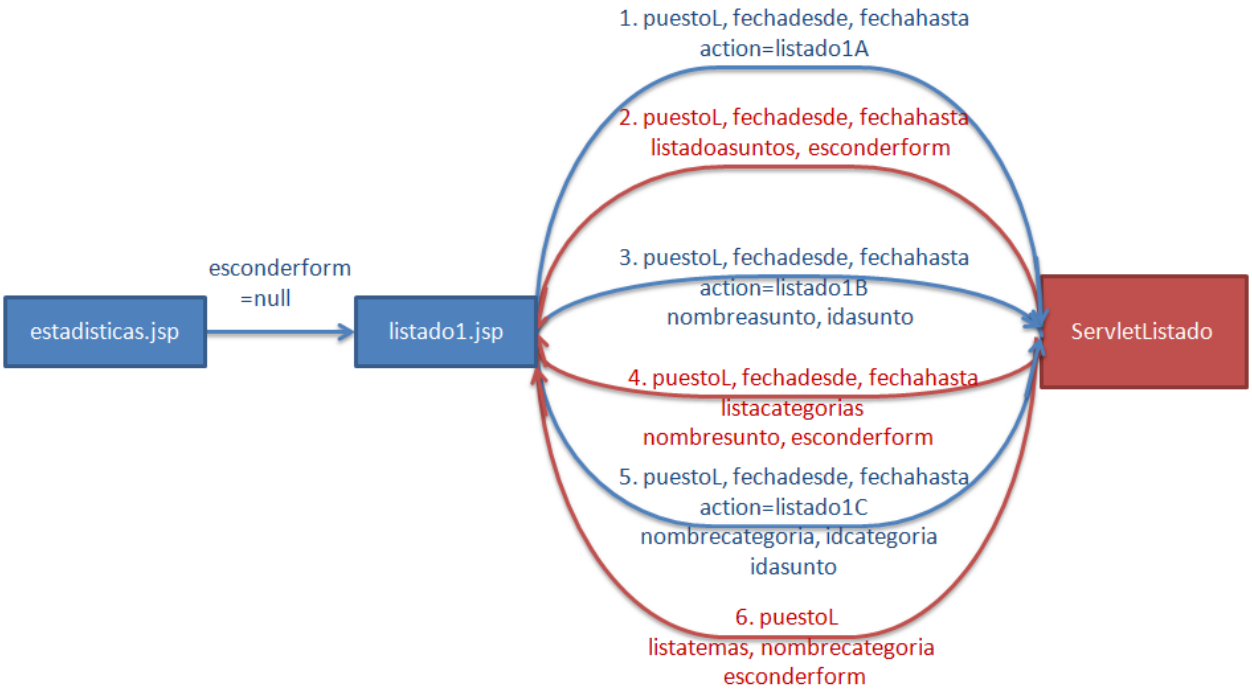


Figura 4.31 Flujo entre listado1.jsp y ServletListado.java

El primer acceso a la página, como se ha indicado para las otras páginas `listadoN.jsp`, muestra al usuario el formulario detallado previamente.

Esta información se envía al servlet `ServletListado`, además de un parámetro oculto que indica que se trata del primer acceso a `ServletListado` desde `listado1.jsp`.

Tras este acceso al servlet, se accede por segunda vez a `listado1.jsp`, donde se muestran los asuntos consultados, que pueden ser de dos tipos, sugerencias y consultas de información, y el número de consultas realizadas de cada tipo.

En el caso de las sugerencias, todas ellas se engloban en un mismo conjunto, pero si se trata de consultas de información, éstas se clasifican en categorías, y las categorías contienen a su vez diversos temas que también pueden ser consultados. Para mostrar al usuario esta información detallada, se proporciona un enlace en el asunto correspondiente a las consultas de información.

Al seleccionar dicho enlace, se accede por segunda vez al servlet `ServletListado`, indicando que se trata del segundo tipo de acceso al servlet desde `listado1.jsp`.

El servlet almacena la información acerca de las categorías consultadas y el número de consultas realizadas a cada una de ellas, y devuelve en control de la ejecución de la aplicación a `listado1.jsp`, que muestra al usuario este contenido.

Finalmente es posible realizar un último tipo de acceso a `ServletListado` desde esta vista, para obtener información acerca de los temas que han sido consultados asociados a una de las categorías y el número de consultas realizadas por cada uno de ellos. Esto será posible en el caso de que hayan sido consultados los temas pertenecientes a categorías mostradas. En aquellas categorías con temas consultados, se proporciona el enlace correspondiente, que envía esta petición a `ServletListado`.

Al obtener `ServletListado` la información relativa a los temas solicitada, y almacenarla en el atributo correspondiente, se lleva a cabo el último acceso a `listado1.jsp`, que muestra los temas consultados de la categoría seleccionada en el anterior acceso a `listado1.jsp`, y el número de consultas realizadas por cada uno de los temas.

En cuanto a la opción ofrecida en cada una de estas vistas de la página que permite al usuario volver a un estado previo, se implementa de forma que se vuelve a cargar la misma página, tal y como se había mostrado en el anterior acceso a ella, es decir, mostrando un nivel de listado más global que el actual.

En el caso de que se trate del primer listado, el listado de asuntos, se vuelve a la página del formulario, con los datos introducidos en la búsqueda anterior en los correspondientes campos.

listado2.jsp: Tiempos medios por temas

Por medio de esta página, el usuario administrador accede a un listado que contiene todos los temas pertenecientes al sistema y el tiempo medio que se ha dedicado a la consulta de cada tema teniendo en cuenta los criterios de búsqueda establecidos.

listado3.jsp: Tiempos medios por operario

En esta página se posibilita al administrador la consulta del tiempo medio que dedica cada operador a realizar un informe. Se proporciona una lista de todos los operadores del sistema y el tiempo medio calculado asociado a cada uno de ellos.

listado4.jsp: Tiempos medios por días

La página `listado4.jsp` ofrece al usuario de tipo administrador la posibilidad de consultar el tiempo medio de atención por días, de forma que presenta un listado de los días incluidos en el periodo seleccionado para la búsqueda y el tiempo medio dedicado a atender a los usuarios en cada uno de esos días, en un puesto indicado o en todos los puestos.

listado5.jsp: Tiempos medios por franjas horarias

Esta página muestra el tiempo medio que dedica un operador a realizar un informe dividido en franjas temporales de una hora de duración, de manera que se pueda consultar en qué momentos del día se dedica más o menos tiempo a atender las llamadas recibidas.

listado6.jsp: Tiempos medios por días de la semana

Esta estadística calcula el tiempo medio dedicado a llevar a cabo una consulta por cada día de la semana, mostrando un listado de los días de la semana y la media por cada uno de los días.

ServletListado.java

Este servlet gestiona las peticiones de las estadísticas solicitadas por las páginas JSP de tipo `listadoN.jsp`, que son las únicas páginas que tienen acceso al servlet, como se puede observar en la *Figura 4.32*.

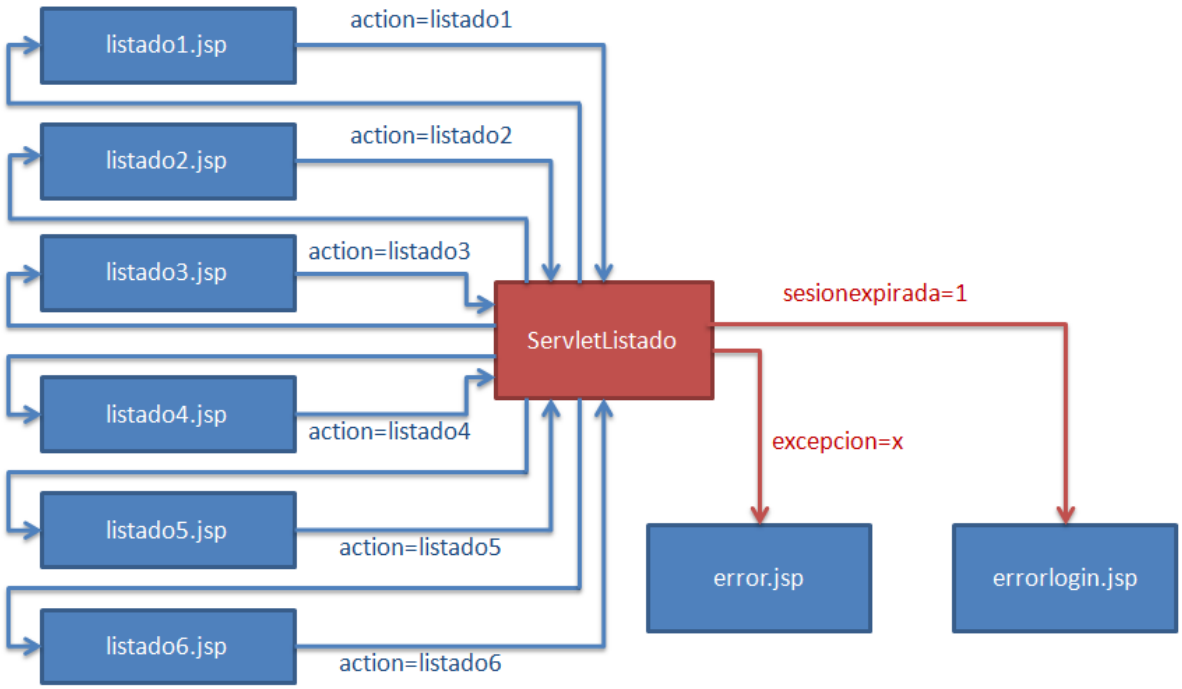


Figura 4.32 Flujo de entrada y salida a ServletListado.java

La función del servlet consiste en recibir la petición, gestionar las correspondientes consultas a la base de datos en función de los parámetros de la búsqueda solicitada, y almacenar el valor del listado en el atributo correspondiente para que pueda ser consultado por la página `listadoN.jsp` que lo llamó.

En primer lugar se obtiene del formulario de la página previa el puesto seleccionado para el cálculo. En caso de haberse seleccionado uno de los puestos pertenecientes al sistema, se recupera el puesto en base a su identificador. Si el administrador ha seleccionado "TODOS LOS PUESTOS", se crea un puesto especial, cuyos atributos indican que se han seleccionado todos los puestos, y por lo tanto todos los grupos, oficinas y entidades existentes.

A continuación se determina la estadística solicitada, en función del correspondiente parámetro recibido en la petición al servlet, para llevar a cabo las acciones correspondientes:

listado1.jsp

En caso de provenir de `listado1.jsp` se requiere diferenciar el punto de ejecución de la aplicación en el que se está accediendo al servlet.

- Si es la primera vez que se accede, se ha solicitado el listado de asuntos consultados.

  Se obtienen las fechas de inicio y fin para el cálculo y, con el puesto correspondiente, se realiza el acceso a la base de datos.

  Para ello se utiliza un objeto de tipo `VistaDAO`, que implementa los correspondientes accesos a la base de datos y los métodos que obtienen las estadísticas a partir de la información extraída. Se obtiene así un listado con los asuntos y el número de veces que se ha consultado cada tipo de asunto.

  Se devuelve a `listado1.jsp` el control de la ejecución, almacenando en los atributos correspondientes los siguientes datos:

  - Las fechas que marcan el inicio y fin del cálculo.
  - El listado de los asuntos.
  - El parámetro que indica a la página JSP que no es el primer acceso a ésta.

- Si se trata del segundo acceso desde `listado1.jsp`, el parámetro a calcular es el listado de consultas de información llevadas a cabo.

  Se sigue el mismo procedimiento que en el caso anterior, recuperando el puesto seleccionado y las fechas de inicio y fin del cálculo. A partir de un objeto `VistaDAO` se realiza la consulta a la base de datos, y se genera el listado con las categorías de información del sistema, indicando el número de veces que ha sido consultada cada una.

  Finalmente se almacena la misma información que en el caso anterior, teniendo en cuenta que en este caso el listado es de las categorías de información consultadas, y se invoca `listado1.jsp`.

- En el caso del último tipo de acceso desde `listado1.jsp`, el cometido del servlet es obtener el listado de los temas asociados a la categoría que se ha seleccionado en la página JSP.

  Al igual que en los casos anteriores, con el puesto seleccionado en listado1.jsp y las fechas de inicio y fin del cálculo estadístico, y mediante el uso de un objeto de tipo `VistaDAO`, se genera, a partir de la información de la base de datos, un listado con los temas pertenecientes a la categoría indicada, obteniendo también el número de veces que se ha consultado cada tema.

  Al finalizar esta acción, se devuelve el control de la ejecución a `listado1.jsp`, almacenando en este caso solamente la información:

  - El listado de los temas.
  - El parámetro que indica a la página JSP que no es el primer acceso a ésta.

No es necesario almacenar de nuevo las fechas de inicio y fin del cálculo estadístico, ya que este es el último tipo de acceso al servlet desde `listado1.jsp`, y no necesita esta información para reenviarla de nuevo al servlet en una futura petición.

listado2.jsp, listado3.jsp, listado4.jsp, listado5.jsp, listado6.jsp

Si la petición al servlet se ha realizado desde otra página `listadoN.jsp` diferente de `listado1.jsp`, las acciones llevadas a cabo por el servlet siguen el mismo patrón, exceptuando el tipo de listado que se solicita.

Además del puesto, se obtienen del formulario los instantes inicial y final para el cálculo de la estadística, y se crea un objeto de tipo `VistaDAO`. Esta clase implementa los métodos que obtienen la información necesaria para el cálculo de las estadísticas, llevan a cabo dicho proceso y devuelven el listado conveniente en cada caso. Por tanto, este objeto llamará en cada caso al método adecuado, conforme a la estadística solicitada, obteniéndose el listado a almacenar en el correspondiente atributo, y reenviar la petición a la página `listadoN.jsp`.

Además se indica mediante el atributo adecuado que se trata del segundo acceso a la página JSP, con lo que se mostrará al usuario el listado obtenido.

❖ Desconexión

ServletSession.java

Este servlet gestiona la desconexión del sistema. Todas las páginas JSP de la aplicación invocan este servlet cuando se ha seleccionado la opción “Desconectar”.

Al ser invocado, el servlet invalida la sesión actual, redirigiendo la aplicación a su inicio, la página de autenticación, de forma que un usuario pueda iniciar una nueva sesión.

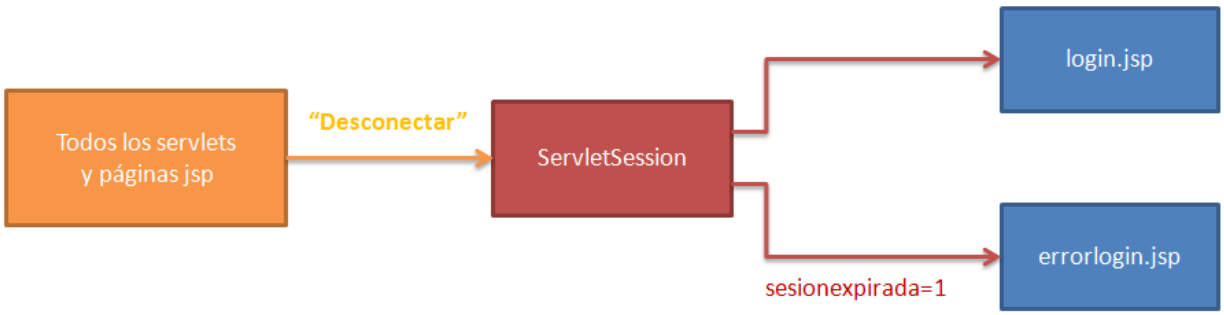


Figura 4.33 Flujo de entrada y salida a ServletSession.java

4.2. ESQUEMA DE ACCESOS EN FUNCIÓN DEL PERFIL

Aunque ya se ha indicado durante la descripción de la implementación del sistema, a continuación se detallan las diferencias en el acceso a las páginas JSP por parte de los diferentes tipos de usuario, según su perfil.

Usuarios administrador-operador y administrador

Ambos acceden a `mantenimiento.jsp`.

En todas las páginas JSP a las que acceden los usuarios administrador-operador y administrador se gestiona la diferencia entre ambos perfiles de usuario, de manera que se cumpla lo siguiente.

En la cabecera de cada una de las páginas, para el usuario administrador-operador, se muestran las opciones de registro y administración:

- Mantenimiento
- Registro
- Estadísticas

En el caso del usuario con perfil de administrador, en la cabecera de las páginas se muestran únicamente las opciones de administración:

- Mantenimiento
- Estadísticas

Usuario operador

Accede a menuRegistro.jsp, desde donde es redirigido a registrosolo.jsp

En esta página JSP se le ofrece la opción que le corresponde a su perfil:

- Registro

Usuarios operador y administrador-operador

Tanto en `mantenimiento.jsp` como en `menuRegistro.jsp` aparece la opción “Registro”, para los usuarios de tipo operador y administrador-operador respectivamente, pero la página JSP del registro es diferente en cada caso.

Para el usuario con perfil administrador-operador se hace una petición a `registro.jsp`, y en el caso del usuario operador, la página que se solicita es `registrosolo.jsp`.

5. RESULTADO OBTENIDO

Tras las acciones previas de estudio de las tecnologías involucradas, la planificación del diseño y el desarrollo de la implementación, se obtiene como resultado el sistema que se expone a continuación, detallando cada fase y las diferentes posibilidades de acción.

El acceso a la aplicación web se realiza a través de la url: "http://servidor/CantabriaPFC/", donde "servidor" será el nombre del servidor en el que se aloje la aplicación.

En función del perfil del usuario que utiliza la herramienta se le ofrecen diferentes opciones, las que corresponden a cada tipo de usuario.

A continuación se hace distinción entre los diferentes tipos de usuarios y su acceso a la aplicación. Como se ha detallado previamente, los perfiles de los usuarios del sistema son los siguientes:

- Perfil 1: Usuario de tipo Administrador – Operador
- Perfil 2: Usuario de tipo Administrador
- Perfil 3: Usuario de tipo Operador

Ya que uno de los objetivos principales de la herramienta consiste en utilizar los datos recopilados en los informes de los operadores para luego ser analizados por la administración, se muestran las fases de la aplicación en el caso de ser utilizada por un usuario de tipo operador, para después detallar el comportamiento en caso de ser un usuario de tipo Administrador. Por último se analiza el caso del usuario con ambas funciones, mostrando todas las posibilidades que se le ofrecen.

Hay determinadas fases de la aplicación que son comunes a los diferentes tipos de usuario. Si el comportamiento varía con el perfil del usuario, se detallan en el apartado dedicado a cada uno de ellos. Si por el contrario el comportamiento de una de estas fases de la aplicación es independiente del tipo de usuario, se detalla de manera conjunta al final de este apartado.

5.1. USUARIO DE TIPO OPERADOR

❖ Autenticación.

La página a la que se accede al inicio de la aplicación es la de autenticación del usuario, que se muestra en la *Figura 5.1*.

Figura 5.1 Página de Autenticación

Esta página es común para todo usuario de la aplicación; en este caso se han introducido las credenciales correspondientes a un usuario del sistema de tipo operador.

Si el usuario de tipo operador está registrado en el sistema pero por error no tiene asignado ningún puesto, no puede acceder al sistema y debe comunicarse con un administrador del servicio para comunicarle su problema. Esta información se le ofrece en un mensaje de error, como se puede ver en la *Figura 5.2*.

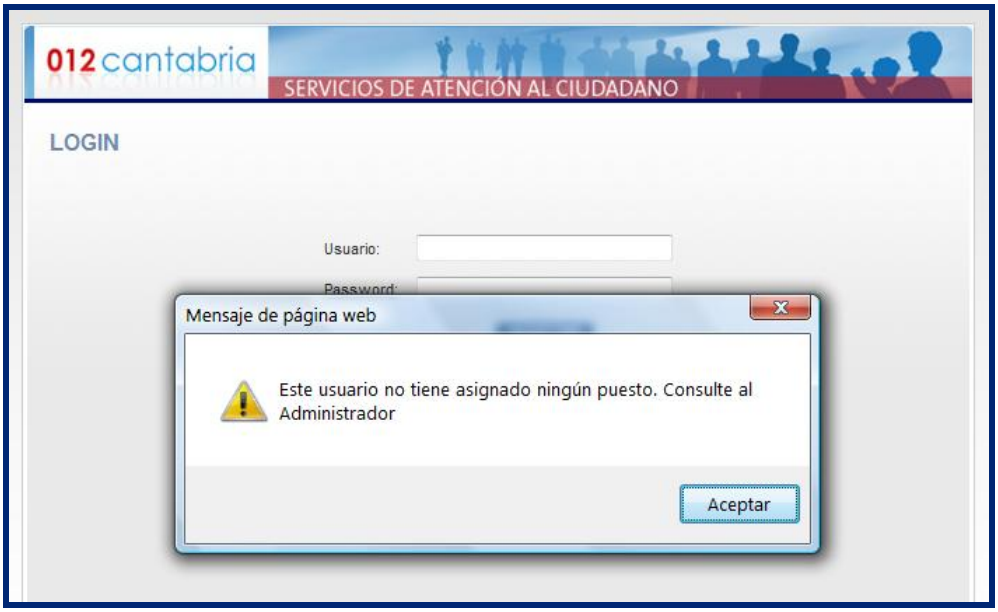


Figura 5.2 Mensaje de Error: Operador sin puesto asignado

❖ Selección de puesto.

Una vez introducidas las credenciales, si el operador puede utilizar más de un puesto, se le muestra la pantalla capturada en la *Figura 5.3*, en la que debe seleccionar el puesto que va a utilizar. Para ello se le ofrece una lista desplegable que contiene los puestos que le corresponden.

Figura 5.3 Página de Selección de puesto

Si al usuario le corresponde un solo puesto, se establece para la sesión el puesto que tiene asociado, sin mostrársele esta pantalla de puestos.

❖ Opción de registro.

Una vez se ha determinado el puesto del operador, se muestran al usuario las opciones de la herramienta a las que tiene acceso. Al tratarse de un usuario de tipo operador puede acceder a la opción de registro.

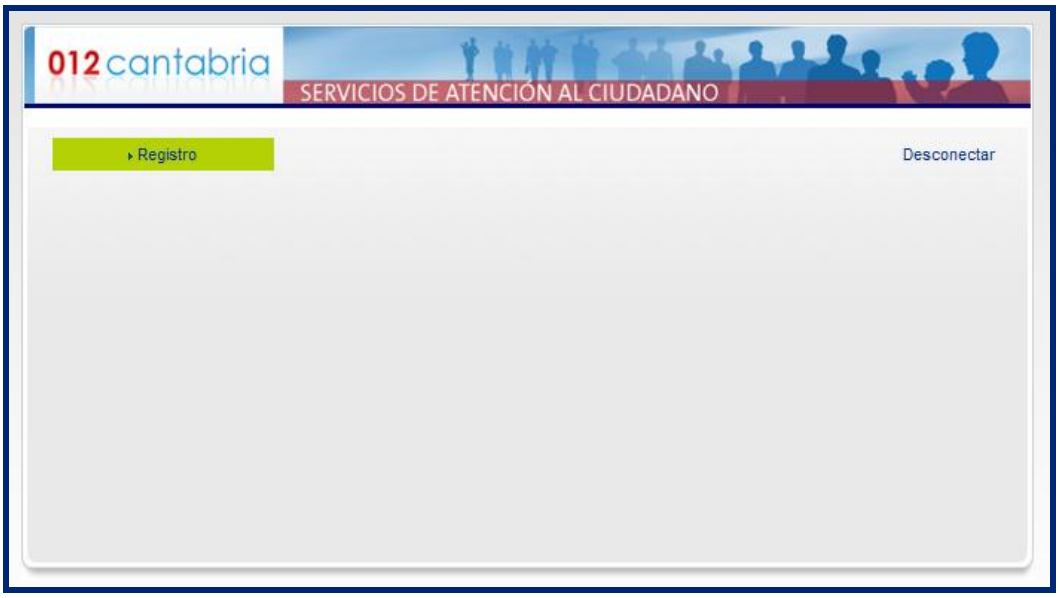


Figura 5.4 Página de Opción: Registro

El operador elegirá dicha opción cuando reciba una llamada telefónica, para poder registrar la información relacionada con el servicio prestado.

❖ Registro.

Cuando el operador selecciona esta opción, se muestra la pantalla que contiene el formulario de registro, que se puede observar en la *Figura 5.5*.

Se presentan al inicio los datos del operador, que la aplicación obtiene de las propiedades del usuario y el puesto establecido para la sesión. Estos datos de identificación del usuario serán registrados como parte del formulario a la hora de almacenar la información relacionada con cada informe.

Así mismo, se presenta la información relacionada con la fecha del informe y la hora de inicio, que se actualiza en el momento en que el operador selecciona el botón “INICIO LLAMADA”.

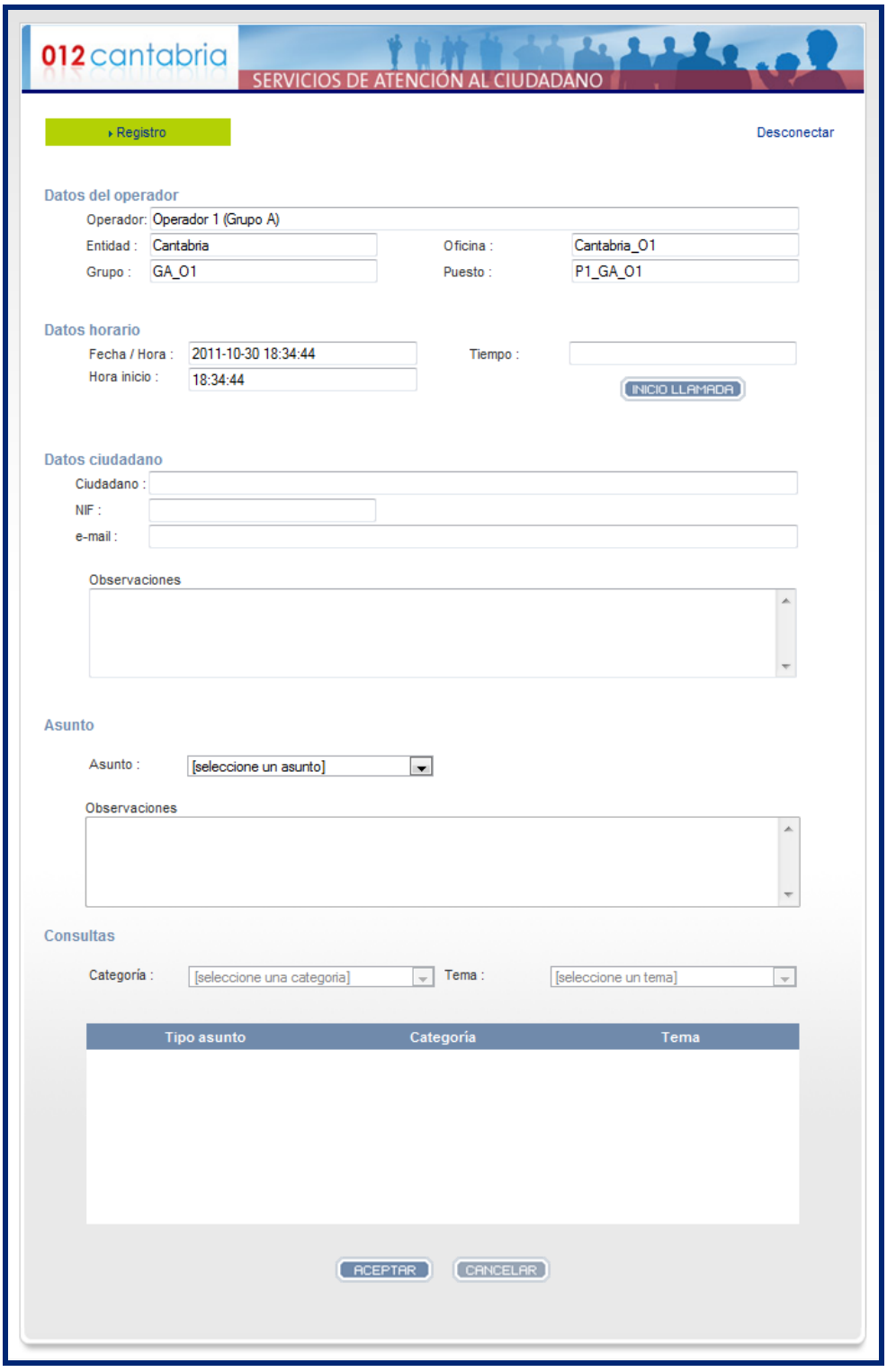


Figura 5.5 Página de Registro

A partir de este instante, el operador puede comenzar a rellenar el formulario. Si antes de seleccionar el botón “INICIO LLAMADA” el operador intenta rellenar los campos del formulario, se le indica que debe iniciar la llamada mediante el mensaje capturado en la *Figura 5.6*.

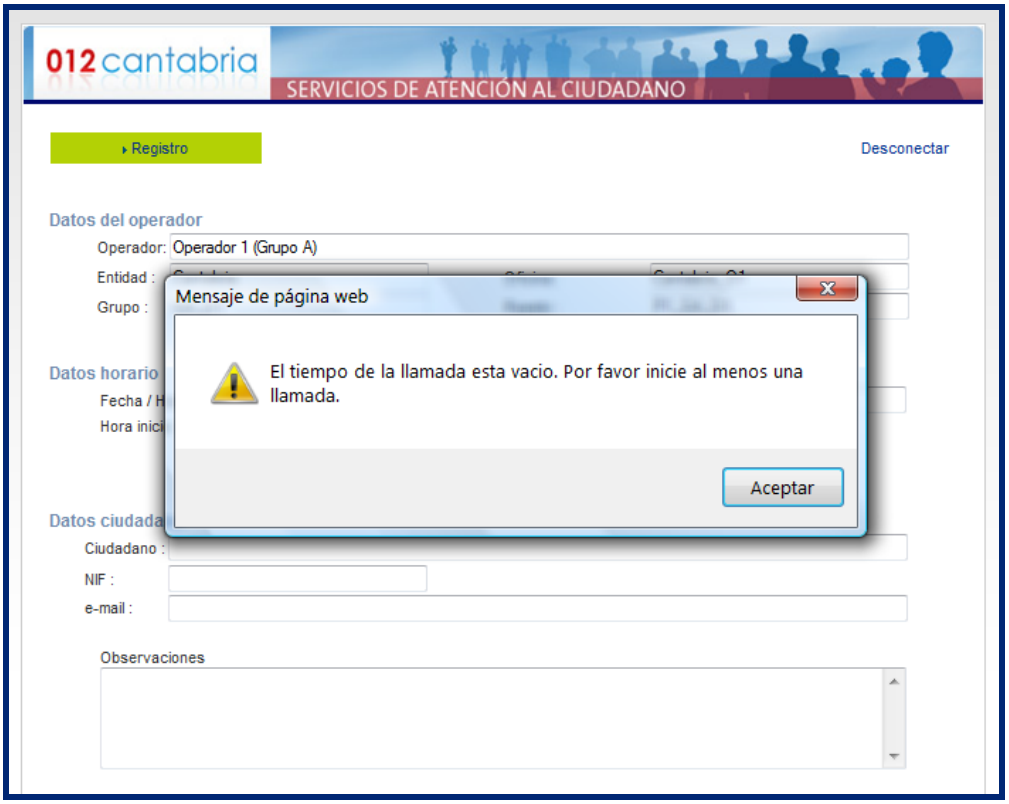


Figura 5.6 Mensaje de advertencia: Inicio de llamada

Una vez iniciada la llamada, no es posible actualizar el valor del tiempo de inicio para esa misma llamada. Si el operador vuelve a seleccionar el botón “INICIO LLAMADA”, se le informa de que la operación ya ha sido iniciada y debe completar el proceso a través del mensaje de alerta mostrado en la *Figura 5.7*.

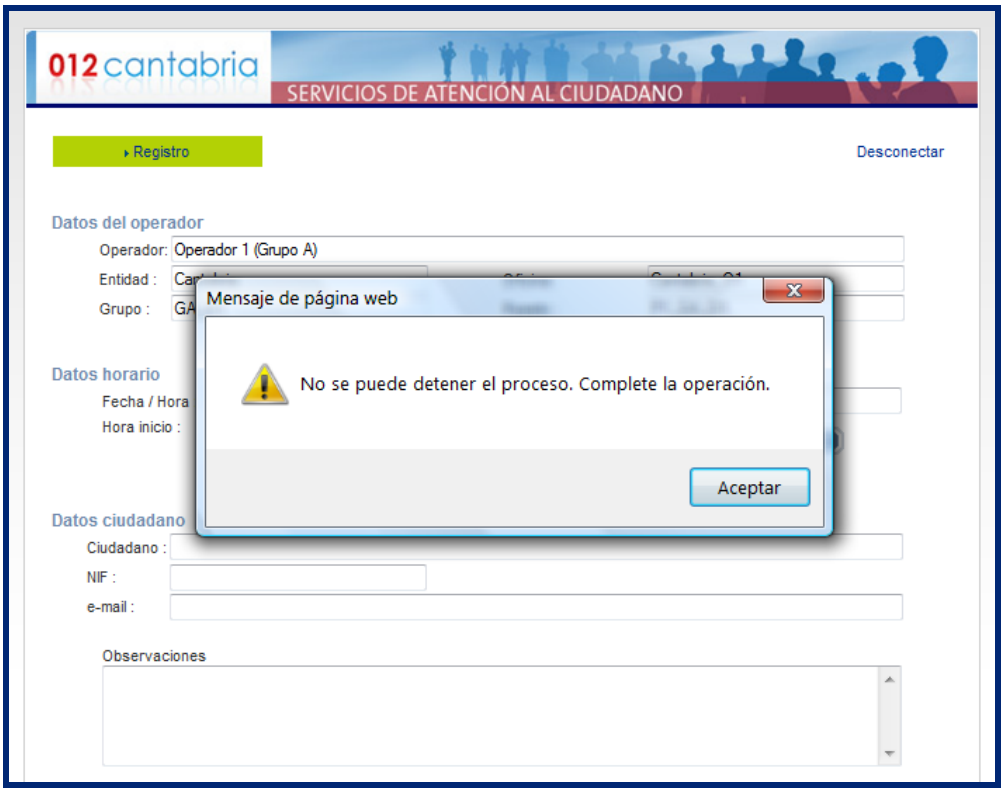


Figura 5.7 Mensaje de advertencia: Operación iniciada

La sección del formulario dedicada a los datos del ciudadano se puede rellenar por el operador de manera opcional, en caso de que esta información sea proporcionada durante la llamada.

En la sección “Asunto”, el operador debe seleccionar un tipo de asunto del menú desplegable que se le ofrece, indicando si la llamada que está atendiendo trata de una consulta de información o una sugerencia por parte del ciudadano.

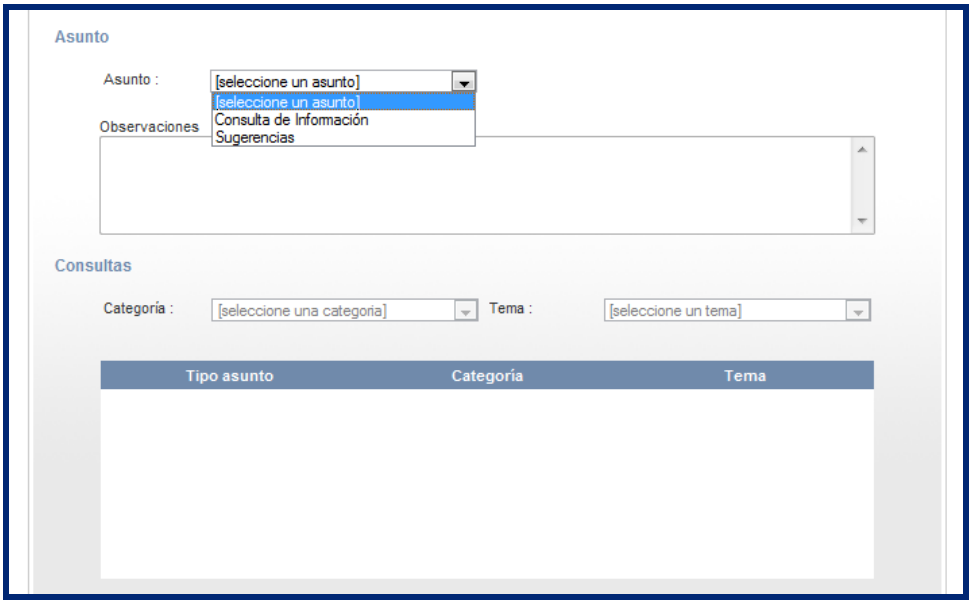


Figura 5.8 Selección de Asunto

En el caso de tratarse de una sugerencia, el operador dispone a continuación del campo “Observaciones” para plasmar el resumen de la idea transmitida por el ciudadano, tras lo cual se daría por finalizado el informe relacionado con la llamada telefónica.

Si el asunto relativo a la llamada es una consulta de información, el operador procede a consultar la información solicitada en la sección “Consultas”, cuyo menú desplegable “Categoría” se desbloquea para que el usuario pueda escoger una de las categorías ofrecidas, como se muestra en la *Figura 5.9*.

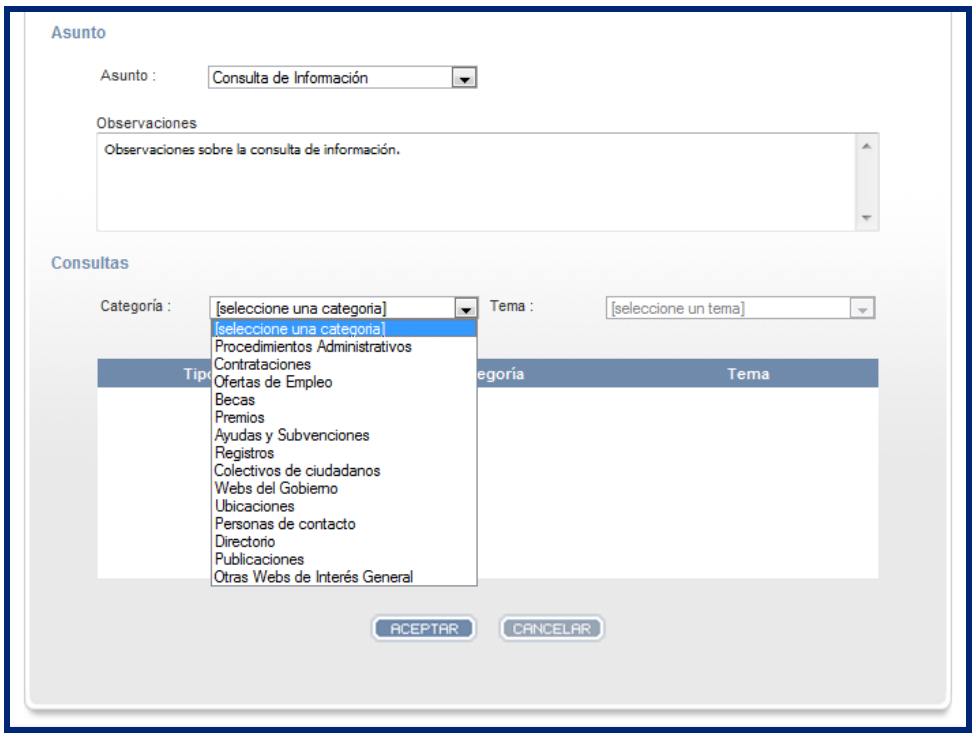


Figura 5.9 Selección de Categoría

Tras la selección de la categoría, el operador elige si desea concretar un tema sobre el que obtener información dentro de la categoría seleccionada.

Si el operador no selecciona un tema, no obtendrá información, por lo que en un caso normal de utilización de la herramienta, contestará afirmativamente a la pregunta que se muestra en la *Figura 5.10*. Se ofrece la posibilidad de no concretar un tema, para casos particulares que puedan surgir, como por ejemplo si el operador, que conoce o ha consultado los temas asignados a una categoría, detecta la necesidad de añadir un nuevo tema. En este caso, seleccionaría la categoría que engloba el tema, indicando en el campo de las observaciones la necesidad de añadir el nuevo tema.

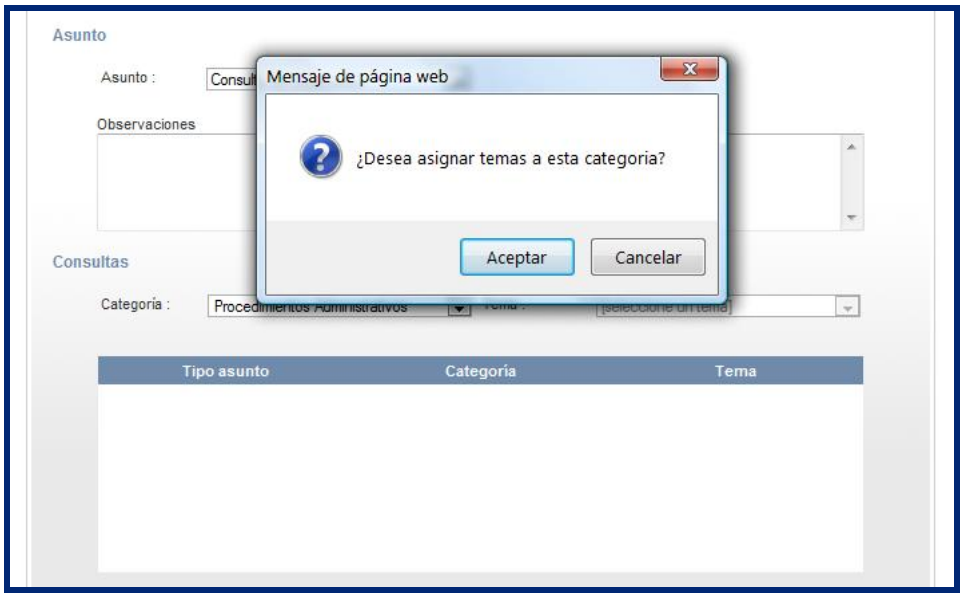


Figura 5.10 Mensaje interrogatorio: Selección de Tema

Por tanto, como se ha indicado, el operador debe seleccionar el tema sobre el que el usuario desea obtener información, y tras aceptar el mensaje de la *Figura 5.10*, se habilita el menú desplegable en el que se ofrecen los temas relacionados con la categoría seleccionada. Esta acción se muestra en la *Figura 5.11*.

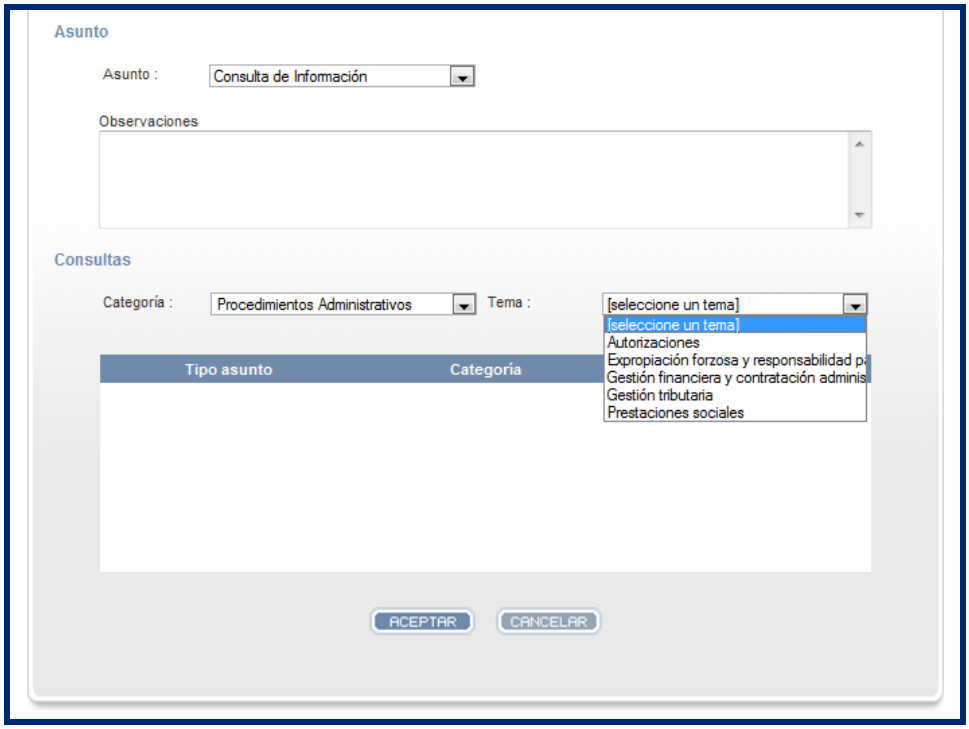


Figura 5.11 Selección de Tema

Una vez seleccionado el tema consultar, el usuario debe confirmar la selección, como se puede observar en la *Figura 5.12*. Esto permite al operador modificar el tema seleccionado si se ha confundido, indicando que desea cancelar la operación.

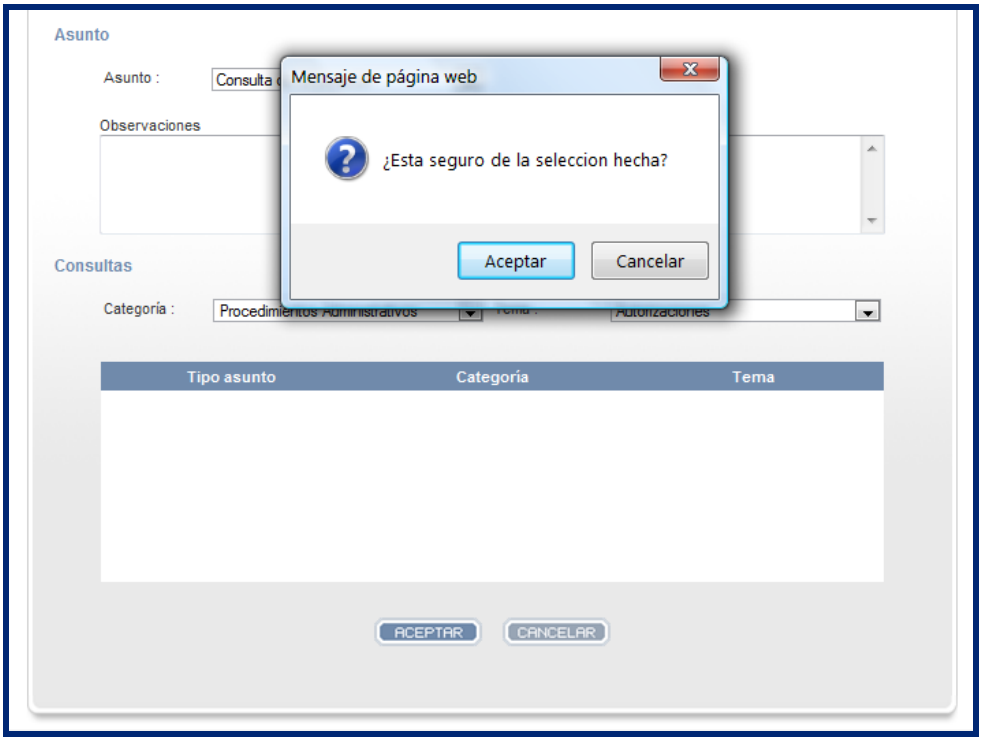


Figura 5.12 Mensaje de confirmación: Selección de Tema

Si el usuario confirma la selección del tema realizada, se abre en otra ventana del explorador la URL asociada a la ruta donde se encuentra la información relativa al tema solicitado.

Además, en la página del registro, se añade en la tabla una nueva entrada, indicando los datos relativos a la consulta realizada, como se puede comprobar en el ejemplo de la *Figura 5.13*.

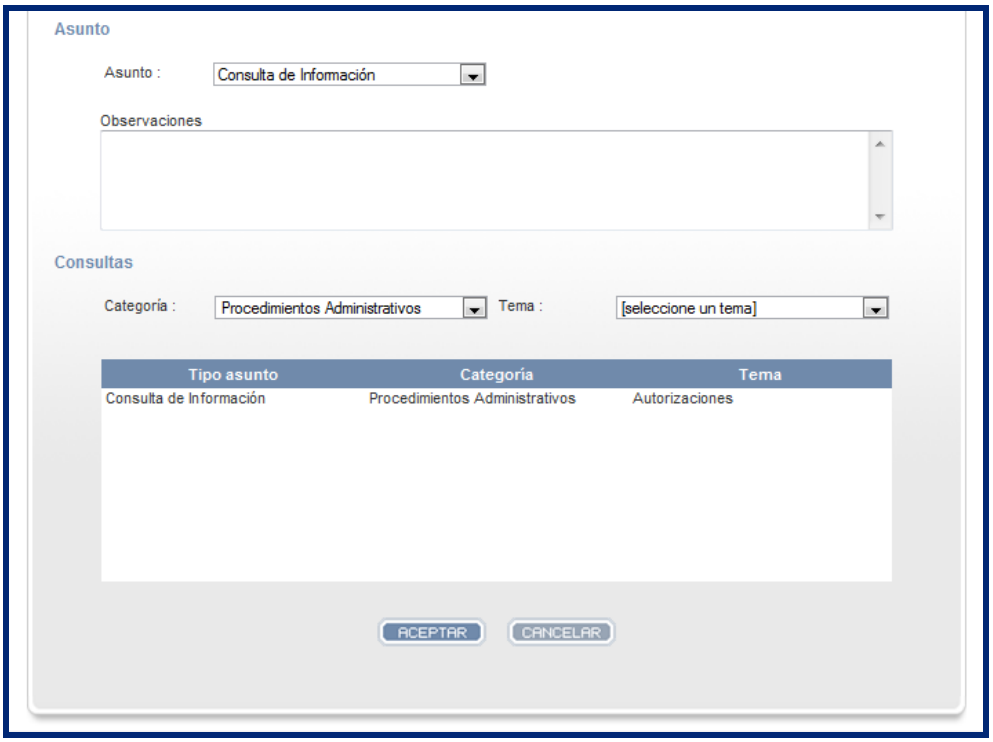


Figura 5.13 Selección realizada

Es posible realizar varias consultas en un mismo informe, basta con volver a seleccionar una categoría en el menú desplegable correspondiente, iniciando así un nuevo proceso de consulta.

Cuando el operador da por finalizado el registro, seleccionando la opción “ACEPTAR”, se realizan varias comprobaciones previas al envío del formulario. Si alguno de los requisitos no se satisface, se indicará al operador que debe modificar los campos que no son válidos, por los motivos que se indican a continuación.

En primer lugar, si se han rellenado los campos correspondientes al NIF o al correo electrónico o “email” de la sección de “Datos del ciudadano”, se comprueba que la información introducida cumple con el formato adecuado en cada caso.

Si el NIF no está formado por una cadena de caracteres consistente en 8 números y su correspondiente letra, se muestra el mensaje de advertencia capturado en la *Figura 5.14*.

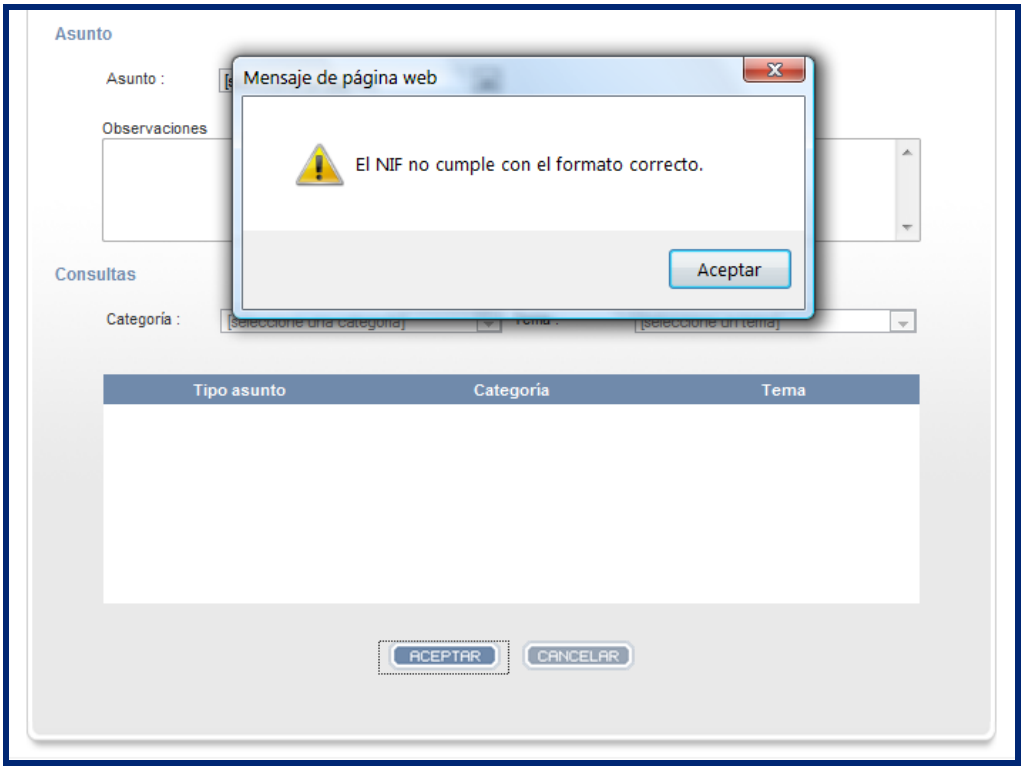


Figura 5.14 Mensaje de advertencia: Formato incorrecto de NIF

Así mismo, si el correo electrónico tiene un formato que no se corresponde con el de una dirección de correo electrónico, formado por una cadena de caracteres seguida del símbolo “@” y a continuación otro grupo de caracteres separados por un punto, se muestra el mensaje de advertencia mostrado en la *Figura 5.15*.

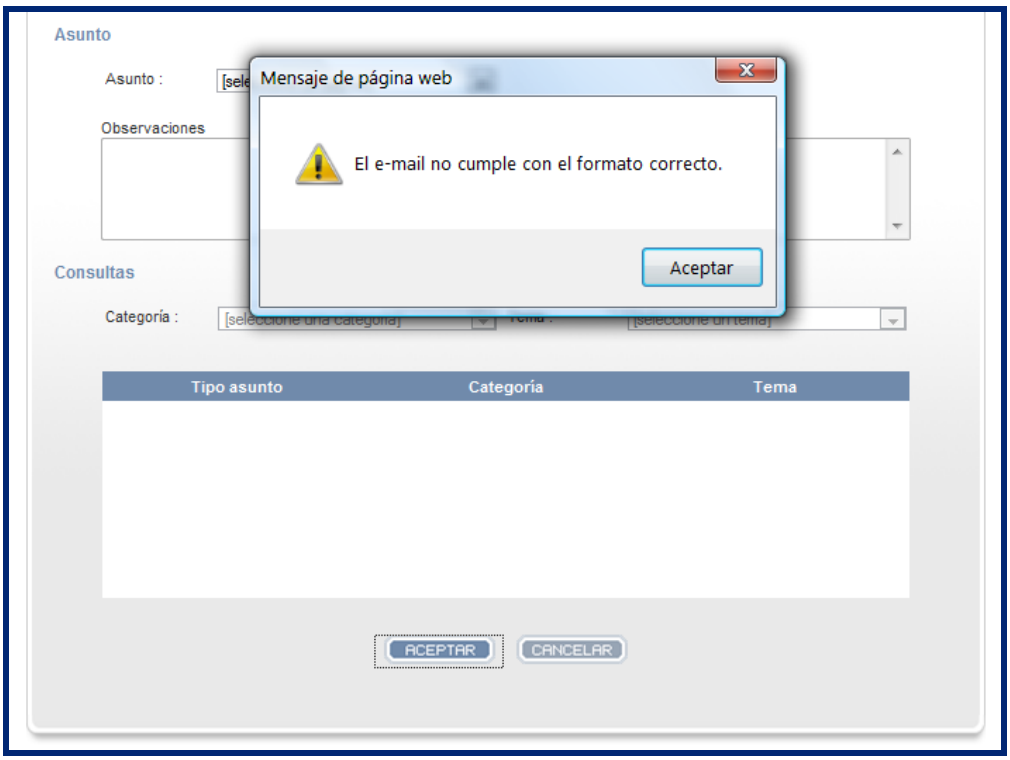


Figura 5.15 Mensaje de advertencia: Formato incorrecto de correo electrónico

Por otro lado, si el asunto seleccionado por el operador es una sugerencia, debe introducir el texto asociado a la sugerencia recibida en el campo “Observaciones”. De lo contrario no podrá enviar el formulario, y ello se le indica en el mensaje que aparece en la *Figura 5.16*.

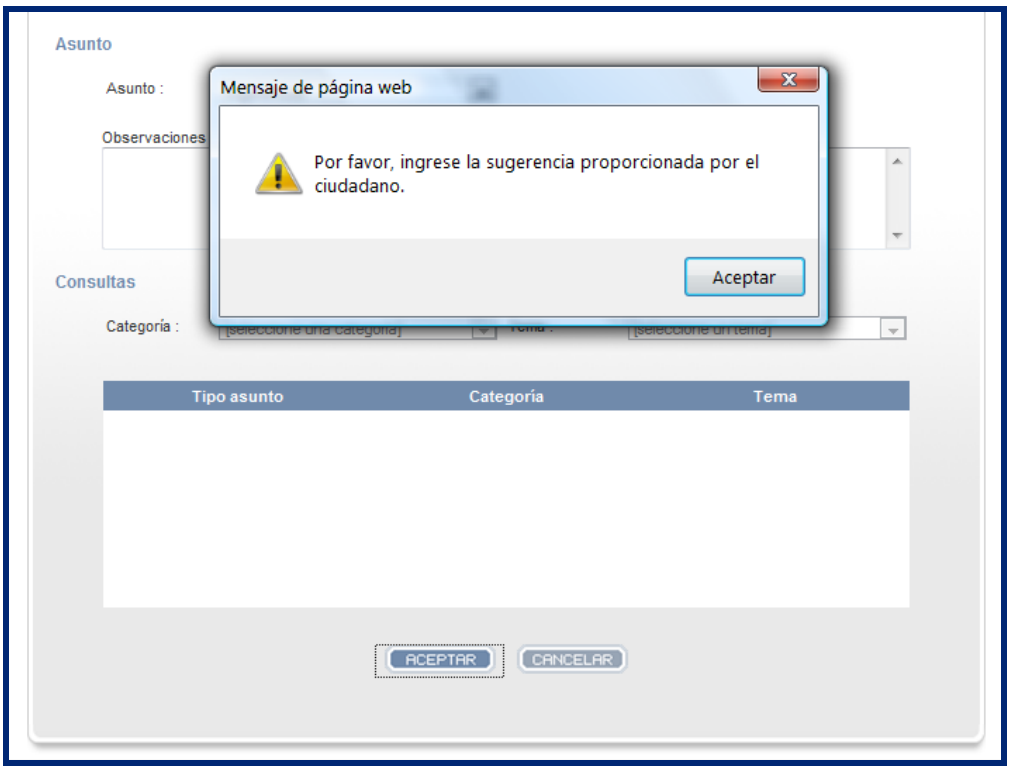


Figura 5.16 Mensaje de advertencia: Sugerencia no introducida

De igual manera, en el caso de que el operador elija realizar una consulta de información, si se dispone a validar el registro sin haber realizado dicha consulta, recibirá un mensaje del sistema indicándole que debe escoger una categoría, como se puede observar en la *Figura 5.17*.

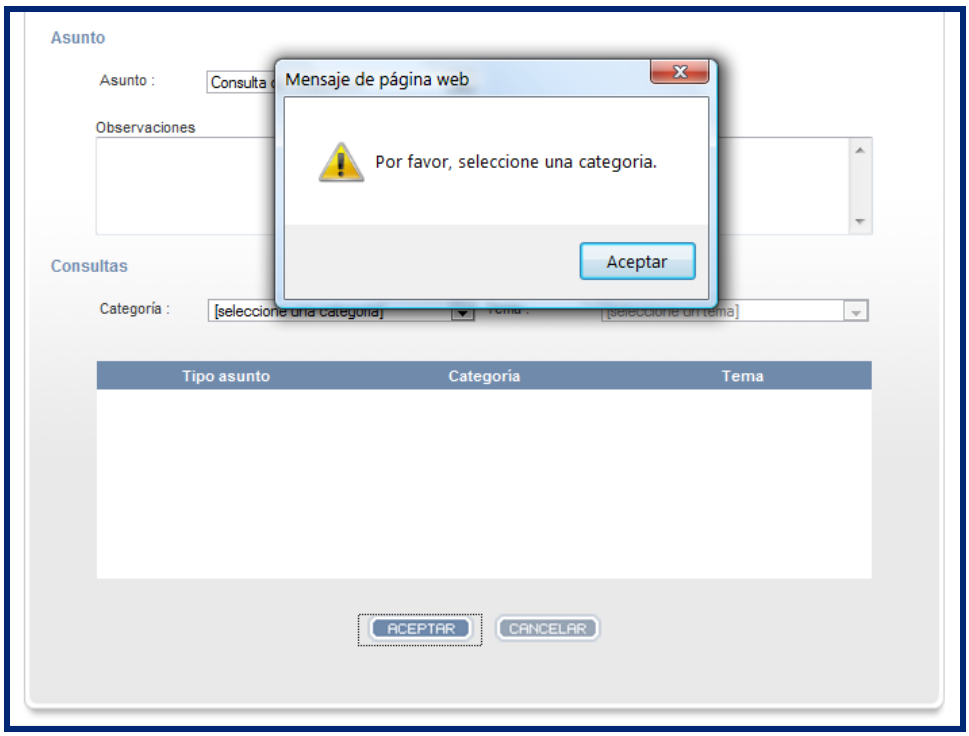


Figura 5.17 Mensaje de advertencia: Categoría no seleccionada

Una vez realizadas estas comprobaciones, cuando se cumplen los requisitos indicados, es posible enviar el registro realizado. La aplicación muestra un mensaje en el que indica que la acción se ha llevado a cabo de manera satisfactoria, ofreciendo a continuación un nuevo informe al operador, como se muestra en la *Figura 5.18*.

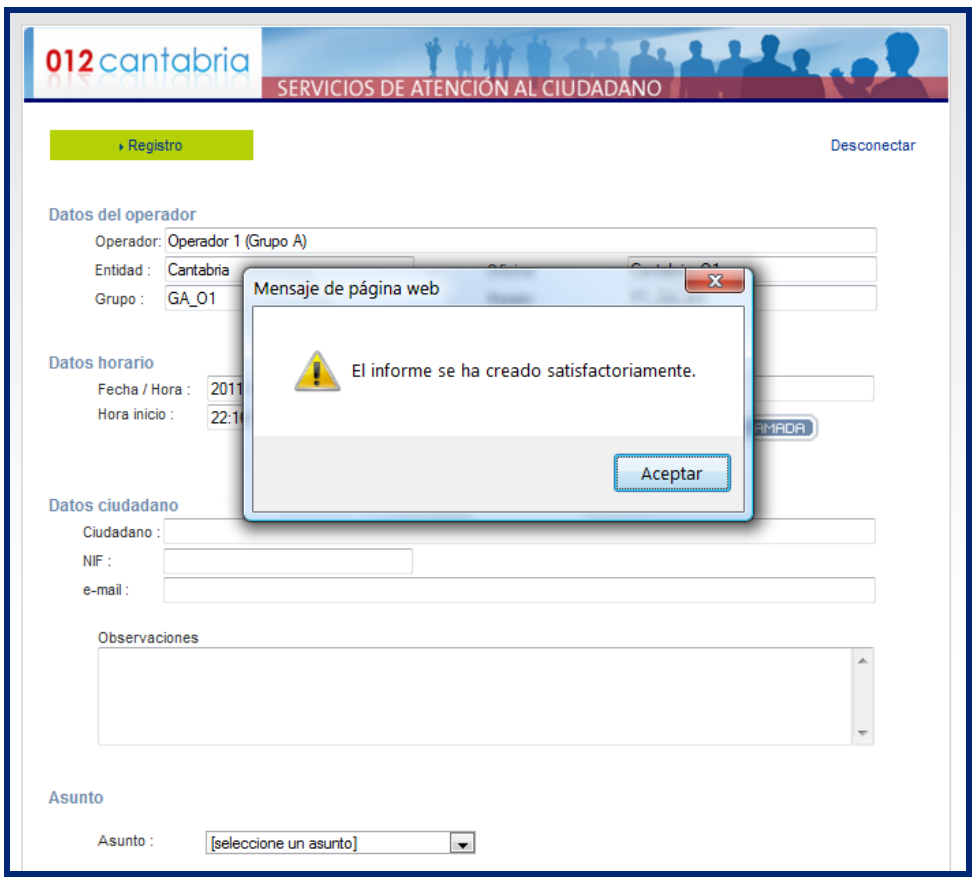


Figura 5.18 Mensaje informativo: Informe creado

5.2. USUARIO DE TIPO ADMINISTRADOR

❖ Autenticación.

La página de autenticación es común para todos los usuarios del sistema. A continuación, en la *Figura 5.19*, se muestra un ejemplo de autenticación de un administrador del sistema.

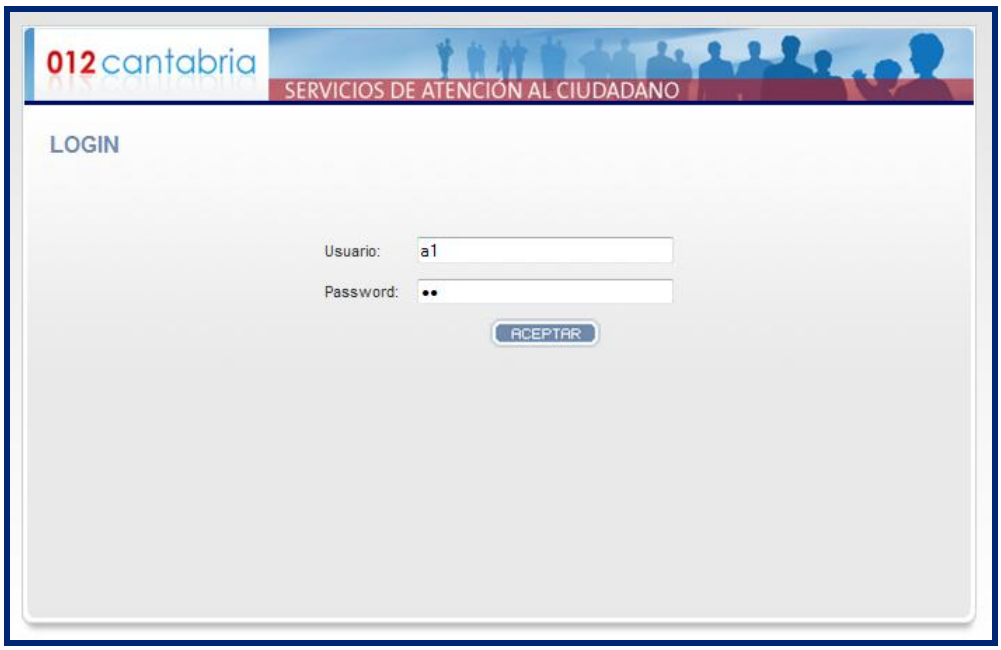


Figura 5.19 Página de Autenticación

A un usuario de tipo administrador no se le asignan puestos del sistema, ya que puede realizar sus funciones desde cualquier puesto, y estas actividades no necesitan del registro de un puesto concreto como en caso de los operadores.

❖ Opciones de administración.

Una vez se ha autenticado al administrador, se le ofrecen las opciones de la aplicación que corresponden a su perfil. En este caso se trata de las opciones de administración, que consisten en el mantenimiento de la herramienta y en la consulta de estadísticas generadas por el sistema, como se puede observar en la *Figura 5.20*.

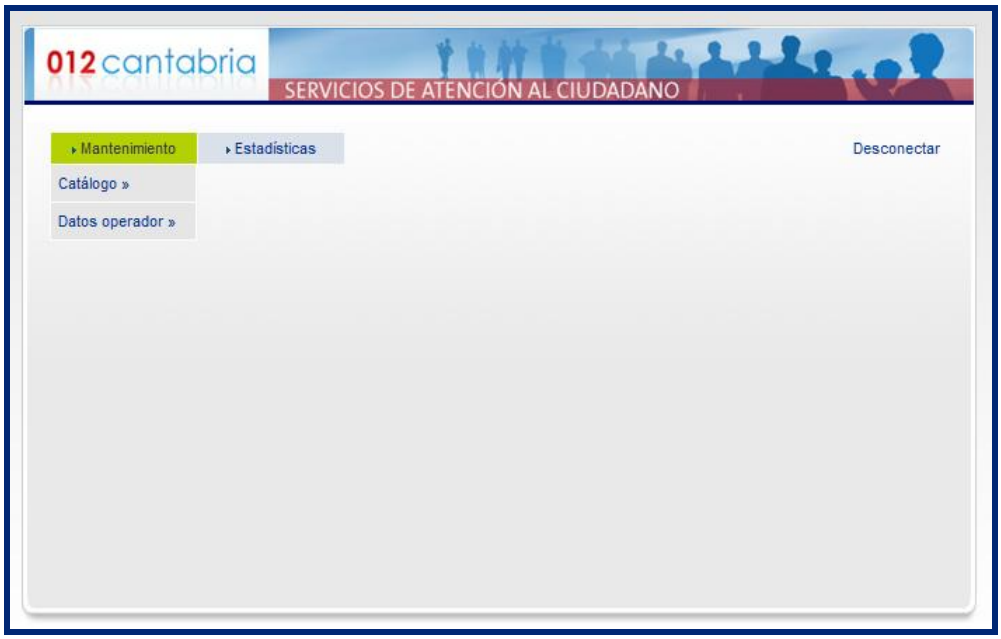


Figura 5.20 Página de Opciones de administración

❖ Opciones de mantenimiento.

En cuanto al mantenimiento de la aplicación, el administrador puede gestionar el contenido del catálogo de información consultado por los operadores telefónicos para prestar el servicio de atención ciudadana, y los componentes del conjunto de operadores con acceso a esta herramienta.

El catálogo de información ofrecido para su consulta está formado por un conjunto de categorías, que a su vez se subdividen en grupos de temas. El administrador puede añadir y eliminar del sistema tanto categorías como temas.

Inicialmente se muestra cómo añadir y eliminar categorías en el sistema. La imagen de la *Figura 5.21* se obtiene colocando el ratón sobre los menús indicados, cuyo fondo se presenta en color gris.

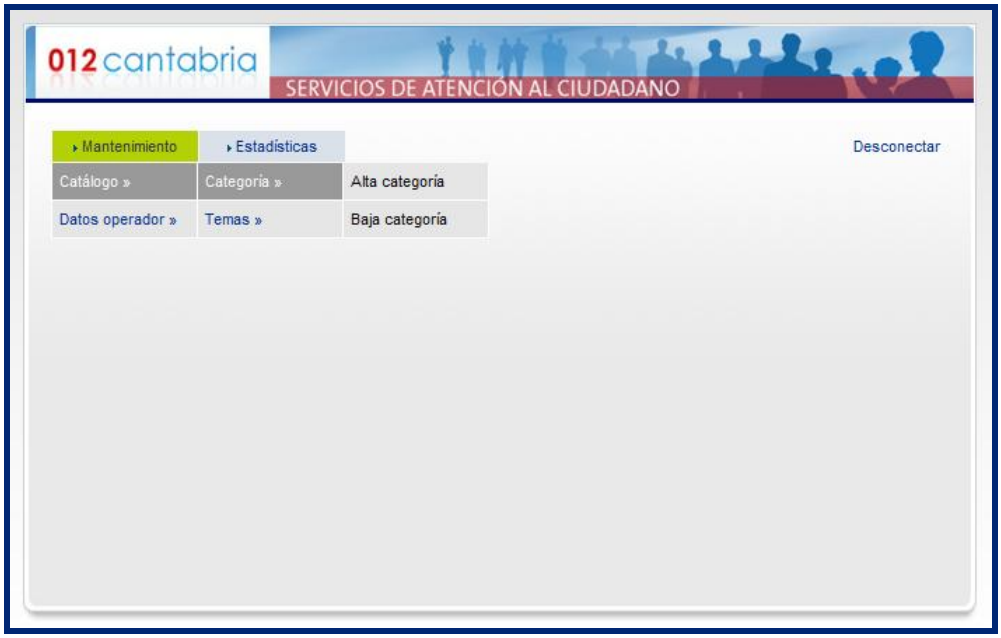


Figura 5.21 Página de Opciones de mantenimiento

Alta de categoría.

Al seleccionar “Alta categoría”, se muestra al usuario el formulario de alta correspondiente. Se trata de un formulario sencillo, como se puede observar en la *Figura 5.22*, en el que el administrador únicamente debe introducir el nombre de la categoría que desea crear, y añadirla al sistema.

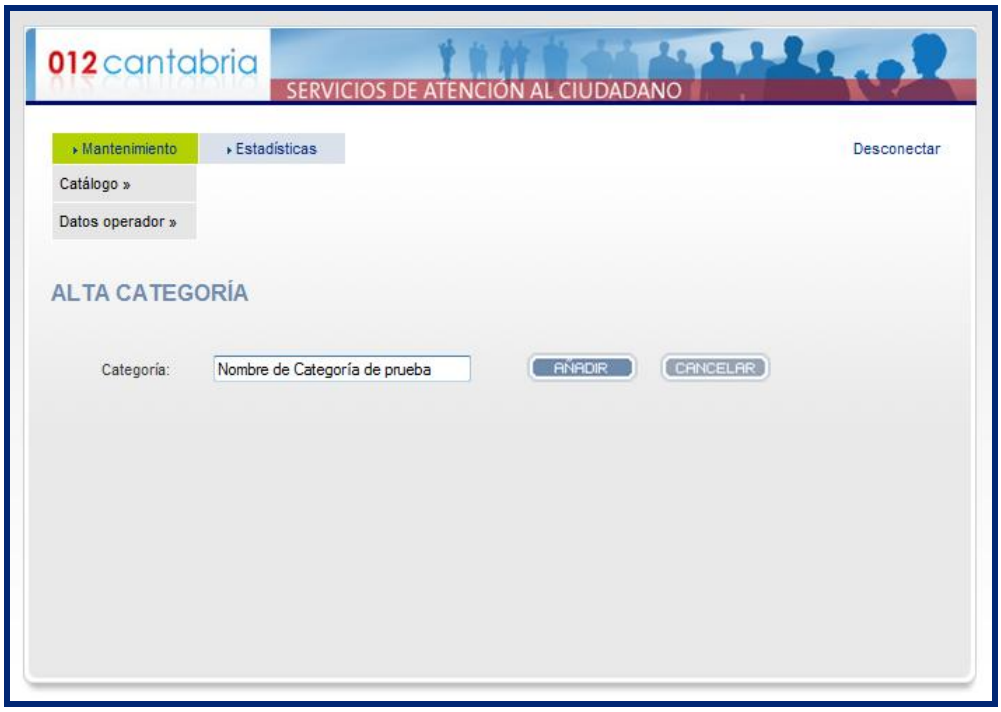


Figura 5.22 Página de Alta de categoría

En todos los procesos de alta se comprueba que se ha introducido el nombre del elemento que se desea dar de alta. Si el administrador no ha introducido un nombre de Categoría y selecciona el botón “AÑADIR”, se le muestra el mensaje que aparece en la *Figura 5.23*.

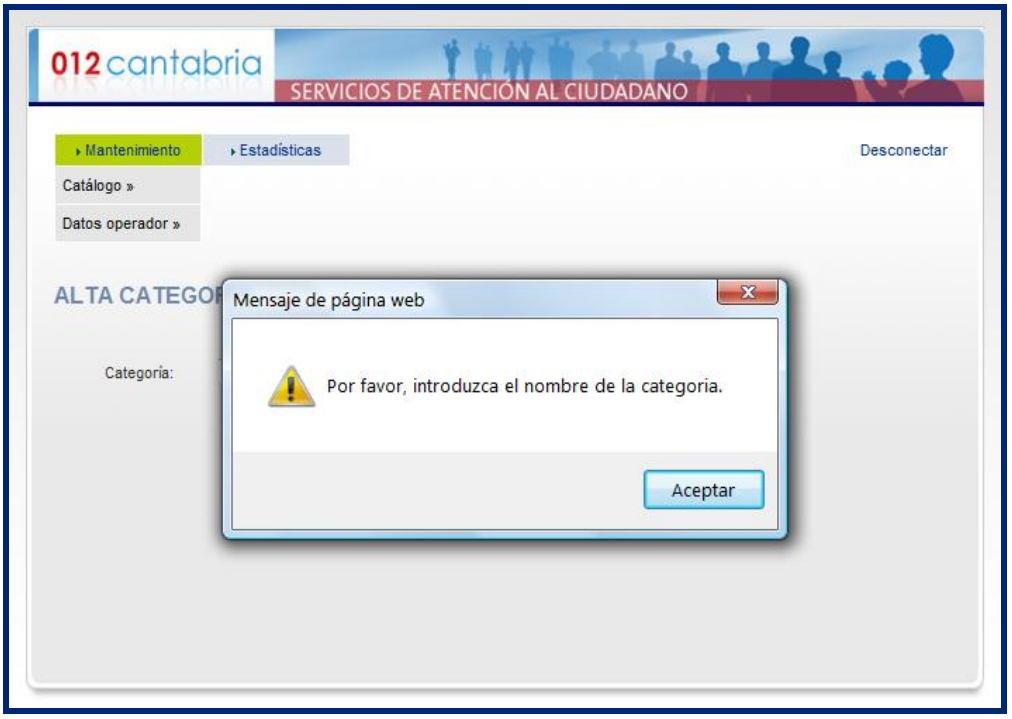


Figura 5.23 Mensaje de advertencia: Categoría no especificada

Si el nombre de la categoría que se desea añadir ya existe, se informa al usuario mediante el mensaje mostrado en la *Figura 5.24*.

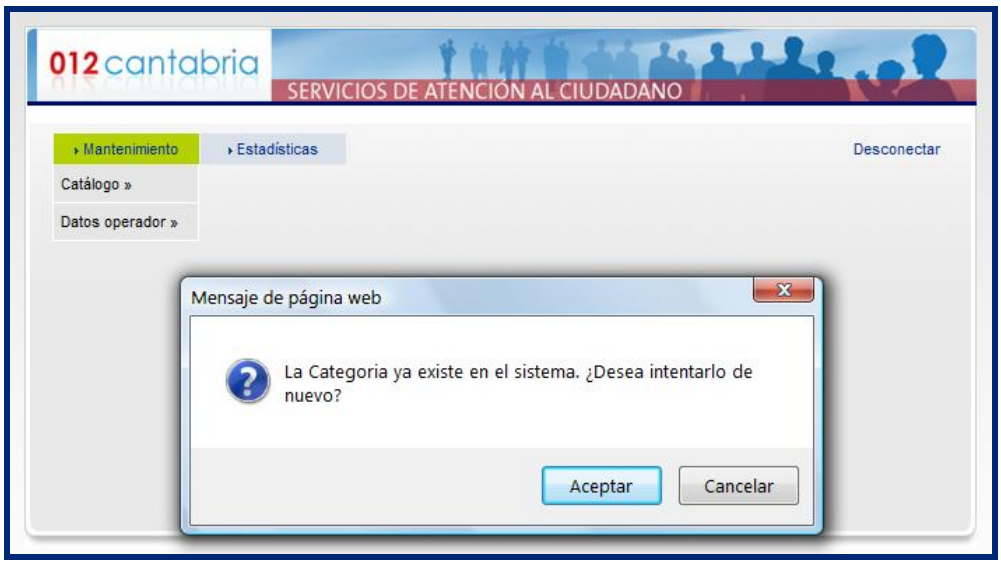


Figura 5.24 Mensaje interrogatorio: Categoría existente

En caso de que el administrador quede satisfecho al conocer que la categoría ya existe, y no desee crear otra diferente, cancelará la operación, volviendo a la página de Opciones de administración.

Si el usuario acepta intentarlo de nuevo, accede a la página de Alta de categoría, en la que puede modificar el texto previamente introducido.

Una vez añadida la categoría al sistema, se informa al administrador del éxito de la operación, mediante el mensaje de la imagen mostrada en la *Figura 5.25*.

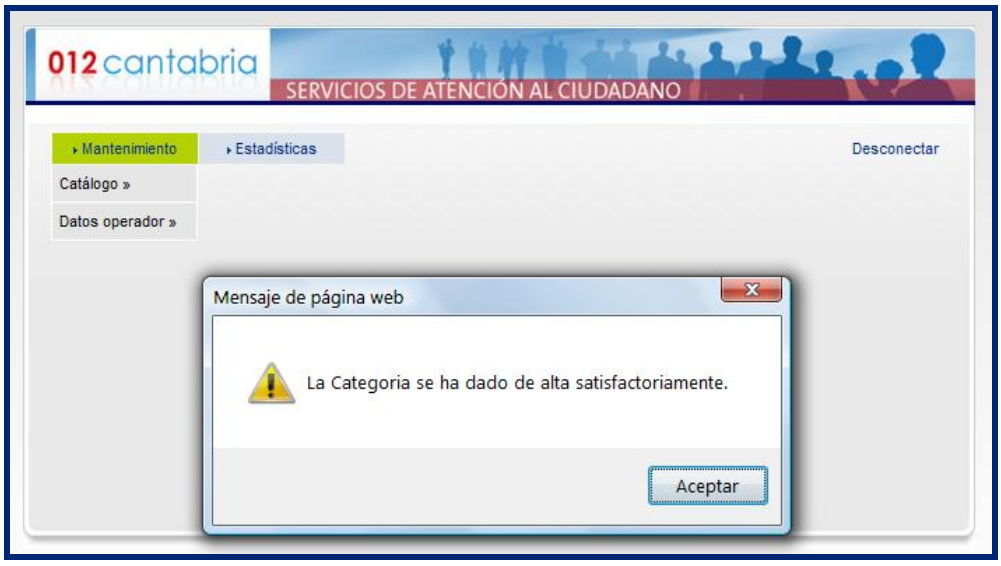


Figura 5.25 Mensaje informativo: Categoría añadida

Baja de categoría.

Cuando el administrador desea eliminar una categoría del sistema, debe seleccionar la opción de mantenimiento "Baja categoría", como se muestra en la *Figura 5.26*.

Figura 5.26 Página de Opciones de Mantenimiento

Tras esta selección, se ofrece al administrador la página de Baja de categoría, en la que se muestran las categorías pertenecientes al sistema en un menú desplegable, como se muestra en la *Figura 5.27*, de modo que pueda seleccionar aquella que desea eliminar del sistema.

Figura 5.27 Página de Baja de categoría

El administrador, tras seleccionar la categoría que quiere dar de baja y eliminarla, debe confirmar dicha acción, aceptando el mensaje mostrado en la *Figura 5.28*.

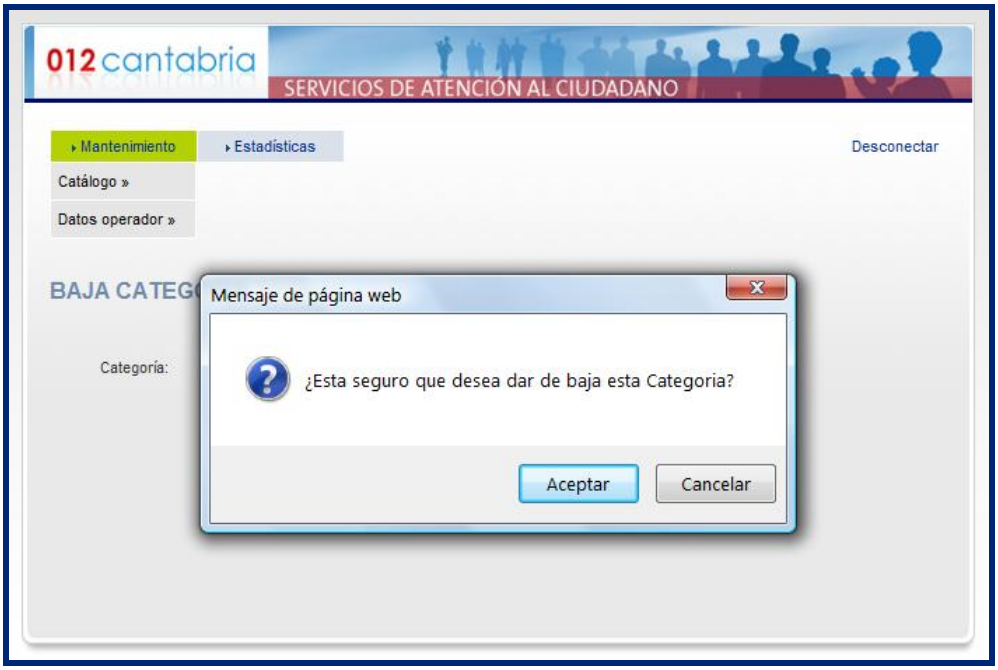


Figura 5.28 Mensaje interrogatorio: Confirmación de baja de categoría

Cuando se confirma la operación de baja, el usuario es informado de que la acción se ha llevado a cabo satisfactoriamente, como se muestra en la *Figura 5.29*.

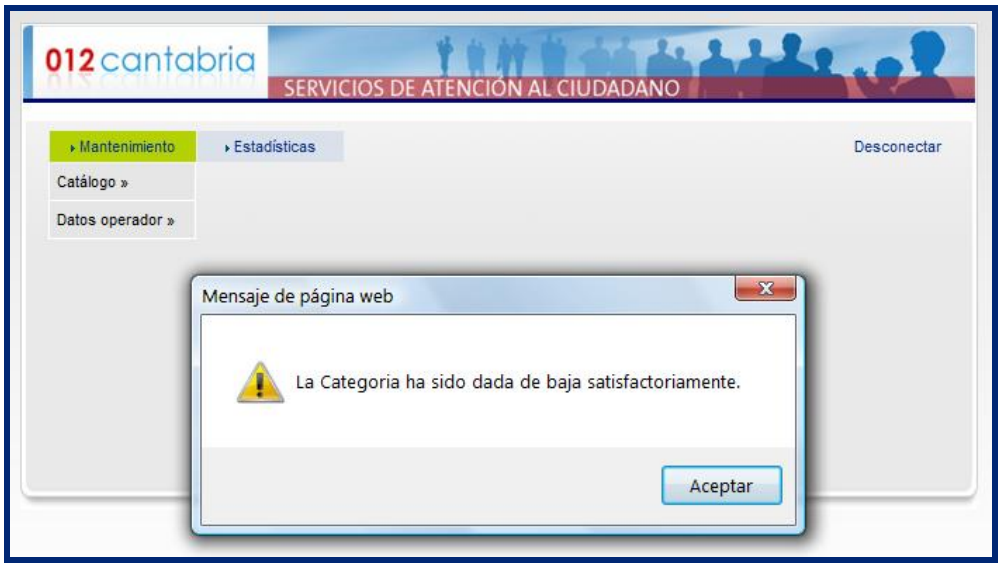


Figura 5.29 Mensaje informativo: Categoría eliminada

Alta de tema.

En cuanto a los temas, el administrador tiene la capacidad de añadirlos eliminarlos del sistema realizando la selección en el Catálogo, como en el caso de las categorías.

A continuación se muestra el proceso de alta de un tema, comenzando por la selección de la operación, que se ha capturado en la imagen de la *Figura 5.30*.

Figura 5.30 Página de "Opciones de Mantenimiento"

Al pulsar sobre "Alta temas", aparece la página en la que el administrador puede añadir un tema al catálogo existente.

Como se muestra en la *Figura 5.31*, en la página de "Alta de tema" se presenta un formulario que el administrador debe completar.

Se debe indicar el nombre del tema a añadir, la URL que contiene la información relacionada con el tema en cuestión, y por último la categoría a la que se quiere asociar este tema, escogiendo en el menú desplegable entre todas las categorías del sistema.

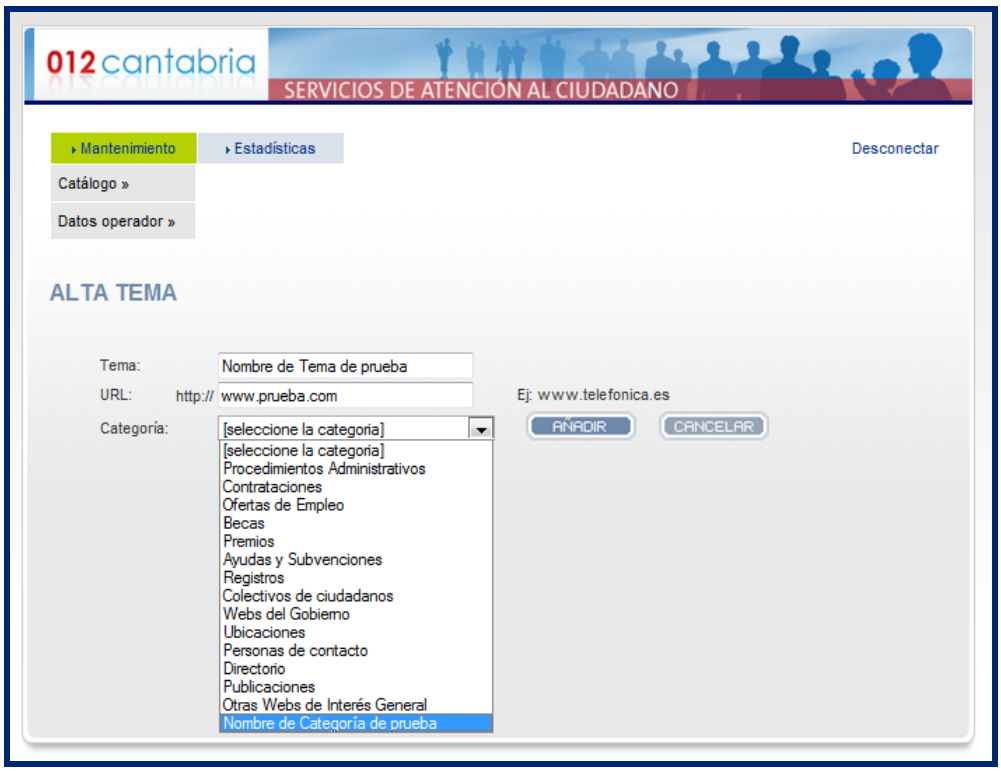


Figura 5.31 Página de "Alta de tema"

En este proceso de alta, además de comprobar que se ha introducido el nombre del tema, se constata que se han especificado también los campos URL y Categoría, indicándoselo al usuario en caso contrario para que proceda a completar la operación.

Si el nombre del tema ya pertenece al grupo de temas de la herramienta, se informa de ello al administrador. No se permite crear dos temas con el mismo nombre, aunque pertenezcan a categorías diferentes, ya que puede dar lugar a confusiones a la hora de consultar un operador la información. Por ello, se muestra el mensaje de la *Figura 5.32* si el nombre del tema escogido forma parte de los temas de la aplicación.

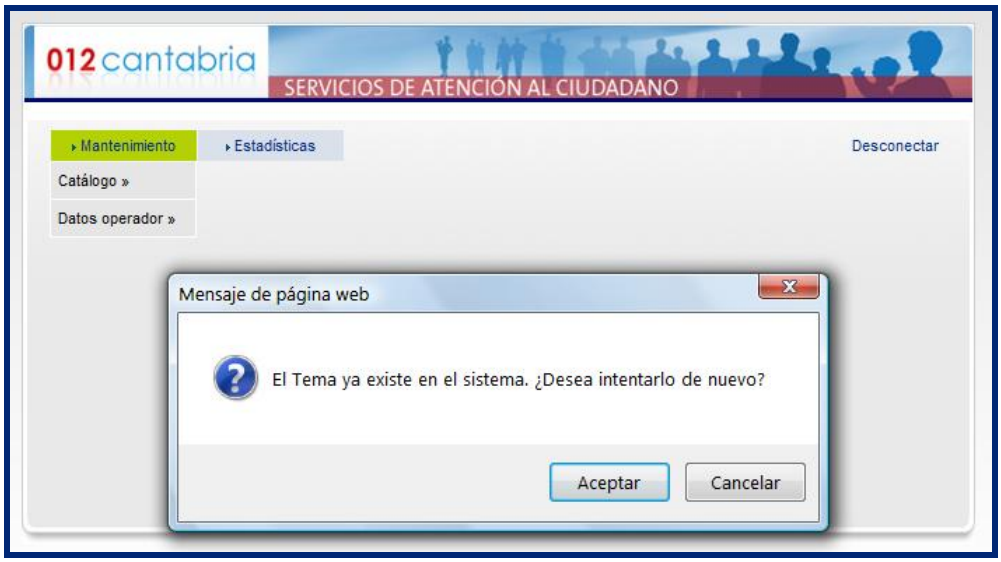


Figura 5.32 Mensaje interrogatorio: Tema existente

El usuario, al ser informado de que el tema que se disponía a añadir ya pertenece al sistema, puede no querer añadir un tema con diferente nombre, y cancelar la operación. En este caso, la aplicación vuelve a presentar la página de “Opciones de administración”.

Si por el contrario desea escoger un nuevo nombre para el tema, aceptará la opción de intentarlo de nuevo, y se le mostrará la página de “Alta de tema”, con el nombre de tema escogido previamente, para que pueda modificarlo.

Al añadir correctamente un tema al sistema, se indica al usuario que se ha llevado a cabo satisfactoriamente el alta del tema, como se muestra en la *Figura 5.33*.

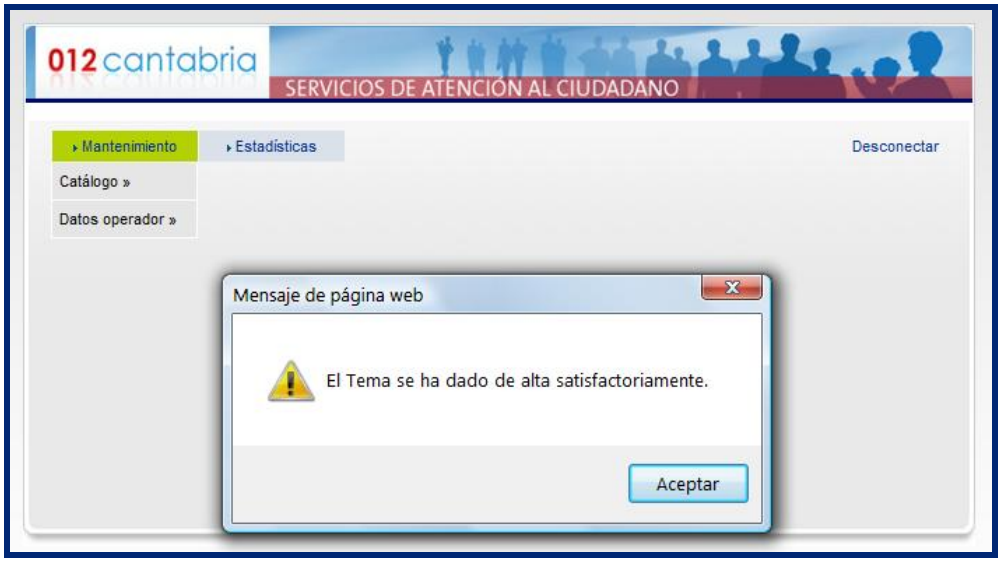


Figura 5.33 Mensaje informativo: Tema añadido

Baja de tema.

Para dar de baja un tema, el administrador realiza la selección que se muestra en la *Figura 5.34*.

Figura 5.34 Página de "Opciones de Mantenimiento"

A continuación, se presenta la página de "Baja de tema", en la que aparecen dos menús desplegables, el primero con las categorías del sistema, y el segundo, que muestra los temas asociados a la categoría seleccionada en el primer menú.

El administrador debe escoger en primer lugar la categoría a la que pertenece el tema que desea eliminar, y una vez realizada esta selección, elegirá el tema en el segundo menú.

En la *Figura 5.35* se muestra el ejemplo de la eliminación del tema creado en el apartado anterior, asociado a su vez a la Categoría de prueba también creada en esta sección. Al seleccionar dicha categoría en el primer menú desplegable, aparece el tema que se desea eliminar en el segundo.

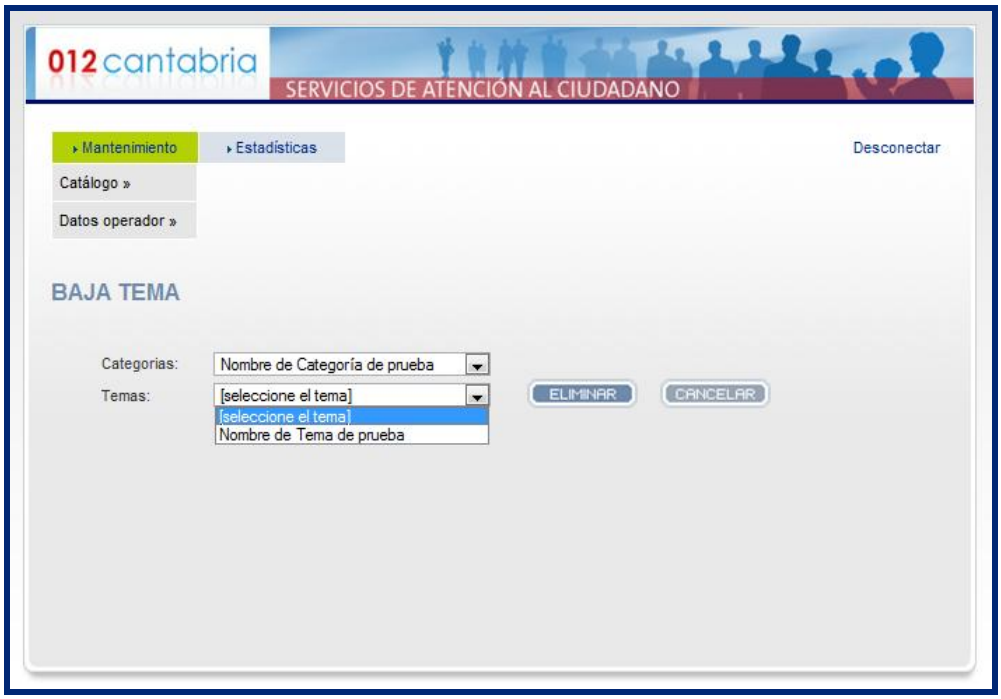


Figura 5.35 Página de "Baja de tema"

Una vez escogido el tema, al pulsar la tecla “ELIMINAR”, se solicita al administrador la confirmación de la baja del tema.

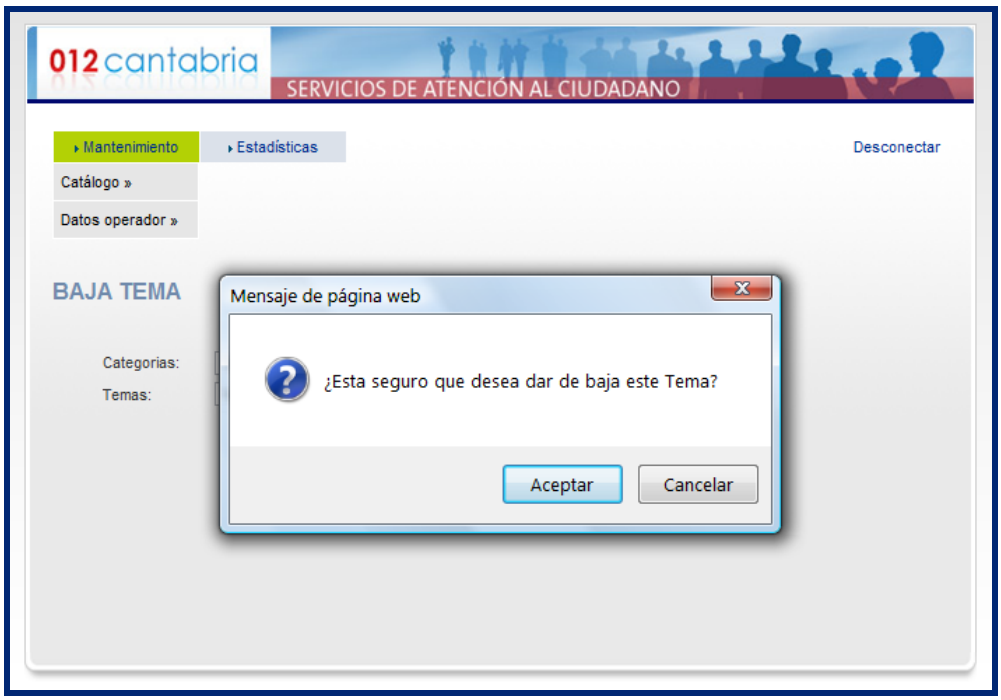


Figura 5.36 Mensaje interrogatorio: Confirmación de baja de tema

Al confirmar el usuario la eliminación del tema, se le informa de que se ha realizado correctamente la operación de baja.

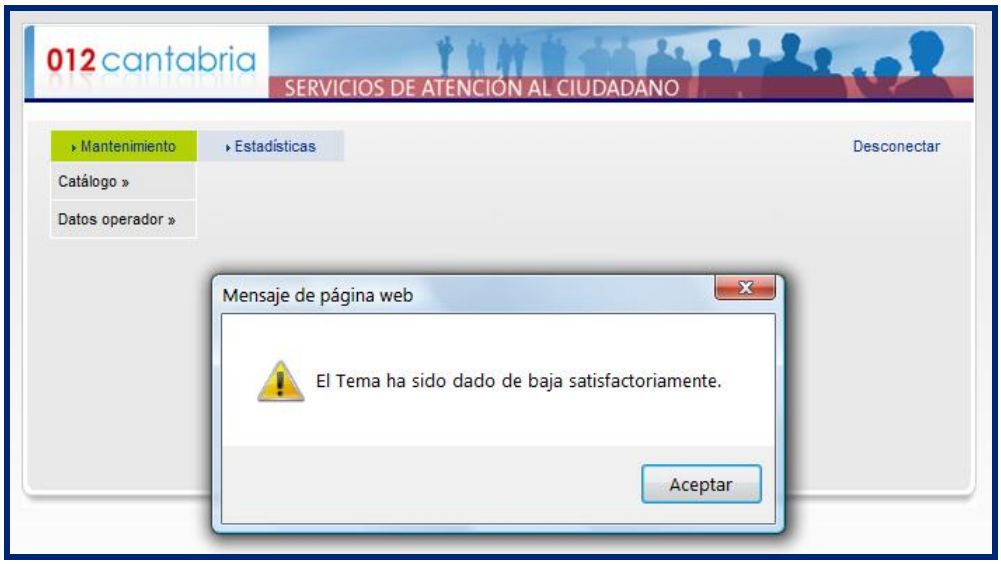


Figura 5.37 Mensaje informativo: Tema eliminado

Alta de operador.

El administrador también se encarga de la gestión de los operadores del sistema, lo que conlleva el alta y baja de usuarios, y la asignación de puestos.

Para realizar el alta de un nuevo operador, se selecciona la opción expuesta en la *Figura 5.38*.

Figura 5.38 Página de "Opciones de Mantenimiento"

Dicha selección redirige a la página de "Alta de operador", en la que se presenta al administrador el formulario de la *Figura 5.39*.

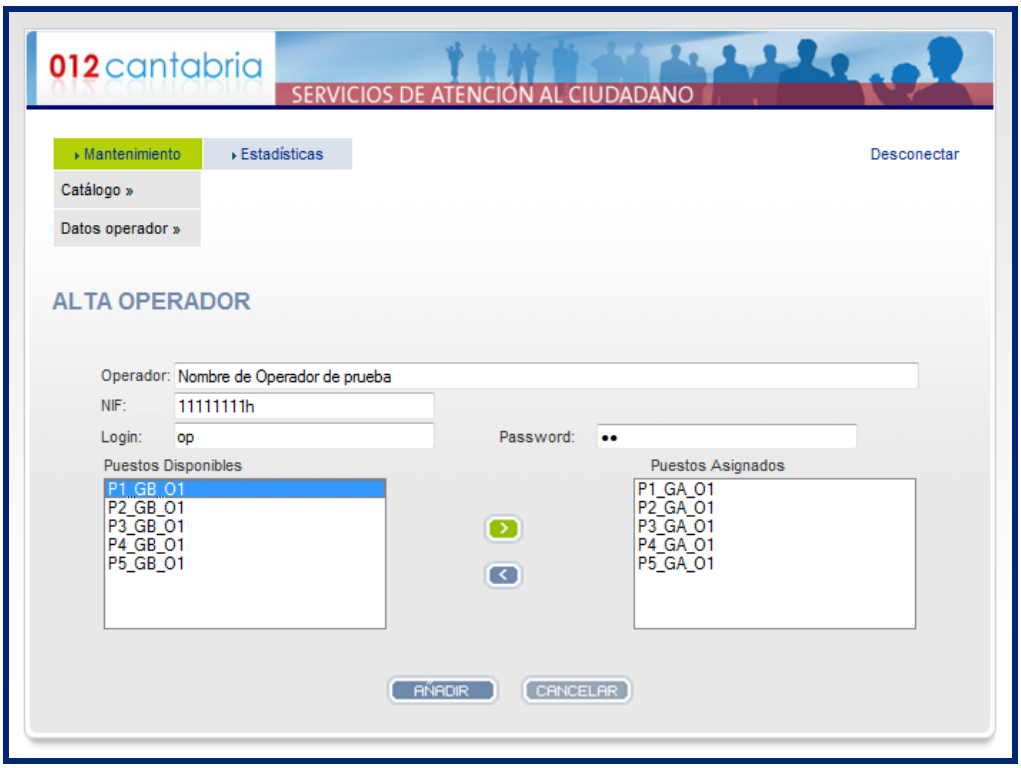


Figura 5.39 Página de "Alta de operador"

Para llevar a cabo el alta de un operador nuevo en el sistema, es necesario especificar varios parámetros.

En cuanto a la información sobre el usuario de tipo operador, se indica su nombre, su NIF y sus datos de autenticación en el sistema, usuario y contraseña.

Además, se realiza la asignación de puestos del sistema al nuevo usuario, escogiendo de la lista de Puestos Disponibles aquellos que desee asignar, y observando cómo al seleccionar el botón con la flecha hacia la derecha, pasan a la lista de Puestos Asignados.

Es posible también cancelar la asignación de un puesto, seleccionándolo de la lista de Puestos Asignados y pulsando el botón con la flecha hacia la izquierda. De esta manera, el puesto vuelve a la lista de Puestos Disponibles.

A la hora de dar de alta al operador seleccionando la opción “AÑADIR”, la aplicación realiza varias comprobaciones.

En primer lugar se confirma que se han especificado todos los campos, indicando al usuario que debe hacerlo en caso de que no se hayan introducido. Por ejemplo, se chequea que la lista de puestos asignados no está vacía. Si no se ha asignado ningún puesto al operador, se muestra el mensaje de la *Figura 5.40*.

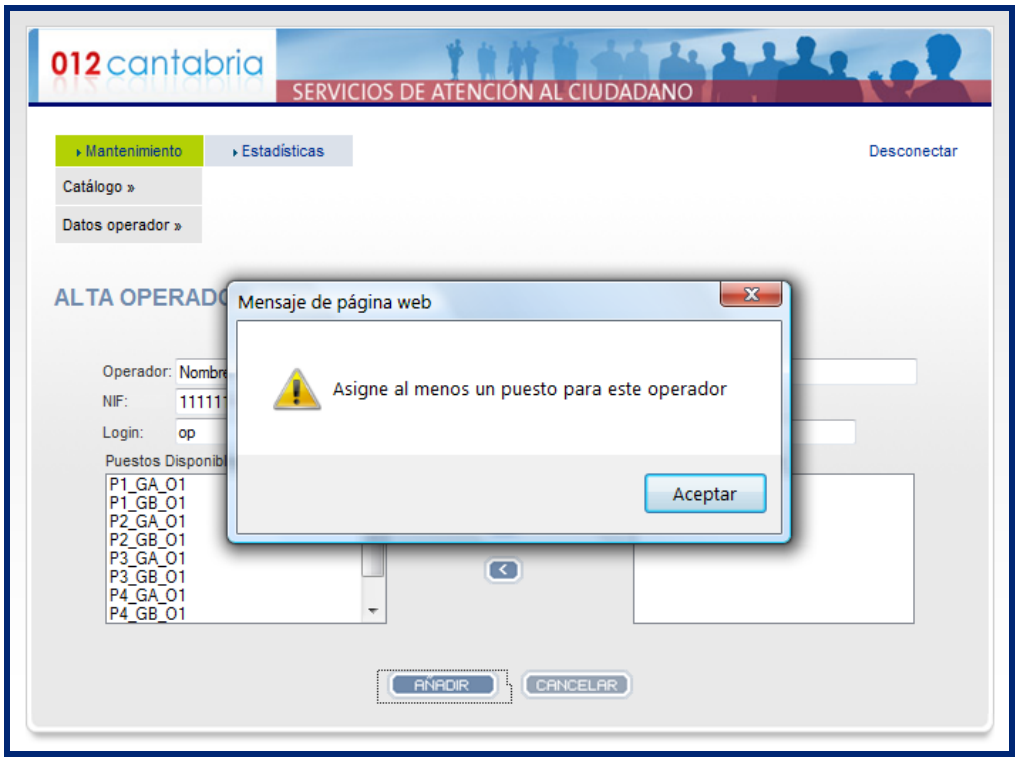


Figura 5.40 Mensaje de advertencia: Puesto no asignado

También es necesario constatar que el formato del NIF es correcto. En caso contrario, se muestra el correspondiente mensaje al administrador.

Una vez se ha comprobado que los campos introducidos cumplen con los requisitos de la aplicación, es posible a dar de alta al operador especificado, informando al administrador sobre el éxito de la acción.

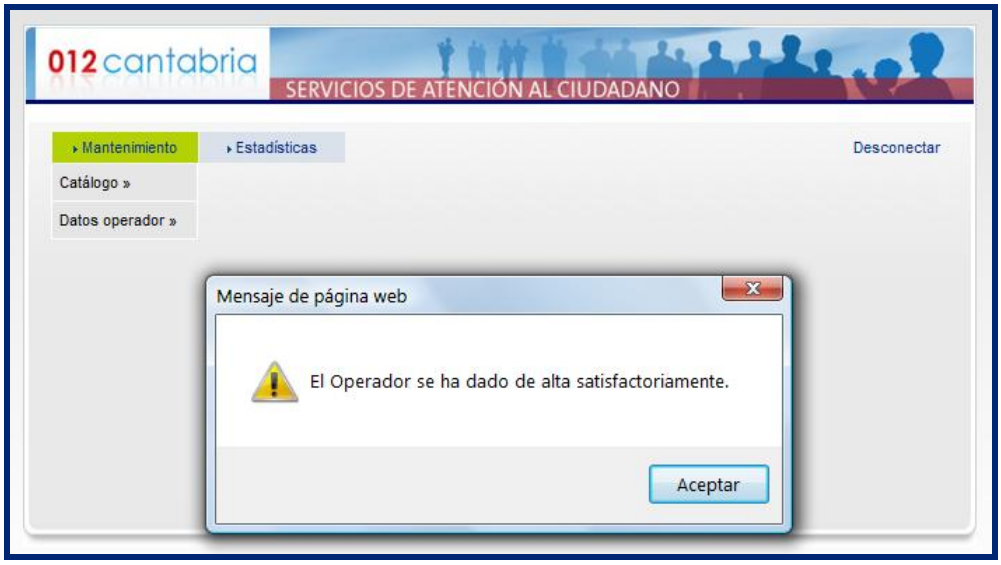


Figura 5.41 Mensaje informativo: Operador añadido

Baja de operador.

Si el administrador desea eliminar un usuario del sistema, debe seleccionar la opción de mantenimiento "Baja Operador". En este caso, se muestra la pantalla de "Baja de operador", que se puede observar en la *Figura 5.42*.

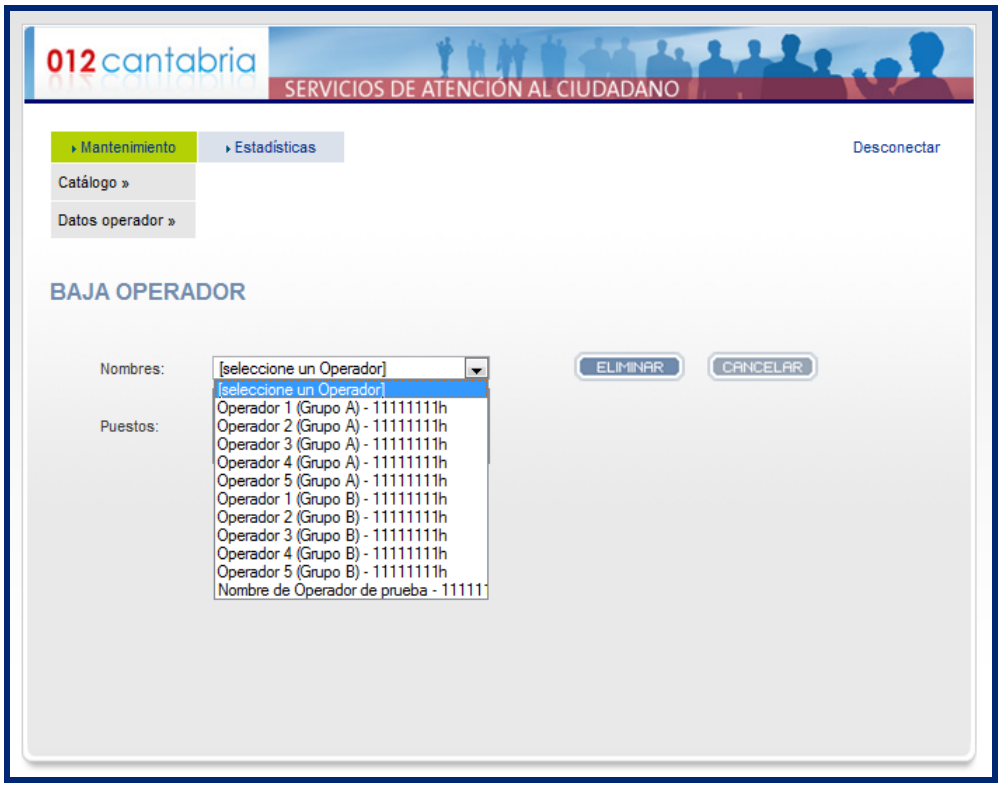


Figura 5.42 Página de "Baja de operador"

Como se puede observar, aparece un listado de los nombres de los operadores del sistema, junto con el NIF que corresponde a cada uno de ellos. Cuando el administrador selecciona el usuario que desea eliminar del sistema, en la siguiente sección del formulario, se indican a título informativo los puestos asociados a dicho usuario, como se muestra en la *Figura 5.43*.

Figura 5.43 Página de "Baja de operador"

Cuando el administrador selecciona la opción "ELIMINAR", debe confirmar la operación. Tras aceptar esta confirmación, el operador escogido se da de baja del sistema, y se informa de que la operación se ha llevado a cabo satisfactoriamente, mediante el mensaje de la *Figura 5.44*.

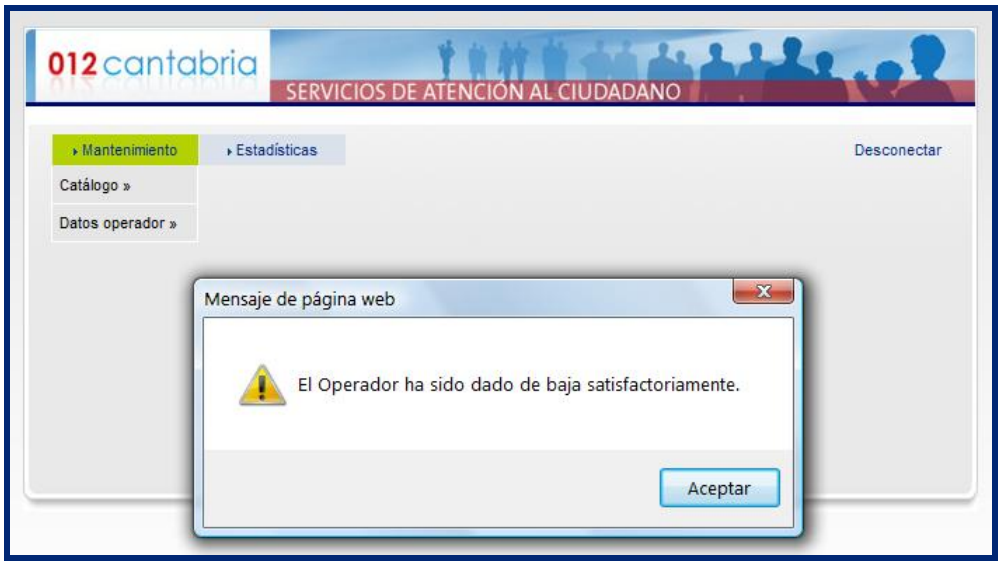


Figura 5.44 Mensaje informativo: Operador eliminado

❖ Estadísticas.

Una vez revisadas las tareas de mantenimiento, se aborda ahora el acceso a las estadísticas del sistema, la segunda opción de administración que se proporciona a los usuarios con perfil de administrador de la herramienta.

Al seleccionar esta opción, se muestra el listado de los diferentes cálculos estadísticos que proporciona el sistema, como se puede observar en la *Figura 5.45*.

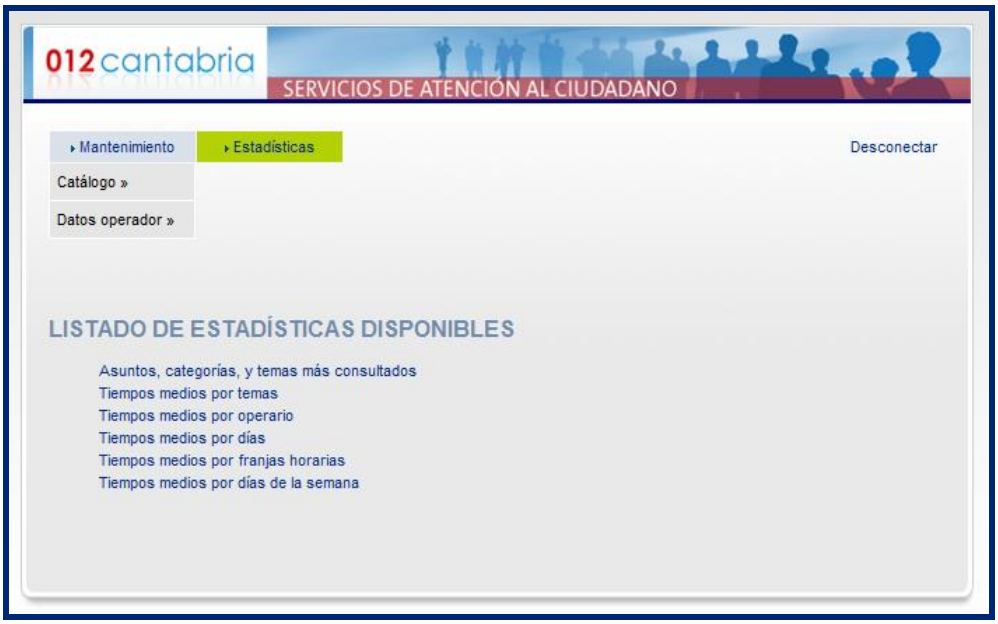


Figura 5.45 Página de Estadísticas

Al seleccionar cada uno de los tipos de estadística del listado, se ofrece al administrador un formulario en el que debe especificar el rango temporal que se quiere tener en cuenta y el o los puestos cuya actividad se desea monitorizar a la hora de realizar el cálculo.

Por ejemplo, en el caso de la primera estadística, Asuntos, categorías, y temas más consultados, el formulario se muestra en la *Figura 5.46*.

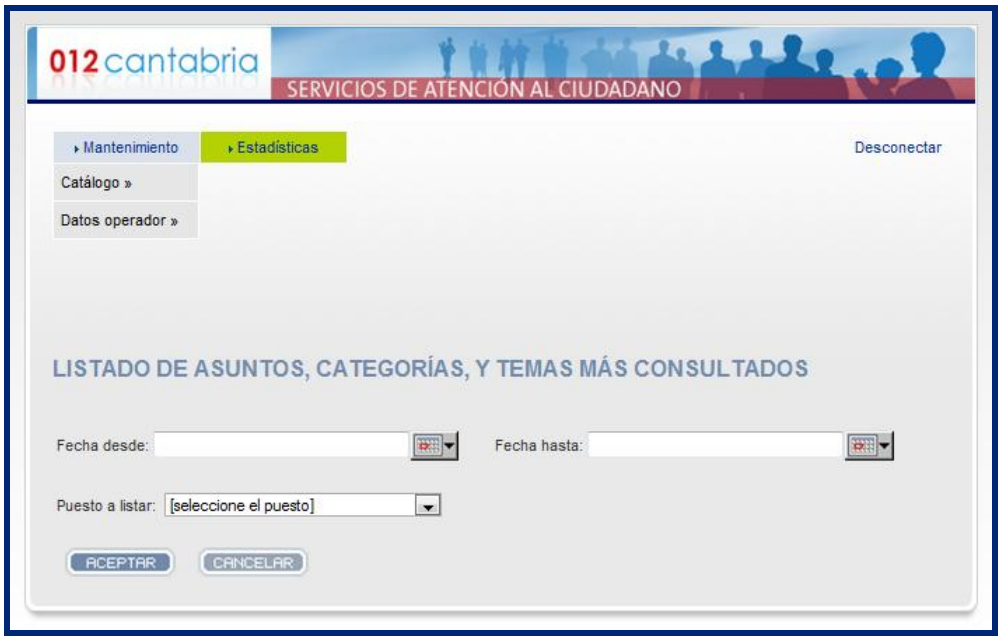


Figura 5.46 Formulario de cálculo estadístico

Una vez rellenado el formulario, al pulsar el botón “ACEPTAR”, se solicita el cálculo con los parámetros indicados.

A continuación se muestra un ejemplo de cálculo de cada una de las estadísticas.

Asuntos, categorías, y temas más consultados

En el ejemplo mostrado a continuación se ha realizado el cálculo para todos los puestos de la oficina, obteniendo el resultado de la *Figura 5.47*.

Al haber sido seleccionado todos los puestos, no se indica un lugar de puesto dentro de la organización, ya que en este caso son todos. Por tanto, este es el valor de los campos “Puesto”, “Grupo”, “Oficina” y “Entidad”. El valor del campo “Usuario” corresponde al nombre de usuario del administrador que está solicitando el cálculo de la estadística.

Se muestra una tabla con los dos tipos de asunto, “Consulta de información” y “Sugerencias”, y el número de consultas que se han realizado en relación con cada asunto, en el periodo indicado.

Figura 5.47 Resultado Estadística 1: Asuntos

Pinchando sobre el enlace “Consulta de Información”, se accede a las Categorías que han sido consultadas en el periodo seleccionado en todos los puestos de operadores, obteniendo la información presentada en la *Figura 5.48*.

Se presenta un listado de las categorías consultadas, y el número de consultas de cada una de ellas.

Figura 5.48 Resultado Estadística 1: Categorías

Seleccionando los enlaces de las categorías es posible obtener información sobre los temas asociados a cada una de ellas que han sido consultados. En el ejemplo, se selecciona la categoría “Ofertas de Empleo” y los temas asociados se muestran en la *Figura 5.49*, indicando también el número de consultas llevadas a cabo en relación con cada tema de la lista.

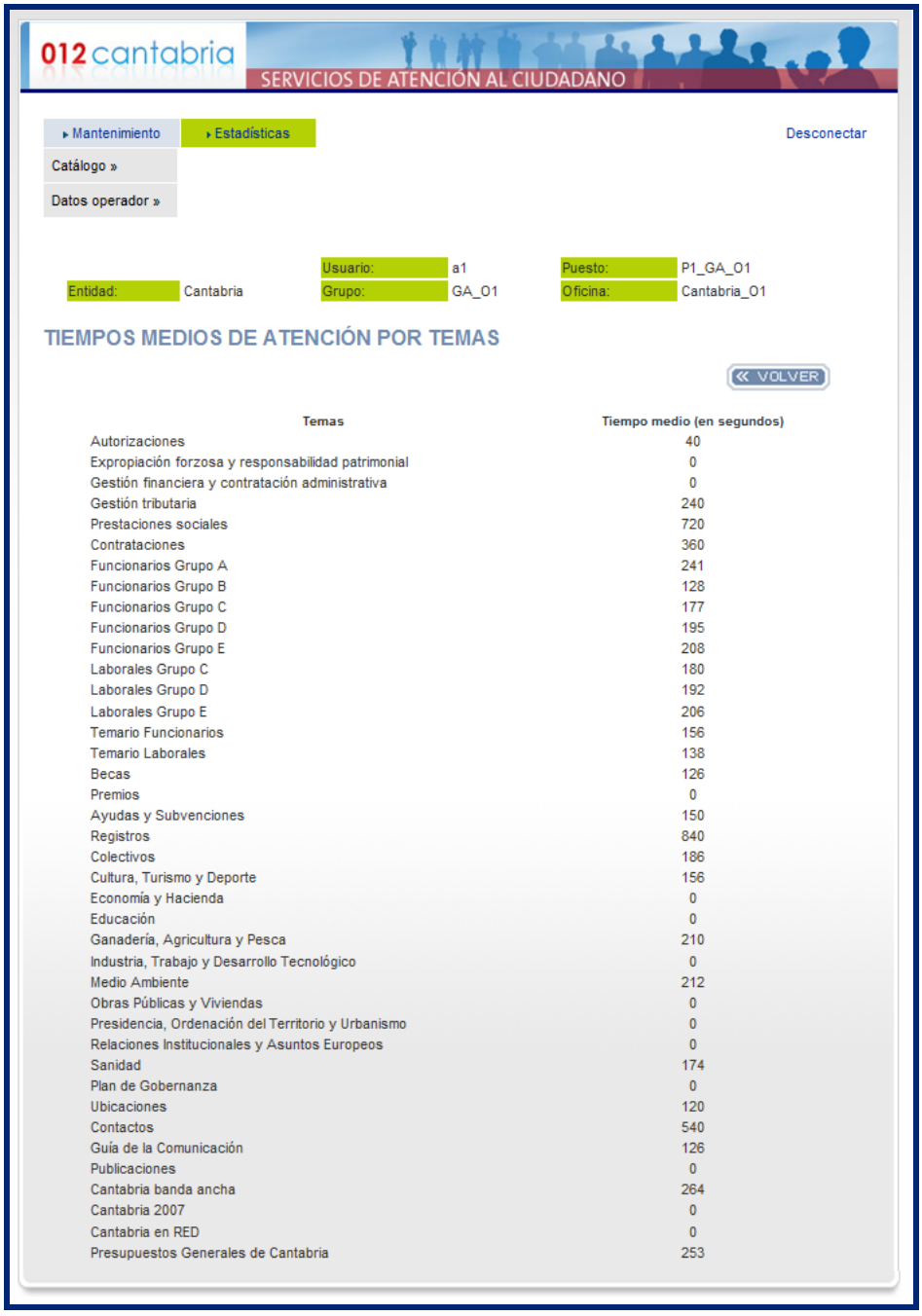
Figura 5.49 Resultado Estadística 1: Temas

Tiempos medios por temas

Para la ilustración de esta estadística se ha llevado a cabo el cálculo para el puesto 1, especificándolo en el formulario, además del rango temporal para el cálculo.

Se puede observar el resultado del cálculo realizado en la *Figura 5.50*, donde se indica en primer lugar el nombre del usuario, y a continuación la posición del puesto en la organización, perteneciendo en este caso el puesto 1 al Grupo A, que está a su vez en la Oficina 1.

En este caso se presenta el listado de todos los temas del sistema, y el tiempo medio dedicado a cada uno de estos temas en el rango de fechas previamente seleccionado.



Temas	Tiempo medio (en segundos)
Autorizaciones	40
Expropiación forzosa y responsabilidad patrimonial	0
Gestión financiera y contratación administrativa	0
Gestión tributaria	240
Prestaciones sociales	720
Contrataciones	360
Funcionarios Grupo A	241
Funcionarios Grupo B	128
Funcionarios Grupo C	177
Funcionarios Grupo D	195
Funcionarios Grupo E	208
Laborales Grupo C	180
Laborales Grupo D	192
Laborales Grupo E	206
Temario Funcionarios	156
Temario Laborales	138
Becas	126
Premios	0
Ayudas y Subvenciones	150
Registros	840
Colectivos	186
Cultura, Turismo y Deporte	156
Economía y Hacienda	0
Educación	0
Ganadería, Agricultura y Pesca	210
Industria, Trabajo y Desarrollo Tecnológico	0
Medio Ambiente	212
Obras Públicas y Viviendas	0
Presidencia, Ordenación del Territorio y Urbanismo	0
Relaciones Institucionales y Asuntos Europeos	0
Sanidad	174
Plan de Gobernanza	0
Ubicaciones	120
Contactos	540
Guía de la Comunicación	126
Publicaciones	0
Cantabria banda ancha	264
Cantabria 2007	0
Cantabria en RED	0
Presupuestos Generales de Cantabria	253

Figura 5.50 Resultado Estadística 2

Tiempos medios por operario

En este ejemplo se realiza el cálculo teniendo en cuenta todos los puestos de la oficina, y se obtiene el resultado mostrado en la *Figura 5.51*, donde se muestra el listado de operadores del sistema, y el tiempo medio que dedica cada uno de ellos a realizar una consulta.

Figura 5.51 Resultado Estadística 3

Tiempos medios por días

A continuación se muestra el ejemplo del cálculo de la estadística “Tiempo medio por días”, teniendo en cuenta todos los puestos del sistema, e indicando un rango temporal del 15-10-2011 al 31-10-2011.

Como resultado, se obtiene el siguiente listado de fechas, que abarca los días comprendidos entre las fechas establecidas para el inicio y fin del cálculo, ambas incluidas, y para cada día se indica el tiempo medio dedicado a realizar consultas durante ese día, en este caso en todos los puestos de la organización.

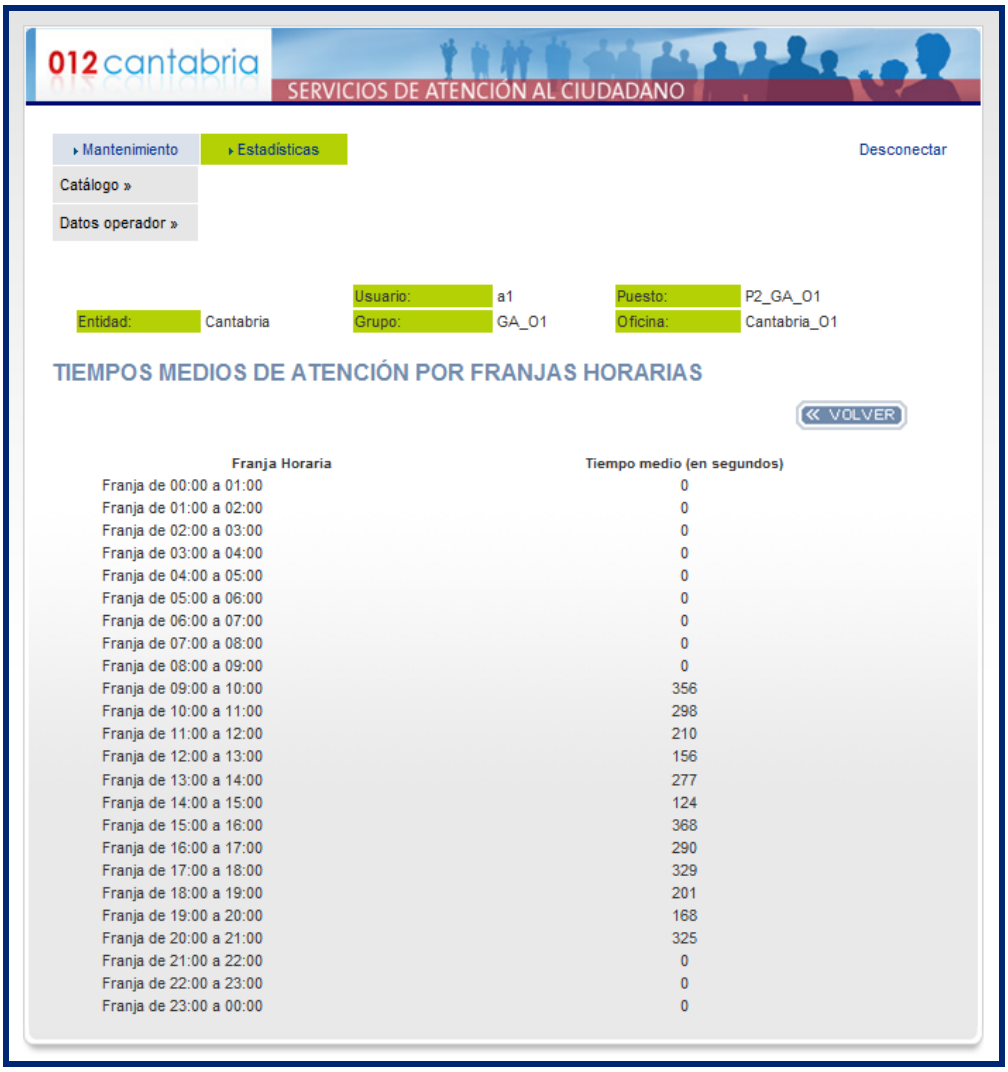
Figura 5.52 Resultado Estadística 4

Tiempos medios por franjas horarias

Para el siguiente ejemplo se ha establecido el cálculo teniendo en cuenta el puesto 2. Por tanto, como se puede observar en la *Figura 5.53*, se indica en el nombre del usuario administrador que realiza la operación, y en el resto de los campos la situación del puesto dentro de la organización.

Se indica que se ha realizado el cálculo teniendo en cuenta el puesto 2 del Grupo A, que pertenece a la Oficina 1.

Se puede observar un listado de horas, donde se observa que hay actividad en el horario de atención al cliente, que es de 9 a 21h de lunes a viernes, y sábados de 9 a 14h, y se especifica la media de tiempo dedicada a realizar consultas durante cada periodo horario.



Franja Horaria	Tiempo medio (en segundos)
Franja de 00:00 a 01:00	0
Franja de 01:00 a 02:00	0
Franja de 02:00 a 03:00	0
Franja de 03:00 a 04:00	0
Franja de 04:00 a 05:00	0
Franja de 05:00 a 06:00	0
Franja de 06:00 a 07:00	0
Franja de 07:00 a 08:00	0
Franja de 08:00 a 09:00	0
Franja de 09:00 a 10:00	356
Franja de 10:00 a 11:00	298
Franja de 11:00 a 12:00	210
Franja de 12:00 a 13:00	156
Franja de 13:00 a 14:00	277
Franja de 14:00 a 15:00	124
Franja de 15:00 a 16:00	368
Franja de 16:00 a 17:00	290
Franja de 17:00 a 18:00	329
Franja de 18:00 a 19:00	201
Franja de 19:00 a 20:00	168
Franja de 20:00 a 21:00	325
Franja de 21:00 a 22:00	0
Franja de 22:00 a 23:00	0
Franja de 23:00 a 00:00	0

Figura 5.53 Resultado Estadística 5

Tiempos medios por días de la semana

En este caso se ha solicitado el cálculo relativo al puesto 3 del sistema. Dicho puesto se indica en la página en la que se muestra el resultado obtenido, especificándose que se ha llevado a cabo el cálculo para el puesto 2, que forma parte del Grupo A, y éste de la Oficina 1.

Además de la situación del puesto, como se puede observar en la *Figura 5.54*, se presenta el listado de días de la semana, detallando el tiempo medio dedicado a atender una consulta en ese puesto en cada uno de los días.

Figura 5.54 Resultado Estadística 6

5.3. USUARIO DE TIPO ADMINISTRADOR - OPERADOR

❖ Autenticación.

Al usuario con perfil de administrador-operador del sistema se le presenta la página de autenticación de la aplicación, en la que introduce sus credenciales, al igual que los otros tipos de usuarios, como se muestra en el siguiente ejemplo capturado en la figura.

Figura 5.55 Página de Autenticación

❖ Selección de puesto.

Al tratarse de un perfil con funciones de operador, a este usuario se le ofrece la selección de puesto, en el caso de que tenga asociado más de un puesto, como es el caso del ejemplo actual, mostrado en la siguiente figura.

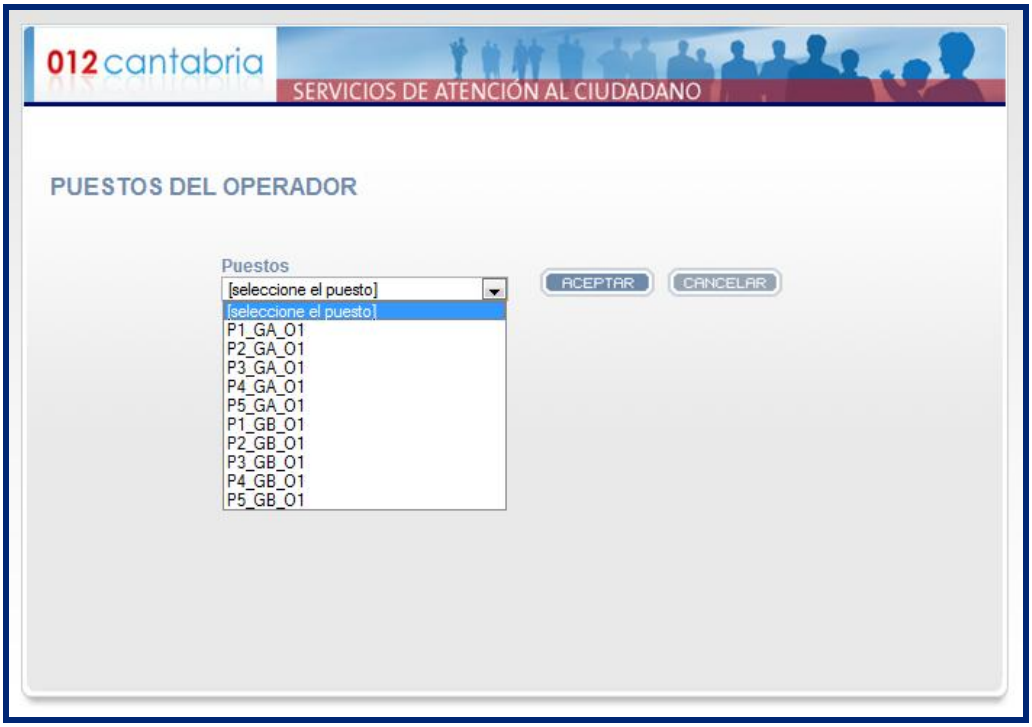


Figura 5.56 Página de "Selección de puesto"

Si el usuario únicamente tiene un puesto asociado, no se le ofrece esta pantalla de "Selección de puesto", y es su puesto el que se establece para la sesión.

En el caso en que debido a un error en el sistema el usuario no tuviera ningún puesto asignado, se le informaría de dicho error, del mismo modo que para el usuario con perfil de operador.

❖ Opciones de administración y registro.

Este tipo de usuario puede acceder a todas las operaciones llevadas a cabo por la herramienta desarrollada, por lo que tras la autenticación y asignación de puesto, se le presenta la siguiente pantalla, donde puede acceder a las opciones de administración, mantenimiento y estadísticas, y también al registro de un informe asociado a una llamada.

Figura 5.57 Página de Opciones de administración y registro

Al usuario con perfil de administrador-operador se le ofrecen en todo momento estas opciones. El acceso a cada una de las operaciones se realiza de la misma manera que lo hace un usuario de tipo administrador en el caso de las acciones de mantenimiento y acceso a las estadísticas, e igual que accede un usuario con perfil de operador al realizar un registro.

5.4. TODO TIPO DE USUARIOS

❖ Gestión de errores y desconexión

Se han comentado algunos casos de gestión de errores para cada tipo de usuario, como por ejemplo el caso en que no se introducen los datos en algún campo de un formulario. Este control de campos se realiza en todas las páginas, comprobando que todos los campos necesarios se completan por parte del usuario que esté utilizando la herramienta en cada caso.

También se ha mencionado el caso del usuario con perfil de operador, o bien de administrador-operador que no tiene ningún puesto asignado. Este tipo de error podría deberse a un problema con la base de datos, ya que como se ha detallado, a la hora de crear un nuevo usuario de tipo operador, el administrador debe asignar al menos un puesto al usuario que quiere añadir al sistema.

A continuación se detallan algunos ejemplos de gestión de errores comunes a los diferentes tipos de usuario.

En el caso de la página de Autenticación, si se introducen unas credenciales que no corresponden a un usuario registrado en el sistema, se muestra la siguiente pantalla de error. A continuación, se vuelve a mostrar de nuevo la página de autenticación, de manera que el usuario pueda intentarlo de nuevo.

Figura 5.58 Página de Error de Autenticación

En el caso de no registrarse actividad por parte del usuario de la aplicación durante un periodo de 30 minutos, se finaliza la sesión, mostrando el siguiente mensaje informativo al respecto.

La siguiente página que se muestra al usuario es la de Autenticación, ofreciéndole la posibilidad de iniciar una nueva sesión.

Figura 5.59 Página de Error: Sesión expirada

Si durante la utilización de la herramienta se produce un error en el acceso a la base de datos, se muestra al usuario el correspondiente mensaje de error, capturado en la siguiente imagen. Al igual que en el caso anterior, se finaliza la sesión del usuario y se ofrece la página de Autenticación.

Figura 5.60 Página de Error: Acceso a Base de Datos

El propio usuario también tiene la opción de provocar la desconexión, seleccionando el botón “Desconectar”, presente en las pantallas de la aplicación en las que el usuario tiene abierta una sesión. En este caso, directamente se ofrece la página de autenticación.

6. PRUEBAS DEL SISTEMA

Las pruebas que se han llevado a cabo sobre el sistema se pueden dividir en dos tipos.

En primer lugar están aquellas que se han realizado durante la fase del desarrollo del sistema. Se trata en este caso de pruebas unitarias, cuya finalidad consiste en comprobar el funcionamiento de cada módulo de código de manera independiente. De esta manera se garantiza el correcto funcionamiento de cada uno de estos módulos por separado.

Una vez finalizado el desarrollo del sistema, se ejecutan los casos de prueba definidos para cada funcionalidad del sistema.

Para la depuración del código se ha hecho uso de la herramienta log4j, para lo que se ha incluido la biblioteca log4j-1.2.14.jar en el sistema. Se trata de una biblioteca que permite al desarrollador crear un registro de mensajes llamados logs, determinando la salida y el nivel de prioridad de cada log.

Mediante el archivo "log4j.properties" se definen parámetros de utilidad como los siguientes:

- Salida de logs en la consola de ejecución (`org.apache.log4j.ConsoleAppender`).
- El "appender" o fichero de destino de estos mensajes (`C:/PFC/logs/CantabriaPFC.log`).
- El tipo de fichero de destino (`org.apache.log4j.RollingFileAppender`), que permite definir a su vez el número de archivos de backup (`MaxBackupIndex=4`), o el tamaño máximo de estos archivos (`MaxFileSize=10MB`).
- la prioridad de los mensajes, de tipo "`DEBUG`" en este caso (el tercero menos prioritario de los 8 niveles establecidos).

Para utilizar esta herramienta es necesario importar la clase `Logger`, crear el objeto de clase de tipo `Logger` indicando la clase actual (en el ejemplo `"servlets.servletLogin"`), e incluir las líneas de debug necesarias. A continuación se muestra un ejemplo de código de la aplicación empleando esta herramienta:

```
import org.apache.log4j.Logger;
public class ServletLogin extends HttpServlet{
    private Logger logger = Logger.getLogger("servlets.servletLogin");
    public void doPost(HttpServletRequest req, HttpServletResponse resp){
       logger.debug("INICIO");

       …
```

Para documentar los casos de prueba se han establecido los siguientes parámetros:

- ID: Número identificador asociado a la prueba.
- Nombre: Nombre que identifica la prueba a realizar.
- Descripción: Texto que describe los pasos a seguir para realizar la prueba.
- Comportamiento esperado: Texto que describe el comportamiento que se espera observar tras realizar las acciones indicadas en la descripción de la prueba.
- Resultado: Indicador de éxito o fallo en el resultado de la prueba.

A continuación se muestra la batería de pruebas realizada una vez desarrollada la aplicación, comprobando el funcionamiento global del sistema y teniendo en cuenta los diferentes casos de uso.

Se han estructurado las pruebas según se detalla en los siguientes apartados, y se han obtenido los resultados indicados en la columna derecha de las tablas.:

6.1. GESTIÓN DE FALLOS

ID	Nombre	Descripción	Comportamiento esperado	Resultado
GF-01	Campos vacíos	No introducir texto en aquellos campos que requieran ser rellenados para continuar la acción.	Se muestra un mensaje al usuario indicándole que debe introducir el campo en cuestión.	OK
GF-02	Credenciales incorrectas	Introducir Usuario incorrecto/ Introducir Password incorrecta / Introducir Usuario y Password incorrectos.	Página de error indicando: "El nombre de usuario o el password son incorrectos".	OK
GF-03	Cancelación de operación	En todas las páginas en las que existe el botón "CANCELAR", seleccionarlo.	Se cancela la acción, volviendo a una página anterior consecuente.	OK
GF-04	Sesión expirada	Tras la autenticación y alguna acción, esperar 30 minutos sin actividad y tras este tiempo, seleccionar alguna opción de la página actual de la aplicación.	Página de error indicando: "La sesión ha expirado".	OK

Tabla 6.1 Batería de pruebas – Gestión de fallos

6.2. AUTENTICACIÓN

ID	Nombre	Descripción	Comportamiento esperado	Resultado
A-01	Autenticación de operador	Introducir las credenciales de un operador registrado en el sistema.	Se autentica correctamente al usuario, mostrándole las opciones correspondientes al perfil de operador: Registro.	OK
A-02	Autenticación de administrador	Introducir las credenciales de un administrador registrado en el sistema.	Se autentica correctamente al usuario, mostrándole las opciones correspondientes al perfil de administrador: Mantenimiento y Estadísticas.	OK
A-03	Autenticación de operador-administrador	Introducir las credenciales de un operador registrado en el sistema.	Se autentica correctamente al usuario, mostrándole las opciones correspondientes al perfil de operador-administrador: Registro, Mantenimiento y Estadísticas.	OK

Tabla 6.2 Batería de pruebas – Autenticación

6.3. REGISTRO

ID	Nombre	Descripción	Comportamiento esperado	Resultado
R-01	Inicio temporizador	Seleccionar cualquier otro campo sin haber seleccionado el botón "INICIO LLAMADA".	Se muestra un mensaje de alerta al usuario indicando que el tiempo de llamada está vacío y debe iniciar una llamada.	OK
R-02	Intento de detención de llamada	Seleccionar el botón "INICIO LLAMADA" después de haber iniciado previamente una llamada.	Se muestra un mensaje de alerta al usuario indicando que no es posible detener el proceso y se debe completar la operación de registro.	OK

ID	Nombre	Descripción	Comportamiento esperado	Resultado
R-03	Campo NIF incorrecto	Introducir un formato de NIF incorrecto en el informe y seleccionar el botón “ACEPTAR”.	Se informa al usuario de que el formato del NIF es incorrecto.	OK
R-04	Campo email incorrecto	Introducir un formato de email incorrecto en el informe y seleccionar el botón “ACEPTAR”.	Se informa al usuario de que el formato del email es incorrecto.	OK
R-05	Campo Categoría inhabilitado	No seleccionar campo “Asunto” / Seleccionar “Asunto: Sugerencias”.	En ambos casos el campo “Categoría” está inhabilitado.	OK
R-06	Campo Categoría habilitado	Seleccionar “Asunto: Consulta de Información”.	El campo “Categoría” está habilitado, mostrando una lista desplegable con las categorías del sistema.	OK
R-07	Campo Tema inhabilitado	1. Seleccionar “Asunto: Consulta de Información”. 2. No seleccionar una “Categoría” / Seleccionar “Categoría” y decidir no asignarle ningún tema.	En ambos casos el campo “Tema” está inhabilitado.	OK
R-08	Campo Tema habilitado	1. Seleccionar “Asunto: Consulta de Información”. 2. Seleccionar “Categoría” y decidir asignarle ningún tema.	El campo “Tema” está habilitado, mostrando una lista desplegable con las categorías del sistema.	OK
R-09	Nueva entrada en la tabla: Categoría	1. Seleccionar “Asunto: Consulta de Información”. 2. Seleccionar “Categoría” y decidir no asignarle ningún tema. 3. Confirmar la selección.	Se añade la correspondiente entrada en la tabla.	OK
R-10	Nueva entrada en la tabla: Categoría y Tema	1. Seleccionar “Asunto: Consulta de Información”. 2. Seleccionar “Categoría” y asignarle un “Tema”. 3. Confirmar la selección.	Se añade la correspondiente entrada en la tabla.	OK
R-11	Repetición consulta Tema	Consultar un tema previamente consultado en el mismo informe.	Se informa al usuario de que el tema ya ha sido consultado, y se abre de nuevo el navegador con la URL relacionada.	OK
R-12	Máximo 10 consultas	Realizar 11 consultas de categorías o temas diferentes.	Es posible realizar correctamente las 10 primeras consultas, y al intentar realizar la consulta número 11 se informa al usuario de que no es posible realizar más consultas de información. El usuario debe iniciar un nuevo registro si quiere continuar realizando consultas.	OK

Tabla 6.3 Batería de pruebas – Registro

6.4. MANTENIMIENTO

ID	Nombre	Descripción	Comportamiento esperado	Resultado
M-01	Confirmación baja	Pulsar “ELIMINAR” en el proceso de baja de un elemento del sistema por parte del administrador.	Se solicita la confirmación de dicha acción antes de llevarse a cabo.	OK
M-02	Alta y Baja de Categoría	1. Añadir una categoría al sistema. 2. Seguidamente dar de baja dicha categoría. 3. Intentar dar de baja dicha categoría de nuevo.	1. La categoría se ha añadido correctamente si es posible darla de baja a continuación. 2. La categoría se ha eliminado correctamente si ya no es posible darla de baja a continuación.	OK
M-03	Alta y Baja de Tema	1. Añadir un tema al sistema. 2. Seguidamente dar de baja dicho tema. 3. Intentar dar de baja dicho tema de nuevo.	1. El tema se ha añadido correctamente si es posible darlo de baja a continuación. 2. El tema se ha eliminado correctamente si ya no es posible darlo de baja a continuación.	OK
M-04	Baja Categoría con Tema asociado	1. Eliminar una categoría que contiene temas asociados. 2. Intentar dar de baja dicha categoría y los temas de nuevo.	1. Se informa al usuario de que la categoría contiene temas asociados, solicitando la confirmación de la eliminación de dichos temas. 2. La categoría y sus temas se han eliminado correctamente si ya no es posible darlos de baja a continuación.	OK
M-05	Alta Operador: Campo NIF incorrecto	Introducir un formato de NIF incorrecto en el informe y seleccionar el botón “ACEPTAR”.	Se informa al usuario de que el formato del NIF es incorrecto.	OK
M-06	Alta y Baja de Operador	1. Añadir un operador al sistema. 2. Seguidamente dar de baja dicho operador. 3. Intentar dar de baja dicho operador de nuevo.	1. El operador se ha añadido correctamente si es posible darlo de baja a continuación. 2. El operador se ha eliminado correctamente si ya no es posible darlo de baja a continuación.	OK

Tabla 6.4 Batería de pruebas – Mantenimiento

6.5. ESTADÍSTICAS

ID	Nombre	Descripción	Comportamiento esperado	Resultado
E-01	Selección links	Seleccionar el link a cada una de las estadísticas.	Se abre la página web que contiene el formulario relativo a cada una de ellas.	OK
E-02	Calendario	En el formulario de cada estadística, seleccionar el botón del calendario en “Fecha desde” y “Fecha hasta”.	Se muestra correctamente el calendario con la fecha actual seleccionada.	OK
E-03	Selección de fecha inicial errónea	En el calendario de “Fecha desde”, intentar seleccionar una fecha posterior a la actual.	Además de aparecer tachadas, al seleccionar una de ellas se indica al usuario mediante un mensaje que la fecha seleccionada está fuera del rango.	OK
E-04	Selección de fecha inicial correcta	En el calendario de “Fecha desde”, seleccionar una fecha anterior a la actual.	La fecha seleccionada se introduce en el campo “Fecha desde”.	OK
E-05	Selección de fecha final errónea	En el calendario de “Fecha hasta”, intentar seleccionar una fecha: Anterior a la fecha inicial introducida / Posterior a la actual.	En ambos casos, además de aparecer tachadas, al seleccionar una de ellas se indica al usuario mediante un mensaje que la fecha seleccionada está fuera del rango.	OK
E-06	Selección de fecha final correcta	En el calendario de “Fecha hasta”, seleccionar una fecha posterior a la fecha inicial introducida y anterior a la actual.	La fecha seleccionada se introduce en el campo “Fecha hasta”.	OK
E-07	Consulta de estadística	Realizar una consulta de cada estadística y comprobar con el contenido de la base de datos.	Se realiza correctamente el cálculo solicitado.	OK

Tabla 6.5 Batería de pruebas – Estadísticas

7. CONCLUSIONES Y TRABAJOS FUTUROS

7.1. ARQUITECTURA DEL SISTEMA

En cuanto a la estructura del sistema, como se detalla en la sección “3.3 Estructura de la aplicación”, se ha decidido implementar un modelo de arquitectura a medida para esta aplicación, en lugar de implementar uno de los modelos definidos por los tutoriales de Sun de JSP [56].

Dichos modelos tienen finalidades diferentes, el Modelo 1 se ajusta a aplicaciones sencillas, que no requieren de una gestión compleja de peticiones del cliente o acceso a datos, y el Modelo 2 establece un diseño basado en componentes con funciones definidas, que proporciona una serie de ventajas tanto a la hora del desarrollo como para el mantenimiento y sostenibilidad de la aplicación, pensado para aplicaciones de mayor complejidad.

La aplicación del Modelo 2 no se ha llevado a cabo en la implementación del sistema desarrollado debido a que exige ajustarse a unos requisitos que en el caso de la aplicación que nos ocupa no se han considerado necesarios. En su lugar, se ha diseñado una estructura con ciertas variaciones sobre el Modelo 2. Ambos esquemas de arquitectura se muestran en la *Figura 7.1* y la *Figura 7.2*.

Figura 7.1. Modelo 2 de arquitectura JSP

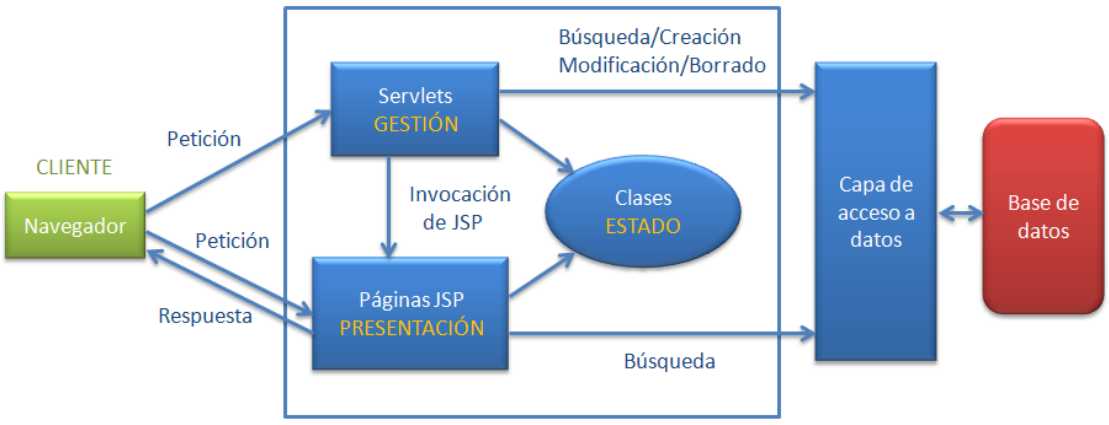


Figura 7.2. Arquitectura del sistema

A continuación, en base a las discrepancias de la arquitectura del sistema con el Modelo 2 de arquitectura JSP, se enumeran algunas posibles mejoras del sistema.

- **Unificación del destino de las peticiones del cliente.**

  La primera variación con respecto al Modelo 2 consiste en que las peticiones del cliente en el sistema diseñado pueden ir dirigidas a los servlets o a los JSPs.

  En lugar de realizar todas las peticiones a los servlets, se ha determinado realizar peticiones directamente a las páginas JSP en aquellos casos en los que la gestión por parte del servlet consistiría en redirigir la petición a dicha página JSP.

  Esta decisión se ha tomado para evitar un paso que se ha considerado innecesario, disminuyendo la carga de gestión en los casos en los que las peticiones requieren directamente un elemento del bloque de presentación.

  Una mejora del sistema consistiría en seguir el diseño del Modelo 2 de arquitectura JSP, recibiendo el controlador todas las peticiones del cliente.

  Se generaría un servlet de recepción de peticiones del cliente al que fueran dirigidas todas las peticiones, que gestionaría la delegación de la petición al siguiente elemento de la aplicación que debiera recibirla.

  De esta manera se aseguraría que todas las peticiones tuvieran un único punto de entrada a la aplicación, y se establecería un esquema común de gestión de peticiones del cliente.

  Si bien lo anterior podría realizarse mediante el uso de filtros, definidos en el descriptor de despliegue, llevando a cabo la mejora descrita se proporciona un medio más claro para implementar cuestiones transversales en la aplicación, tales como la seguridad, validaciones, etc. y se facilitan futuras modificaciones o ampliaciones del sistema.

- **Separación de la capa de control y la capa de modelo de acción.**

  Como se ha indicado previamente, en el Modelo 2 de arquitectura JSP, el bloque del modelo contiene la lógica de negocio de la aplicación, y sus componentes se dividen en dos tipos:

  – Componentes de estado.

  Mediante estos componentes se encapsula el estado de la aplicación, ofreciendo además los métodos necesarios para el acceso y las posibles modificaciones de éste.

  – Componentes de acción.

  Este tipo de componentes se utiliza para definir los posibles cambios del estado en respuesta a los eventos.

  En el caso de la aplicación desarrollada, los cambios en los elementos de la capa de estado de la aplicación son realizados por los servlets. Se ha fusionado la capa de control y la de modelo de acción en lo que se ha denominado bloque de gestión. El modelo del sistema está formado por los componentes de estado, y se denomina bloque de estado.

  El bloque gestor recibe las peticiones del cliente, de manera que cada servlet gestiona y lleva a cabo las acciones relacionadas con los componentes de estado que maneja.

  Implementando Modelo 2 de arquitectura JSP, el servlet delegaría en los componentes de acción la ejecución de las operaciones relacionadas con el estado del sistema, siendo el modelo el que se encargaría de realizar los accesos a base de datos requeridos.

  De esta manera se separa en dos bloques diferenciados la gestión de las peticiones del cliente, por parte de los servlets, y la lógica de negocio, contenida en el modelo, facilitando el mantenimiento a la aplicación.

- **Acceso a la base de datos.**

  Los accesos a la base de datos, como se puede ver en la *Figura 7.2*, se realizan desde el bloque de gestión y desde la capa de presentación, a través de la capa de acceso a datos.

  El bloque de gestión lleva a cabo operaciones CRUD (C*reate, Read, Update and Delete*) [11], en castellano Crear, Obtener, Actualizar y Borrar, sobre el contenido de la base de datos, mientras que desde el bloque de presentación solamente se llevan a cabo operaciones de tipo *Read*, u obtención de datos.

  El acceso a la base de datos desde el bloque de presentación consiste en el volcado de datos de todas las entradas del campo solicitado de una tabla, para ser mostrados al usuario en algunos formularios.

  El sistema podría mejorarse modificando la manera de obtener los datos procedentes de la base de datos por parte del bloque de presentación.

  - Habiendo implementado la mejora anterior, "Separación de la capa de control y la capa de modelo de acción", el servlet delega en el modelo las peticiones que requieran llevar a cabo accesos a base de datos. En el modelo se modifica el estado del sistema, quedando almacenada en éste la información obtenida de la base de datos.

    En este caso, el bloque de presentación, accediendo al estado de la aplicación, recuperaría los datos del modelo, presentándolos al usuario sin necesidad de recurrir al acceso a base de datos.

  - En caso de no haberse implementado la mejora mencionada, los servlets no delegan en el modelo las peticiones que requieran llevar a cabo accesos a base de datos, sino que son los servlets los que realizan los accesos a base de datos.

    La mejora consistiría en que el acceso a la base de datos, en lugar de realizarse desde el bloque de presentación, se llevara a cabo en el bloque de gestión, al igual que el resto de accesos.

    Por último, el servlet almacenaría el resultado de la operación en atributos, bien de la petición o de la sesión, para ser accedidos desde el JSP invocado y finalmente mostrados al cliente.

  Esta mejora permite unificar el acceso a base de datos desde uno de los componentes de la arquitectura, y separar las responsabilidades de los diferentes componentes.

7.2. DISEÑO DEL SISTEMA

En cuando al diseño del sistema, un ejemplo de una posible modificación del sistema está relacionado con las páginas JSP de las estadísticas.

Con respecto al formulario de introducción de parámetros para el cálculo de cada estadística, como se ha explicado previamente en la sección dedicada a las estadísticas del apartado "4.1.2 Gestión y Presentación", se implementó un formulario para cada estadística, teniendo en cuenta que se pudieran manejar diferentes parámetros para el cálculo de cada una de ellas, a indicar por el usuario con perfil de administrador en dicho formulario.

En función de este diseño, el JSP relacionado con cada estadística, teniendo en cuenta el punto de ejecución de la aplicación, muestra bien el formulario, o bien los datos obtenidos por el servlet que devuelve el resultado del cálculo, una vez realizado.

Finalmente en esta página el formulario contiene parámetros comunes para todas las estadísticas, por lo que se podría pensar en modificar el diseño inicial, unificando en una página JSP el formulario para todas las estadísticas.

Además, si se decidiera modificar los parámetros del formulario para alguna de las estadísticas, esta página JSP contendría la lógica necesaria para, en función de la estadística seleccionada, mostrar unos u otros campos en el formulario.

Con esto se mejoraría el mantenimiento global, agrupando en un solo JSP la recepción de las peticiones de todas las estadísticas, mostrando un formulario que en un principio es común a todas las estadísticas, pero que permite una posible particularización para cada una de ellas.

De esta manera también se reducen los tiempos de mantenimiento, permitiendo realizar modificaciones para todos los formularios desde un sólo JSP.

7.3. PRESENTACIÓN

También se podrían hacer algunos cambios en la capa de presentación, en relación con las estadísticas, que mejorarían el aspecto de la aplicación.

- Concretamente en la Estadística 5, que calcula los tiempos medios de atención por franjas horarias, como se puede ver en la *Figura 5.53*, se muestran todas las horas del día.

  El cambio consistiría en no mostrar las horas en las que no hay servicio, ya que siempre van a mostrar un valor nulo.

- También se podría modificar en el mismo sentido la presentación del cálculo de la Estadística 6. En este caso se calculan los tiempos medios por días de la semana, como se observa en el ejemplo de la *Figura 5.54*.

  En este caso, podría no mostrarse el tiempo medio de las consultas realizadas en domingo, ya que no se presta servicio, y será siempre nulo.

- Siguiendo la misma idea, en la Estadística 4, que calcula los tiempos medios por días como se ilustra en la *Figura 5.52*, podría de la misma manera no mostrarse los domingos abarcados por el rango temporal indicado para el cálculo.

7.4. FUNCIONALIDAD

En cuanto a las operaciones ofrecidas a los usuarios de la aplicación, se podrían ampliar, según se indica a continuación.

- El usuario administrador ejerce funciones de mantenimiento, donde puede dar de alta y de baja por un lado las categorías y temas del sistema, y también a los operadores registrados en la herramienta.

  Sería de utilidad implementar, además de las operaciones de añadir y eliminar categorías y temas, la operación de modificación de estos temas y categorías existentes. De manera sencilla, mediante el uso de la sentencia MySQL de tipo “UPDATE” [49], se puede incorporar esta funcionalidad.

  Así, cuando un administrador quisiera modificar el nombre de una categoría o de un tema, no tendría que eliminar dicho elemento del sistema y luego crearlo de nuevo con el nombre modificado, sino que podría hacer el cambio directamente.

- Una nueva funcionalidad de la aplicación podría consistir en realizar un análisis estadístico avanzado sobre los datos obtenidos en el apartado de estadísticas.

  Actualmente, en dicha sección se obtienen los elementos más consultados, y las medias temporales de los registros en función de diversos parámetros. El administrador debe analizar estos datos, y sacar las conclusiones acerca de las posibles modificaciones a realizar en el sistema.

Se podría implementar en esta sección el análisis adecuado, que permitiera identificar las modificaciones a realizar, en base a dicho análisis, de manera que el administrador recibiera directamente las directivas a llevar a cabo, por ejemplo, eliminando temas y categorías no consultados en 5 meses, o añadiendo y actualizando la información relativa a los 3 apartados más consultados.

7.5. POLÍTICAS DE SEGURIDAD

Se ha utilizado una base de datos MySQL para el almacenamiento de la información, ya que aporta ventajas como el alto rendimiento, la flexibilidad o su alta disponibilidad.

Además de estas opciones, MySQL incorpora características de seguridad para la protección de los datos [69].

MySQL por defecto utiliza conexiones sin cifrar entre el cliente y el servidor, pero ofrece soporte para conexiones seguras, cifrando la información enviada entre los clientes y el servidor, mediante el uso del protocolo SSL (Secure Sockets Layer) [35]. También es posible conectarse a un servidor MySQL remoto para obtener una conexión segura mediante SSH (Secure Shell) [12].

Una mejora del sistema consistiría en introducir estas características de seguridad, implementando estos mecanismos.

Otro posible trabajo futuro consistiría en implementar un mecanismo de backup de datos. Para ello, MySQL ofrece también utilidades de backup y recuperación por parte de MySQL y terceros, que permiten copias completas, tanto lógicas como físicas, así como otras características avanzadas de recuperación de datos.

7.6. TECNOLOGÍAS AVANZADAS

En este apartado se indican algunas de las posibles mejoras relacionadas con diversas tecnologías, empleadas comúnmente para el desarrollo de aplicaciones web J2EE.

- **Uso de etiquetas JSTL.**

  Las etiquetas JSTL [40] son utilizadas para reemplazar los scriplets en las páginas JSP. Existen varios motivos por los que resulta ventajoso evitar el uso de código Java en los JSP.

  La depuración de código Java en un JSP resulta complicada, por lo que si es posible prescindir de este código en los JSP, se evita lidiar con este escollo.

  El código de Java dentro de scriptlets en los JSP no puede ser reutilizado por otros JSP, por lo que la lógica común necesita repetirse en múltiples páginas.

  Por otro lado, los JSP deben estar libres de toda lógica de negocio, lo que apoya la idea de evitar el uso de código Java.

  Una buena manera de evitar los scriplets es sustituirlos por etiquetas JSTL, manteniendo intacta su funcionalidad y facilitando la lectura posterior a la hora de hacer posibles modificaciones.

- **Definición de un conjunto propio de etiquetas.**

  Un paso más con respecto al punto anterior consiste en la definición de etiquetas propias, práctica habitual en arquitecturas más amplias y complejas.

  El uso de estas etiquetas personalizadas permite definir mediante una única etiqueta cierta funcionalidad que requiere ser usada en múltiples puntos de la aplicación.

  En la aplicación desarrollada podría resulta útil por ejemplo para mostrar listados siempre con un mismo formato. Al incluir el recorrido del listado en la definición de la etiqueta, se consigue también reducir el tamaño de los JSP.

De esta manera se facilita el mantenimiento y la comprensión de la aplicación.

- **Integración de elementos de interfaz de usuario avanzada con *jQuery UI*.**

  La interfaz de usuario jQuery [45] proporciona componentes de utilidad a nivel de presentación para el desarrollo de aplicaciones web.

  En el caso de la aplicación desarrollada, podrían utilizarse *widgets* [14] como el *Datepicker* para simplificar el calendario actual, ya que el *Datepicker* de *jQuery UI* contiene multitud de funcionalidades integradas, como gestión de varios calendarios a la vez, o habilitación e inhabilitación de ciertos rangos de fechas.

  También se podrían utilizar menús de tipo *tabs*, otro *widget* de *jQuery UI,* para mejorar la experiencia de usuario.

- **Uso de filtros.**

  En el diseño e implementación de la aplicación no se ha considerado el uso de este tipo de elementos, pero podría ser útil para gestionar algunas cuestiones transversales de la aplicación, tales como la seguridad, o la validación de parámetros.

- **Internacionalización.**

  Aunque en este caso se trata de una aplicación local y no resulta especialmente necesario gestionar diferentes idiomas, sí es muy útil externalizar las cadenas de texto utilizadas en la aplicación, a archivos de tipo *Properties* [9], para facilitar el mantenimiento.

PRESUPUESTO

En esta sección se indica el presupuesto que podría haber sido presentado por la empresa al Gobierno de Cantabria para la asignación y realización del proyecto.

A. FASES DEL PROYECTO

La elaboración del presupuesto requiere, en primer lugar, la identificación de las fases de las que va a constar el proyecto, que en el caso de la aplicación desarrollada se detallan a continuación.

Cada una de las fases engloba un conjunto de tareas que serán llevadas a cabo en el tiempo establecido para dicha fase, y que se tienen en cuenta a la hora de realizar la planificación.

La enumeración de fases a continuación sigue un orden cronológico de ejecución, exceptuando la última fase de Jefatura de proyecto, que tiene lugar en momentos puntuales, a lo largo de la duración del proyecto.

Toma de especificaciones y requisitos

- Elaboración del documento de descripción de la aplicación
- Definición de tecnologías a utilizar (frameworks, librerías utilizadas)
- Definición del entorno de desarrollo y preproducción a partir de las especificaciones de producción
- Definición de asuntos transversales (Autenticación y autorización)

Análisis

- Elaboración del funcional de la aplicación
- Definición de la arquitectura sw de la aplicación
- Definición de la base de datos
- Definición de herramientas de desarrollo
- Estimación de recursos y tiempos

Desarrollo

- Adquisición de conocimiento
- Implementación del código fuente
- Creación de interfaces gráficas
- Creación y optimización de consultas sql
- Documentación técnica

Pruebas

- Pruebas de funcionalidades
- Depuración del código

Documentación

- Elaboración de la memoria del proyecto

Puesta en producción

- Despliegue en el servidor de aplicaciones
- Creación de esquemas de BD
- Ejecución de scripts de carga inicial
- Batería de pruebas en entorno de cliente

Jefatura de proyecto

- Gestión del proyecto
- Comunicación con el cliente final

B. ROLES ESTABLECIDOS

En base a los requerimientos del proyecto, la empresa reúne un equipo de tres personas, con los siguientes roles definidos.

Consultor Senior

Las funciones principales de este rol están relacionadas con la gestión del proyecto, englobando las tareas asociadas al trato con el cliente, como son la toma de especificaciones y requisitos al inicio, o las reuniones llevadas a cabo durante el desarrollo del proyecto.

Analista Programador

El analista programador es el responsable del proyecto, y ejerce tareas de consultor funcional y analista.

Lleva a cabo la fase de análisis, apoyo en el proceso de desarrollo y la puesta en producción del proyecto.

Programador junior

Este rol se asocia a un perfil técnico, que lleva a cabo el desarrollo de la aplicación y la fase de pruebas y depuración de código.

C. ASIGNACIÓN DE RESPONSABILIDADES

En la *Tabla P.1* se indica el rol responsable de cada una de las fases y la duración de cada una de ellas, medida en jornadas de trabajo dedicadas a cada tarea.

Fase	Responsable	Dedicación (jornadas)
Toma de especificaciones y requisitos	Consultor senior	2
Análisis	Analista programador	4
Desarrollo	Programador junior (80%) y analista programador (20%)	60
Pruebas	Programador junior	20
Documentación	Programador junior	15
Puesta en producción	Analista programador	5
Jefatura de proyecto	Consultor senior	5

Tabla P.1 Responsabilidad y dedicación

D. PLANIFICACIÓN

Mediante Diagrama de Gantt [16] que se muestra en la *Figura P.1* se representa la planificación temporal del proyecto, detallando el tiempo dedicado a cada una de las fases y las relaciones de precedencia entre las tareas establecidas.

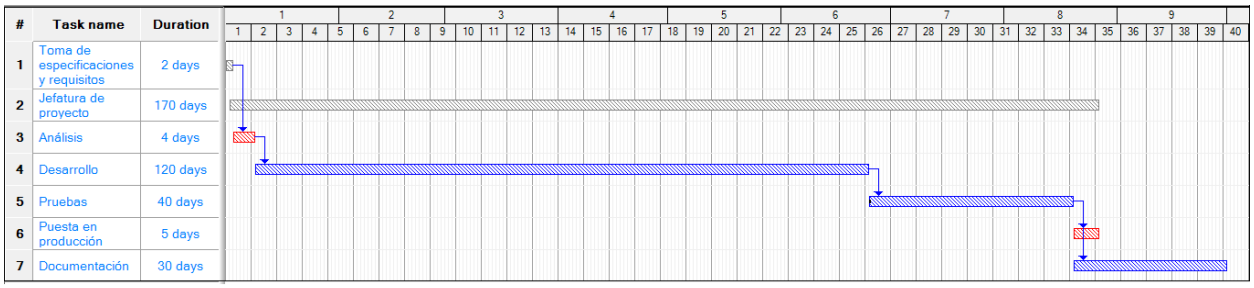


Figura P.1 Diagrama de Gantt del proyecto

Mediante el color se indica la responsabilidad de cada tarea:

- *Gris*: Consultor senior.
- *Rojo*: Analista programador.
- *Azul*: Programador junior.

En la fase de desarrollo intervienen dos roles, se ha utilizado el color azul porque predomina el rol de programador, aunque como se ha indicado previamente en el apartado C, dedicado a la asignación de responsabilidades, también interviene la figura del analista programador.

La jefatura de proyecto abarca todo el periodo, aunque la dedicación por parte del consultor senior se realiza en momentos puntuales durante este periodo, con una duración total de 5 jornadas.

Se duplica la duración de las fases de desarrollo y pruebas, debido a que la dedicación del rol de programador junior, y del analista programador en la fase de desarrollo, es de 4 horas diarias, en lugar de las 8 horas propias de la jornada completa.

E. COSTES DIRECTOS

❖ Recursos

A continuación se presenta el cálculo de los costes que consumen los recursos dedicados al proyecto, en base a los salarios establecidos y a las jornadas dedicadas por cada rol.

El salario se ha determinado en base a una ponderación de las tarifas medias ofrecidas para dichos perfiles durante el año 2011.

Como se muestra en la *Tabla P.2*, se calcula el coste por cada recurso, y por último el coste total de los recursos, calculado como la suma de los anteriores.

En la dedicación del programador junior no se incluyen las horas de documentación a la hora de calcular los costes, ya que no forma parte de los requisitos de la empresa.

Rol	Dedicación (jornadas)	Salario/jornada	Coste del recurso
Consultor senior	7	380€	2.660,00€
Analista programador	21	240€	5.040,00€
Programador junior	68	150€	10.200,00€

Tabla P.2 Coste de los recursos

El coste total de los recursos es de 17.900,00€.

❖ Material

En cuanto al material utilizado, se tienen en cuenta los equipos de los diferentes roles que forman parte del grupo de trabajo del proyecto.

Rol	Equipo	Coste
Consultor senior	TOSHIBA Satellite Pro T110-11M. C 743 / 1.3 GHz VUB - RAM 2 GB - 320 GB. XP Professional	645,00€
Analista programador	Portátil HP Compaq 8710p Intel Core 2 Duo CPU T7500 2.20GHz 3GB RAM Windows Vista Business	890,00€
Programador junior	Portátil HP Compaq 8710p Intel Core 2 Duo CPU T7500 2.20GHz 3GB RAM Windows Vista Business	890,00€

Tabla P.3 Coste de los equipos

El software que se ha utilizado en el desarrollo es de libre distribución, por lo que los gastos en material se reducen al coste de los equipos, desglosado en la *Tabla P*.3.

El coste total del material se calcula utilizando la fórmula de amortización del VII Programa Marco de Investigación y Desarrollo de la Unión Europea [23].

$\frac{A}{B} \times C \times D$, donde:

- A: periodo de utilización del equipo (en meses)
- B: periodo de amortización del equipo (en meses): habitualmente 60 meses, con excepción de los equipos informáticos, cuyo periodo de amortización es de 36 meses.
- C: base imponible del coste real del equipo
- D: porcentaje de utilización del equipo en el proyecto

En la siguiente tabla se indican los parámetros considerados para el cálculo de cada equipo:

Rol	Dedicación (jornadas)	Porcentaje de utilización	Coste	Periodo de amortización	Coste del material
Consultor senior	7	60,00%	685,00€	36 meses	3,99€
Analista programador	21	40,00%	890,00€	36 meses	10,38€
Programador junior	68	80,00%	890,00€	36 meses	67,24€

Tabla P.4 Coste del material

Por tanto, el coste total asociado al material es de 81,62€.

❖ Total

El coste directo total se calcula como la suma de los costes directos asociados a los salarios y el material, siendo éste de 17.981,62€.

F. COSTES INDIRECTOS

Es necesario tener también en cuenta los costes indirectos, para lo cual se hace una estimación del gasto que no está identificado directamente con el material o el salario del personal involucrado [10].

Estos costes tienen en cuenta el material de oficina, servicios como el agua, la luz, internet y otros gastos en los que incurre la empresa.

Los costes indirectos se calculan según el VII Programa Marco de Investigación y Desarrollo de la Unión Europea, con el método de la tarifa plana [20], teniendo en cuenta una tarifa del 20% del valor de los costes directos.

En función de estos cálculos, el coste indirecto del proyecto es de 3.596,32€.

G. COSTES TOTALES

Finalmente, se calculan los costes totales como la suma de los costes directos e indirectos, obteniendo el valor final del presupuesto, de 21.577,94€.

REFERENCIAS

1. Active Server Pages, Wikipedia – 1/diciembre/2011
   http://es.wikipedia.org/wiki/Active_Server_Pages - [Último acceso: 6/diciembre/2011]
2. Apache HTTP Server Documentation - Copyright © 2011 The Apache Software Foundation
   http://httpd.apache.org/docs/ - [Último acceso: 1/diciembre/2011]
3. Apache Tomcat - Copyright © 1999-2011, The Apache Software Foundation
   http://tomcat.apache.org/ - [Último acceso: 1/diciembre/2011]
4. Aplicaciones Web de Servidor. Arquitectura y diseño: Patrón MVC – 9/ Mayo/2008
   http://www.vc.ehu.es/jiwotvim/ISOFT2007-2008/Teoria/BloqueV/MVC-tr.ppt -
   [Último acceso: 1/diciembre/2011]
5. Bean, Wikipedia – 28/septiembre/2011
   http://es.wikipedia.org/wiki/Bean - [Último acceso: 4/diciembre/2011]
6. CalendarXP.net - Copyright© 2003- 2011 Idemfactor Solutions Inc.
   http://www.calendarxp.net/ - [Último acceso: 3/diciembre/2011]
7. Cantabria.es - Gobierno de Cantabria
   http://www.gobcantabria.es/ - [Último acceso: 4/diciembre/2011]
8. Class GenericServlet – JavaTM 2 Platform, Enterprise Edition, v 1.3 API Specification
   http://docs.oracle.com/javaee/1.3/api/javax/servlet/GenericServlet.html -
   [Último acceso: 1/diciembre/2011]
9. Class Properties, JavaTM 2 Platform Std. Ed. v1.4.2 - © 2003, 2010 Oracle and/or its affiliates.
   http://docs.oracle.com/javase/1.4.2/docs/api/java/util/Properties.html -
   [Último acceso: 4/diciembre/2011]
10. Coste indirecto, Wikipedia – 7/octubre/2010
    http://es.wikipedia.org/wiki/Coste_indirecto - [Último acceso: 4/diciembre/2011]
11. Create, read, update and delete, Wikipedia - 17 November 2011
    http://en.wikipedia.org/wiki/Create,_read,_update_and_delete - [Último acceso: 4/diciembre/2011]
12. Criptografía - Caparazón Seguro (protocolo SSH)
    http://es.kioskea.net/contents/crypto/ssh.php3#q=ssh&cur=3&url=%2F –
    [Último acceso: 6/diciembre/2011]
13. Data Access Object, Wikipedia – 27/mayo/2011
    http://es.wikipedia.org/wiki/Data_Access_Object - [Último acceso: 4/diciembre/2011]
14. Demos & Documentation, JQuery User Interface - © 2010 The jQuery Project and the jQuery UI Team.
    http://jqueryui.com/demos/ - [Último acceso: 4/diciembre/2011]
15. Designing Enterprise Applications with the J2EETM Platform, Second Edition - © 2010, Oracle Corporation and/or its affiliates
    http://java.sun.com/blueprints/guidelines/designing_enterprise_applications_2e/ -

[Último acceso: 4/diciembre/2011]

16. Diagrama de Gantt, Wikipedia – 15/noviembre/2011

    http://es.wikipedia.org/wiki/Diagrama_de_Gantt - [Último acceso: 4/diciembre/2011]

17. El ABC de JDBC - Copyright (c) 2003, Abraham Otero.

    http://www.javahispano.org/portada/2011/8/1/el-abc-de-jdbc.html - [Último acceso: 1/diciembre/2011]

18. Enterprise JavaBeans 3.0 Final Release, Java Community Process – © 2011, Oracle Corporation and/or its affiliates.

    http://download.oracle.com/otndocs/jcp/ejb-3_0-fr-oth-JSpec/ - [Último acceso: 1/diciembre/2011]

19. Facebook: Gobierno de Cantabria. Desde 6/mayo/2009.

    http://www.facebook.com/gobcantabria - [Último acceso: 4/diciembre/2011]

20. Factsheet - Indirect costs in FP7

    http://docs.energiehelpline.co.uk/Indirect_cost_calculation_March_09.pdf -

    [Último acceso: 4/diciembre/2011]

21. Foreign key, Databases, About.com - ©2011 About.com.

    http://databases.about.com/cs/specificproducts/g/foreignkey.htm - [Último acceso: 4/diciembre/2011]

22. GROUP BY (Aggregate) Functions , MySQL 5.0 Reference Manual

    http://dev.mysql.com/doc/refman/5.0/en/group-by-functions.html - [Último acceso: 4/diciembre/2011]

23. GUÍA BÁSICA PARA LA PREPARACIÓN DE PROPUESTAS AL SÉPTIMO PROGRAMA MARCO

    http://www.upct.es/opect/docs/pub/11GUIA%20BASICA%20PREPARACION%20DE%20PROPUESTAS%207PM.pdf - [Último acceso: 4/diciembre/2011]

24. Guía Breve de CSS, W3C World Wide Web Consortium - 9/enero/2008

    http://www.w3c.es/divulgacion/guiasbreves/hojasestilo - [Último acceso: 4/diciembre/2011]

25. HTTP Made Really Easy - © 1997 James Marshall

    http://www.jmarshall.com/easy/http/ - [Último acceso: 1/diciembre/2011]

26. InnoBD y AUTOCOMMIT, MySQL - © 1997, 2011, Oracle and/or its affiliates.

    http://dev.mysql.com/doc/refman/5.0/es/innodb-and-autocommit.html -

    [Último acceso: 4/diciembre/2011]

27. Interface HttpServlet – JavaTM 2 Platform, Enterprise Edition, v 1.3 API Specification

    http://docs.oracle.com/javaee/1.3/api/javax/servlet/http/HttpServlet.html -

    [Último acceso: 1/diciembre/2011]

28. Interface HttpServletRequest – JavaTM 2 Platform, Enterprise Edition, v 1.3 API Specification

    http://docs.oracle.com/javaee/1.3/api/javax/servlet/http/HttpServletRequest.html -

    [Último acceso: 1/diciembre/2011]

29. Interface HttpServletResponse – JavaTM 2 Platform, Enterprise Edition, v 1.3 API Specification

    http://docs.oracle.com/javaee/1.3/api/javax/servlet/http/HttpServletResponse.html -

    [Último acceso: 1/diciembre/2011]

30. Interface RequestDispatcher – JavaTM 2 Platform, Enterprise Edition, v 1.3 API Specification

http://download.oracle.com/javaee/1.3/api/javax/servlet/RequestDispatcher.html -

[Último acceso: 4/diciembre/2011]

31. Interface Servlet – JavaTM 2 Platform, Enterprise Edition, v 1.3 API Specification

http://docs.oracle.com/javaee/1.3/api/javax/servlet/Servlet.html -

[Último acceso: 1/diciembre/2011]

32. Interface ServletContext – JavaTM 2 Platform, Enterprise Edition, v 1.3 API Specification

http://download.oracle.com/javaee/1.3/api/javax/servlet/ServletContext.html -

[Último acceso: 4/diciembre/2011]

33. Interface ServletRequest – JavaTM 2 Platform, Enterprise Edition, v 1.3 API Specification

http://docs.oracle.com/javaee/1.3/api/javax/servlet/ServletRequest.html -

[Último acceso: 1/diciembre/2011]

34. Interface ServletResponse – JavaTM 2 Platform, Enterprise Edition, v 1.3 API Specification

http://docs.oracle.com/javaee/1.3/api/javax/servlet/ServletResponse.html -

[Último acceso: 1/diciembre/2011]

35. Introducción a SSL, Criptografía - Secure Sockets Layers (SSL)

http://es.kioskea.net/contents/crypto/ssl.php3 - [Último acceso: 6/diciembre/2011]

36. Introducción a WebSphere, IBM developerWorks R

http://www.ibm.com/developerworks/ssa/websphere/newto/index.html -

[Último acceso: 1/diciembre/2011]

37. JARS, WARS, EARS - 7/septiembre/2005

http://www.osmosislatina.com/java/wars.htm - [Último acceso: 4/diciembre/2011]

38. Java Database Programming - Copyright © 1996 MageLang Institute.

http://java.sun.com/developer/onlineTraining/Database/JDBCShortCourse/jdbc/jdbc.html - [Último acceso: 1/diciembre/2011]

39. JavaScript, Wikipedia – 13/noviembre/2011

http://es.wikipedia.org/wiki/JavaScript - [Último acceso: 1/diciembre/2011]

40. JavaServer Pages Standard Tag Library, The J2EE(TM) 1.4 Tutorial

http://docs.oracle.com/javaee/1.4/tutorial/doc/JSTL.html - [Último acceso: 1/diciembre/2011]

41. JavaTM 2 Platform Std. Ed. v1.4.2. Interface PreparedStatement- Copyright © 2003, 2010 Oracle and/or its affiliates

http://download.oracle.com/javase/1.4.2/docs/api/java/sql/PreparedStatement.html -

[Último acceso: 4/diciembre/2011]

42. JavaTM 2 Platform Std. Ed. v1.4.2. Interface ResultSet - Copyright © 2003, 2010 Oracle and/or its affiliates

http://download.oracle.com/javase/1.4.2/docs/api/java/sql/ResultSet.html - [Último acceso: 4/diciembre/2011]

43. JDBC - Java Database Connectivity Tutorial – 12/Marzo/2008 http://www.roseindia.net/jdbc/index.shtml - [Último acceso: 1/diciembre/2011]
44. JDBC Overview - Oracle Corporation http://www.oracle.com/technetwork/java/overview-141217.html - [Último acceso: 1/diciembre/2011]
45. JQuery User Interface - © 2010 The jQuery Project and the jQuery UI Team. http://jqueryui.com/home - [Último acceso: 4/diciembre/2011]
46. Linkedin: Empresas > Gobierno de Cantabria http://www.linkedin.com/company/279052?trk=tyah - [Último acceso: 4/diciembre/2011]
47. Manejo de datos de la sesión en Servlets – Enero/2006 http://www.proactiva-calidad.com/java/servlets/sesion.html - [Último acceso: 4/diciembre/2011]
48. MySQL Tutorial - Copyright © by www.mysqltutorial.org - Desde 2008 http://www.mysqltutorial.org/ - [Último acceso: 1/diciembre/2011]
49. MySQL UPDATE Query, Tutorials point. http://www.tutorialspoint.com/mysql/mysql-update-query.htm - [Último acceso: 5/diciembre/2011]
50. Netscape Server Application Programming Interface, Wikipedia – 13/agosto/2011 http://en.wikipedia.org/wiki/Netscape_Server_Application_Programming_Interface - [Último acceso: 1/diciembre/2011]
51. OnJ2EE. Introducción Servlets – 7/diciembre/2009 http://onj2ee.blogspot.com/2009/12/introduccion-servlets.html- [Último acceso: 4/diciembre/2011]
52. OnJ2EE. Session Management (SCWCD) - 8/enero/2010 http://onj2ee.blogspot.com/2010/01/session-management.html - [Último acceso: 4/diciembre/2011]
53. Outlook Express, Wikipedia – 3/septiembre/2011 http://es.wikipedia.org/wiki/Outlook_Express - [Último acceso: 1/diciembre/2011]
54. Prepared Statement Set Object. Rose India Technologies Pvt. Ltd. – 13/abril/2007 http://www.roseindia.net/jdbc/jdbc-mysql/PreparedStatementSetObject.shtml - [Último acceso: 4/diciembre/2011]
55. Primary key definition, Databases, About.com - ©2011 About.com. http://databases.about.com/cs/administration/g/primarykey.htm - [Último acceso: 4/diciembre/2011]
56. SCWCD Exam Study Kit - Java Web Component Developer Certification (2nd Edition 2005), Hanumant Deshmukh, Jignesh Malavia, and Matthew Scarpino. Manning Publications Co. http://onj2ee.blogspot.com/2010/01/ebooks-scwcd.html - [Último acceso: 1/diciembre/2011]
57. Server-side JavaScript, Wikipedia – 28/noviembre/2011 http://en.wikipedia.org/wiki/Server-side_JavaScript - [Último acceso: 1/diciembre/2011]
58. Servlet API Documentation - Copyright © 1999-2011 The Apache Software Foundation. http://tomcat.apache.org/tomcat-5.5-doc/servletapi/index.html - [Último acceso: 1/diciembre/2011]
59. Sintaxis SQL de sentencias preparadas, MySQL - © 1997, 2011, Oracle and/or its affiliates.

http://dev.mysql.com/doc/refman/5.0/es/sqlps.html - [Último acceso: 4/diciembre/2011]

60. SQL Tutorial, schools.com
    http://www.w3schools.com/sql/default.asp - [Último acceso: 1/diciembre/2011]
61. Sun Java System Application Server Platform Edition 8, The J2EE(TM) 1.4 Tutorial
    http://docs.oracle.com/javaee/1.4/tutorial/doc/Overview8.html - [Último acceso: 1/diciembre/2011]
62. The J2EE™ Tutorial. Packaging Web Components - Copyright © 2010 Oracle and/or its affiliates
    http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/WebComponents3.html -
    [Último acceso: 4/diciembre/2011]
63. The Java Tutorials - Copyright © 1995, 2011 Oracle and/or its affiliates
    http://download.oracle.com/javase/tutorial/deployment/jar - [Último acceso: 4/diciembre/2011]
64. Tutorial de FTP, desarrolloweb.com – Diciembre/ 2005
    http://www.desarrolloweb.com/manuales/72/ - [Último acceso: 1/diciembre/2011]
65. Tutoriales HTML, htmlquick.com - Diego Ponce de León
    http://www.htmlquick.com/es/tutorials.html - [Último acceso: 1/diciembre/2011]
66. Twitter: cantabria.es - Gobierno de Cantabria. Desde 21/ enero/ 2009.
    http://twitter.com/#!/cantabriaes - [Último acceso: 4/diciembre/2011]
67. Understanding Web Security Using web.xml Via Use Cases – 11/febrero/2009
    http://java.dzone.com/articles/understanding-web-security - [Último acceso: 4/diciembre/2011]
68. URI URL URN Relationship - © 2011 Digital Purview
    http://www.digitalpurview.com/uri-url-urn-relationship/ - [Último acceso: 1/diciembre/2011]
69. Using Secure Connections, dotConnect for MySQL- © 2002-2011 Devart.
    http://www.devart.com/dotconnect/mysql/docs/SecureConnections.html -
    [Último acceso: 6/diciembre/2011]
70. VBScript, Wikipedia – 2/noviembre/2011
    http://es.wikipedia.org/wiki/VBScript - [Último acceso: 1/diciembre/2011]
71. Youtube: INNOVACION de portalgobcantabria – 15/noviembre/2010
    http://www.youtube.com/user/portalgobcantabria - [Último acceso: 4/diciembre/2011]
72. WAR (archivo), Wikipedia – 10/marzo/2011
    http://es.wikipedia.org/wiki/WAR_(archivo) - [Último acceso: 4/diciembre/2011]
73. Web Server, Web Container & Application Server – 10/junio/ 2008
    http://geekexplains.blogspot.com/2008/06/web-server-web-container-application.html -
    [Último acceso: 1/diciembre/2011]