



## Sistema para la publicación y difusión de noticias

Departamento de Arquitectura y Tecnología de los  
Computadores

## Proyectos Informáticos

Ángel Daniel Sanjuán Espejo

David Armenteros Escabias

Tutores:

José Luis Bernier Villamor

José Carlos Calvo Tudela

Junio de 2011



D. José Luis Bernier Villamor y D. José Carlos Calvo Tudela, profesores del Departamento de Arquitectura y Tecnología de los Computadores de la Universidad de Granada, como Directores del Proyecto Informático de los alumnos D. Ángel Daniel Sanjuán Espejo y D. David Armenteros Escabias,

DECLARAN

que este trabajo, con título “BLUFEEDME: Sistema para integración y publicación de noticias” ha sido realizado por los alumnos mencionados y autorizan su defensa ante el tribunal que corresponda.

Y para que conste, firmarán el presente documento, en Granada, a \_\_\_\_ de julio de 2011.

D. José Luis Bernier Villamor

D. José Carlos Calvo Tudela



D. Ángel Daniel Sanjuán Espejo, con DNI nº 75172388A, y D. David Armenteros Escabias, con DNI nº 77349194X, alumnos de Ingeniería Informática de la Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada, autorizan que se expida una copia del presente documento de Proyecto Informático a la Biblioteca del Centro, a fin de poder ser consultada o referenciada por aquellas personas que lo deseen.

Granada, a \_\_\_\_ de julio de 2011

D. Ángel Daniel Sanjuán Espejo

D. David Armenteros Escabias



*A las personas que más aprecio*



## Prólogo

Hoy día, la mayor parte de los usuarios ligados a una institución reciben gran cantidad de información a través de Internet. Dicha información, en el caso de la Universidad de Granada, se encuentra bastante dispersa en diferentes plataformas web. La obligación de obtener información de un elevado número de fuentes hace el día a día del usuario más laborioso, pudiendo llegar a visitar diariamente alrededor de seis sitios web diferentes para obtener información sobre la evolución del curso, gestiones administrativas y ofertas de ocio relacionadas con la universidad, invirtiendo un tiempo considerable.

Por todo lo anteriormente mencionado, es indudable que se necesitan herramientas que unifiquen la información ofrecida por las diferentes plataformas. Con estas herramientas se conseguiría que toda la información interesante para el usuario le sea transmitida de una forma inmediata y sin necesidad de que realice una labor de búsqueda, recolección o filtrado. Con esto se facilitaría en gran medida la interacción y relación diaria entre el usuario y el organismo o empresa, en el caso, la Universidad de Granada.

Hasta ahora era todo un reto conseguir que los distintos sistemas de información fueran “transparentes” entre sí tanto en el hardware como en el software, de tal manera que los usuarios de los servicios no estén “enterados” de la estructura sobre el que funcionan. En la actualidad está en auge los conceptos como cloud computing, Web Services, redes sociales o computación online que permiten que tanto los usuarios como otros sistemas accedan a distintos servicios de una forma estándar, transparente y accesible a todo el mundo a través de la web.

El proyecto BLUFEEDME, presenta un procedimiento estándar basado en Web Services con toda la funcionalidad necesaria, que garantiza y abarca todas las ventajas mencionadas. La facilidad de integración en otros sistemas que ofrece BLUFEEDME permite aunar la información procedente de múltiples sistemas externos, resolviendo el problema de la dispersión de la información.

Para cualquier sistema de publicación actual es imprescindible poner a disposición de los usuarios diferentes medios para acceder a la información. Siendo consecuente en la idea de hacer llegar la información a través de diferentes canales, el proyecto BLUFEEDME acentúa fundamentalmente las ideas de personal y accesible.

Permite el acceso de forma inmediata, captando la atención del usuario, a través de pantallas colocadas y distribuidas de forma estratégica a lo largo de todo el edificio. De esta forma se concede al usuario la posibilidad de obtener información de carácter general mientras desarrolla su actividad diaria en el

edificio.

En lo referente a lo personal el proyecto BLUFEEDME se hace cercano al usuario mediante el uso de tecnología Bluetooth. Hoy día un dispositivo móvil se convierte en un complemento diario más y en la mayoría de los casos necesario. BLUFEEDME aprovecha este elemento para ofrecer una información personal, filtrando la información, y acentuando la sensación en el usuario de que es una pieza más en todo el entramado de la organización del organismo (Universidad de Granada).

Además, el hecho de que la información llegue al usuario de forma directa, sin que éste tenga que ir a buscarla, y que al almacenarla de forma local en su dispositivo móvil disponga de ella en cualquier momento, hace aún más atractivo al proyecto BLUFEEDME. La posibilidad de que el usuario pueda almacenar la información aumenta el radio de publicación de noticias, permitiendo que el usuario se convierta en un medio más de difusión de las mismas.

Por otra parte, la tecnología Bluetooth recientemente ha adquirido mayor popularidad debido a su bajo consumo eléctrico, por ser tecnología de tipo inalámbrica y su bajo costo. Se incluye en una gran variedad de productos tales como dispositivos móviles, impresoras, modems, entre otros.

# Agradecimientos

En primer lugar agradecer a nuestros tutores, José Luis Bernier Villamor y José Carlos Calvo Tudela, el habernos brindado la oportunidad de desarrollar el presente proyecto y la gran ayuda facilitada en los momentos necesarios durante el desarrollo del mismo. Agradecer también la cercanía que han tenido con nosotros y por habernos propuesto este proyecto tan interesante, útil y diferente, que ha hecho que su elaboración nos fuera más amena.

Agradecer a todas y cada una de las personas junto a las que he crecido y me han inculcado cada uno de los valores que hoy forman parte de mi.

De forma especial ofrecer un agradecimiento a mis padres por todo su esfuerzo y sacrificio durante todos estos años para hacer posible el camino que me ha conducido hasta aquí, por sus consejos y apoyo en cada una de las decisiones tomadas; porque gracias a ello he llegado a realizar la más grande de mis metas.

A mi padre por regalarme las palabras que, en momentos complicados, me han vuelto a hacer confiar en mi mismo y que gracias a ello, hoy veo llegar a su fin una de las etapas de mi vida. A mis hermanos por estar siempre ahí para ayudarme y por todo el apoyo que me brindan diariamente.

A mi madre, por el gran amor y la devoción que tienes a tus hijos, por el apoyo incondicional que siempre me has dado, por haberme formado como un hombre de bien, no hay palabras en este mundo para agradecerte.

Por último, agradecer a la persona que ha estado junto a mí en cada uno de los momentos vividos en el transcurso del desarrollo del proyecto, el conseguir aportar ilusión a mi día a día, ser el pilar sobre el que me he apoyado durante estos meses, esa persona que me llena día a día y me enseña lo que es querer y ser querido y que sigue a mi lado pese a las dificultades que han surgido en todo este tiempo; gracias por todo.

A todos mis amigos y compañeros de universidad por todos los buenos y malos momentos que hemos vivido durante la carrera, tanto dentro como fuera de la universidad.



# Índice general

<b>1 Estructura de la memoria.....</b>	<b>21</b>
1.1 Estructura del documento.....	21
1.2 Estructura de los elementos adjuntos al proyecto.....	23
<b>2 Introducción.....</b>	<b>24</b>
2.1 Motivación del proyecto.....	25
2.2 Análisis de mercado.....	26
2.2.1 Proyectos basados en cliente-servidor.....	27
2.2.2 Proyectos basados en transferencia de ficheros.....	27
2.3 Objetivos del proyecto.....	28
2.4 Licencia y liberación de código.....	29
2.5 Glosario de Términos.....	31
<b>3 Solución para el proyecto.....</b>	<b>33</b>
3.1 Sistema Integrable.....	33
3.2 Accesibilidad.....	36
3.3 Interacción cercana al usuario.....	38
3.4 Escenario.....	38
3.5 Arquitectura Física o Hardware.....	39
3.6 Arquitectura Software.....	42
3.6.1 Modelo de Arquitectura Software.....	43
3.6.2 Lenguajes de programación.....	44
3.6.2.1 PHP.....	44
3.6.2.2 JAVA.....	46
<b>4 Metodología de desarrollo y herramientas.....</b>	<b>49</b>
4.1 Metodología General.....	50
4.2 Determinación de recursos.....	54
4.2.1 Herramientas Software.....	54
4.2.1.1 Lenguajes de programación.....	54
4.2.1.2 IDEs (entornos de desarrollo).....	55
4.2.1.3 Base de Datos.....	55
4.2.1.4 Librerías.....	56
4.2.1.5 Navegadores web.....	57
4.2.1.6 Servidores.....	57
4.2.1.7 Software Planificación, Modelado, Análisis y Diseño.....	58
4.2.1.8 Control de versiones.....	59
4.2.1.9 Paquetes ofimáticos y documentación.....	60
4.2.1.10 Sistemas Operativos.....	60
4.2.2 Herramientas Hardware.....	60
4.3 Estándares seguidos.....	61
<b>5 Desarrollo software del proyecto.....</b>	<b>65</b>
5.1 Planificación.....	65
5.1.1 Declaración de alcance.....	65
5.1.1.1 Objetivo general.....	65
5.1.1.2 Objetivos iteraciones.....	66
5.1.2 Planificación temporal y organizativa.....	71

5.2 Modelado del sistema.....	88
5.2.1 Especificación de requisitos.....	88
5.2.1.1 Funcionales.....	88
5.2.1.2 No Funcionales.....	92
5.2.2 Modelo funcional y subsistemas funcionales.....	96
5.2.2.1 Web.....	96
5.2.2.2 Bluetooth.....	99
5.3 Análisis y Diseño del sistema.....	101
5.3.1 Web.....	101
5.3.1.1 Análisis: Modelo estático.....	102
5.3.1.2 Diseño: Arquitectura.....	104
5.3.1.3 Diseño: Interfaz de usuario.....	124
5.3.2 Bluetooth.....	135
5.3.2.1 Análisis: Modelo estático.....	135
5.3.2.2 Diseño: Arquitectura.....	139
5.3.2.3 Diseño: Interfaz de usuario.....	147
5.3.3 Base de Datos.....	152
5.3.3.1 Modelización semántica.....	152
<b>6 Detalles de la implementación.....</b>	<b>157</b>
6.1 Subsistema Gestión Web Service y cliente Web Service.....	157
6.1.1 Web Service.....	157
6.1.2 Cliente Web Service.....	163
6.2 Subsistemas de Publicación de noticias.....	176
6.2.1 Subsistema Web pantallas.....	176
6.2.2 Subsistema Bluetooth.....	181
6.2.2.1 Clase DispositivoBT.....	181
6.2.2.2 Clase EnviarAdispositivo.....	184
6.2.2.3 Concurrencia.....	186
6.2.2.4 Interfaz.....	190
<b>7 Manual de usuario.....</b>	<b>193</b>
7.1 Puesta en marcha y mantenimiento.....	193
7.1.1 Instalación/Configuración.....	193
7.1.1.1 Base de datos.....	193
7.1.1.2 Servicio Web.....	193
7.1.1.3 Web pantallas.....	198
7.1.1.4 Aplicación Bluetooth.....	198
7.1.2 Mantenimiento.....	199
7.2 Guía usuario.....	200
7.2.1 Web Service.....	200
7.2.2 Cliente Web Service.....	207
7.2.3 Pantallas.....	217
7.2.4 Bluetooth.....	218
7.2.4.1 Administrador.....	218
7.2.4.2 Usuario.....	221
<b>8 Conclusiones.....</b>	<b>222</b>
<b>9 Anexos.....</b>	<b>224</b>
9.1 Anexo I: Descripción detallada de los casos de uso.....	224
9.1.1 Subsistema Web.....	225

9.1.2 Subsistema Bluetooth.....	241
9.2 Anexo II: Diccionario de datos análisis y diseño.....	248
9.3 Anexo III: Diccionario de datos Base de Datos.....	257
9.4 Anexo IV: Web Service.....	261
Introducción.....	261
Aplicaciones clásicas frente a aplicaciones orientadas a servicios SOA.....	261
Arquitectura SOA y modelo conceptual (JAVA).....	262
Fundamentos Arquitectura Orientada a servicios. Web Services.....	265
SOAP.....	267
WSDL.....	272
UDDI.....	283
Web Services JAVA.....	284
9.5 Anexo V: Bluetooth:JSR 82.....	290
Bluetooth.....	290
Nociones sobre Bluetooth.....	290
Establecimiento de la conexión.....	293
APIs Java para Bluetooth.....	294
Inicialización.....	295
BCC (Bluetooth Control Center).....	295
Inicialización de la Pila.....	295
Discovery.....	296
Descubrir dispositivos (Device discovery).....	296
Descubrir servicios (Service Discovery).....	297
Clases del Service Discovery.....	297
Registro del servicio(Service Registration).....	299
Comunicación.....	302
Perfil del puerto serie (SPP).....	302
Establecimiento de la conexión.....	304
L2CAP.....	306
Protocolo de intercambio de objetos (OBEX).....	310
<b>Bibliografía.....</b>	<b>315</b>

# Índice de ilustraciones

Ilustración 3.5.1: Arquitectura Hardware.....	41
Ilustración 4.1.1: Proceso de desarrollo OO iterativo simplificado.....	51
Ilustración 4.1.2: Desarrollo OO iterativo o incremental.....	53
Ilustración 5.1.1: Planificación temporal iteración 1.....	72
Ilustración 5.1.2: Planificación recursos humanos iteración 1.....	75
Ilustración 5.1.3: Planificación temporal iteración 2.....	76
Ilustración 5.1.4: Planificación recursos iteración 2.....	79
Ilustración 5.1.5: Planificación temporal iteración 3.....	80
Ilustración 5.1.6: Planificación recursos iteración 3.....	83
Ilustración 5.1.7: Planificación temporal iteración 4.....	84
Ilustración 5.1.8: Planificación recursos humanos iteración 4.....	87
Ilustración 5.2.1: Diagrama Casos de Uso subsistema Web.....	98
Ilustración 5.2.2: Diagrama Casos de Uso subsistema Bluetooth.....	100
Ilustración 5.3.1: Diagrama clases análisis.....	103
Ilustración 5.3.2: Arquitectura Multicapa.....	105
Ilustración 5.3.3: Arquitectura Cliente/Servidor.....	106
Ilustración 5.3.4: Arquitectura Par a par.....	107
Ilustración 5.3.5: Arquitectura de repositorio.....	108
Ilustración 5.3.6: Arquitectura transformacional; Ejemplo.....	109
Ilustración 5.3.7: Arquitectura Modelo-Vista-Controlador.....	109
Ilustración 5.3.8: Arquitectura subsistema Web Service en capas.....	111
Ilustración 5.3.9: Diagrama clases paquete implementación.....	112
Ilustración 5.3.10: Clase NewsService.....	113
Ilustración 5.3.11: Diagrama paquetes de paquete service.....	113
Ilustración 5.3.12: Diagrama clases paquete exception.....	114
Ilustración 5.3.13: Comunicación Cliente Web Service-Servidor Web Service.....	117
Ilustración 5.3.14: Arquitectura subsistema web pantallas en capas.....	119
Ilustración 5.3.15: Esquema de diseño repositorio.....	120
Ilustración 5.3.16: Diagrama de despliegue subsistemas Web Service y pantallas.....	123
Ilustración 5.3.17: Interfaz Web Pantallas.....	124
Ilustración 5.3.18: Interfaz cliente Web Service; Pantalla principal.....	125
Ilustración 5.3.19: Interfaz cliente Web Service; Login.....	126
Ilustración 5.3.20: Interfaz cliente Web Service; Menú.....	126
Ilustración 5.3.21: Interfaz cliente Web Service; Formulario Introducir Noticia.....	127
Ilustración 5.3.22: Interfaz cliente Web Service; Formulario Modificar Noticia.....	128
Ilustración 5.3.23: Interfaz cliente Web Service; Formulario Eliminar Noticia.....	129
Ilustración 5.3.24: Interfaz cliente Web Service; Formulario Eliminar Noticia(continuación).....	129
Ilustración 5.3.25: Interfaz cliente Web Service; Lista Noticias.....	130
Ilustración 5.3.26: Interfaz cliente Web Service; Formulario Introducir Dispositivo.....	131
Ilustración 5.3.27: Interfaz cliente Web Service; Formulario Modificar Dispositivo.....	132
Ilustración 5.3.28: Interfaz cliente Web Service; Formulario Eliminar Dispositivo.....	132
Ilustración 5.3.29: Interfaz cliente Web Service; Formulario Asociar Dispositivo.....	133
Ilustración 5.3.30: Interfaz cliente Web Service; Formulario Desasociar Dispositivo....	134

Ilustración 5.3.31: Interfaz cliente Web Service; Formulario Lista Categorías.....	135
Ilustración 5.3.32: Diagrama de clases de análisis del modelo.....	136
Ilustración 5.3.33: Diagrama de clases de análisis subsistema bluetooth.....	137
Ilustración 5.3.34: Diagrama de clases análisis Bluetooth completo.....	138
Ilustración 5.3.35: Diseño arquitectura Bluetooth.....	140
Ilustración 5.3.36: Diagrama de clases Bluetooth de diseño (parcial).....	141
Ilustración 5.3.37: Diagrama del diseño de la interfaz gráfica bluetooth.....	143
Ilustración 5.3.38: Diagrama de clases de diseño bluetooth completo.....	144
Ilustración 5.3.39: Diagrama de paquetes bluetooth.....	145
Ilustración 5.3.40: Diagrama de despliegue bluetooth.....	147
Ilustración 5.3.41: Aspecto de fichero recibido en dispositivo móvil.....	148
Ilustración 5.3.42: Pantalla ficheros de propiedades bluetooth.....	149
Ilustración 5.3.43: Pantalla de auditoría del proceso de envío bluetooth.....	150
Ilustración 5.3.44: Diálogo información de la aplicación bluetooth.....	152
Ilustración 5.3.45: Base de Datos; Diagrama Entidad-Relación.....	154
Ilustración 6.1.1: Algoritmo SHA-1.....	160
Ilustración 6.2.1: Diagrama secuencia (hebras).....	188
Ilustración 6.2.2: Diagrama dependencias de ejecución.....	189
Ilustración 7.1.1: Lista de puertos Glassfish.....	194
Ilustración 7.1.2: Web Administración Glassfish.....	195
Ilustración 7.1.3: Web Administración Glassfish (opción implementar).....	196
Ilustración 7.1.4: Web Administración Glassfish (opción implementar) cont.....	196
Ilustración 7.1.5: Web Administración Glassfish (opción implementar) cont. II.....	197
Ilustración 7.1.6: Web Administración Glassfish (opción implementar) cont. III.....	197
Ilustración 7.2.1: Manual cliente Web Service; Pantalla principal.....	207
Ilustración 7.2.2: Manual cliente Web Service; Login.....	208
Ilustración 7.2.3: Manual cliente Web Service; Menú.....	208
Ilustración 7.2.4: Manual cliente Web Service; Formulario Introducir Noticia.....	209
Ilustración 7.2.5: Manual cliente Web Service; Formulario Modificar Noticia.....	210
Ilustración 7.2.6: Manual cliente Web Service; Formulario Eliminar Noticia.....	211
Ilustración 7.2.7: Manual cliente Web Service; Formulario Eliminar Noticia(continuación).....	211
Ilustración 7.2.8: Manual cliente Web Service; Lista Noticias.....	212
Ilustración 7.2.9: Manual cliente Web Service; Formulario Introducir Dispositivo.....	213
Ilustración 7.2.10: Manual cliente Web Service; Formulario Modificar Dispositivo.....	214
Ilustración 7.2.11: Manual cliente Web Service; Formulario Eliminar Dispositivo.....	214
Ilustración 7.2.12: Manual cliente Web Service; Formulario Asociar Dispositivo.....	215
Ilustración 7.2.13: Manual cliente Web Service; Formulario Desasociar Dispositivo.....	216
Ilustración 7.2.14: Manual cliente Web Service; Formulario Lista Categorías.....	217
Ilustración 7.2.15: Pantalla ficheros de propiedades bluetooth.....	218
Ilustración 7.2.16: Pantalla de auditoría del proceso de envío bluetooth.....	219
Ilustración 9.4.1: Aplicaciones clásicas VS Aplicaciones SOA.....	262
Ilustración 9.4.2: Arquitectura n-layer.....	263
Ilustración 9.4.3: Componente orientado a servicios.....	264
Ilustración 9.4.4: Estructura Web Service.....	266
Ilustración 9.4.5: Estándares empleados en proceso de comunicación.....	267
Ilustración 9.4.6: Comunicación consumidor-proveedor web services.....	268
Ilustración 9.4.7: Estructura mensaje SOAP.....	270

Ilustración 9.4.8: Mensaje SOAP request.....	271
Ilustración 9.4.9: Mensaje SOAP response.....	271
Ilustración 9.4.10: Estructura WSDL.....	272
Ilustración 9.4.11: Ejemplo WSDL.....	274
Ilustración 9.4.12: Ejemplo WSDL (continuación).....	275
Ilustración 9.4.13: Estructura UDDI.....	284
Ilustración 9.4.14: Archivo .jar.....	285
Ilustración 9.4.15: Archivo .war.....	285
Ilustración 9.4.16: JAXB;Operaciones Unmarshal y Marshal.....	286
Ilustración 9.4.17: JAXB; Operaciones Unmarshal y Marshal II.....	287
Ilustración 9.4.18: API JAXB.....	287
Ilustración 9.5.1: Formato cabecera Bluetooth.....	291
Ilustración 9.5.2: Piconet.....	292
Ilustración 9.5.3: Colaboracion entre la implementación y la aplicación servidora para el registro del servicio.....	300

# Índice de tablas

Tabla 1: Estimación temporal iteración 1.....	74
Tabla 2: Estimación temporal iteración 2.....	78
Tabla 3: Estimación temporal iteración 4.....	86
Tabla 4: Plantilla descripción Casos de Uso.....	225
Tabla 5: Descripción caso de uso: Crear Noticia.....	226
Tabla 6: Descripción caso de uso: Modificar Noticia.....	228
Tabla 7: Descripción caso de uso: Eliminar Noticia.....	229
Tabla 8: Descripción caso de uso: Obtener Noticias.....	230
Tabla 9: Descripción caso de uso: Obtener Categorías.....	231
Tabla 10: Descripción caso de uso: Dar de alta dispositivo móvil.....	233
Tabla 11: Descripción caso de uso: Dar de baja dispositivo móvil.....	235
Tabla 12: Descripción caso de uso: Modificar dispositivo móvil.....	236
Tabla 13: Descripción caso de uso: Asociar categoría a dispositivo móvil.....	237
Tabla 14: Descripción caso de uso: Desasociar categoría de dispositivo móvil.....	238
Tabla 15: Descripción caso de uso: Obtener dispositivos móviles.....	239
Tabla 16: Descripción caso de uso: Mostrar noticias pantallas.....	240
Tabla 17: Descripción caso de uso: Inicializar Pila.....	241
Tabla 18: Descripción caso de uso: Escanear Dispositivos.....	242
Tabla 19: Descripción caso de uso: Escanear Servicios.....	243
Tabla 20: Descripción caso de uso: Realizar Pairing.....	245
Tabla 21: Descripción caso de uso: Generar Noticias Dispositivo.....	245
Tabla 22: Descripción caso de uso: Enviar Noticias.....	247
Tabla 23: Diccionario datos Análisis.....	256



# 1 ESTRUCTURA DE LA MEMORIA

---

En el presente apartado se explicará con detalle cómo está estructurado el documento que recoge todo el trabajo realizado en este proyecto. Para ello se definirá el significado de cada uno de los capítulos de los que consta esta memoria del proyecto, así como toda la estructura del material adicional que se adjunta.

## 1.1 *Estructura del documento*

En el segundo capítulo se muestra una pequeña introducción sobre las necesidades que hay en la actualidad para el campo de la comunicación o publicación de noticias y que llevó a plantear la elaboración del sistema propuesto en el presente documento para paliarlas. Para ello se analizaron los beneficios que supondría la implantación del sistema, en este caso especial, la Universidad de Granada o en cualquier otra empresa u organismo, y que actualmente no dispone de ellos. Para no cometer los errores existentes y tratar de imitar los aciertos en algunos de los sistemas software que implementan soluciones similares a la que se propone, se realiza el análisis de algunos programas comerciales que hay actualmente en el mercado. Con este estudio de mercado podremos ver qué tecnologías son las más utilizadas, los estándares más comunes y cuáles de ellos se adecúan mejor a las necesidades del proyecto. Una vez analizado el mercado y las tecnologías existentes, se expone brevemente la finalidad que tendrá el proyecto y la solución que se ha adoptado para poder solventar exitosamente el problema que estamos tratando. En este mismo capítulo se trata también todo el tema legal y de licencias utilizadas, así como la licencia bajo la cual será publicado el proyecto, ya que se ha optado por liberar el código del mismo y que esté disponible para la comunidad.

En el tercer capítulo se realiza un análisis detallado sobre todas las posibilidades que han sido propuestas para la resolución del proyecto, así como una comparación ventajas/inconvenientes entre las mismas. A continuación del estudio realizado se expone de forma detallada la solución elegida para el proyecto, tanto la arquitectura software elegida del sistema (tecnologías utilizadas, librerías, protocolos, etc.), como toda la arquitectura hardware en la que se implantará.

En el cuarto capítulo de esta memoria se indica la metodología y proceso de desarrollo del software que ha sido utilizado para el desarrollo del proyecto. Se lista todo el software utilizado, tanto las librerías utilizadas en el código como las aplicaciones y herramientas en las que se ha apoyado el desarrollo del proyecto. También se indica el hardware que se ha utilizado durante todo el proceso de desarrollo, así como el necesario para poder implantar el sistema de

## 1.1 .Estructura del documento

---

una forma práctica. Posteriormente se definen todos los estándares y recomendaciones que se han seguido, ya sean estándares definidos por los organismos o simplemente reglas o formas de trabajo para lograr una homogeneidad en el código y la documentación por parte de los distintos participantes en el proyecto.

El quinto capítulo recoge toda la documentación técnica de ingeniería del software que se ha realizado en el proyecto. Se parte de una planificación inicial del proyecto, tanto temporal como de recursos, estableciendo los distintos objetivos a cumplir y que posteriormente serán comparados con los datos obtenidos a la finalización del proyecto. En este tema se reúne un resumen de los diagramas, tablas, estudios y otros documentos realizados durante las distintas etapas de planificación, modelado, análisis y diseño del proyecto.

El sexto capítulo explica los todos los detalles que se han tenido en cuenta o que son dignos de mencionar durante la proceso de implementación del sistema, así como los problemas que se encontraron durante el desarrollo del proyecto.

En el séptimo capítulo se indica todos los procedimientos paso a paso para la instalación, configuración y mantenimiento de todo el sistema por parte de los administradores de éste. También está destinado a los usuarios que utilizarán la aplicación y que no necesitan ningún conocimiento previo.

En el octavo capítulo se exponen todas las conclusiones que han sido recopiladas tras la experiencia obtenida durante el desarrollo del proyecto y que podrán ser útiles para futuros proyectos. Se valoran posibles mejoras futuras en el sistema desarrollado.

El noveno capítulo recoge todos los anexos que se considera que podrían ser útiles para los lectores de la memoria y que no se han incluido en el cuerpo principal, ya sea por su longitud o porque son excesivamente técnicos y específicos.

Finalmente se muestran las referencias, tanto físicas como digitales, utilizadas para el desarrollo del proyecto.

---

## 1.2 .Estructura de los elementos adjuntos al proyecto

---

### ***1.2 Estructura de los elementos adjuntos al proyecto***

La gran mayoría de la documentación del sistema está recogida en el presente libro. El código fuente, junto con el resto de recursos utilizados para la realización del proyecto, están recopilados en el CD que se encuentra adjunto al presente libro. La totalidad de la documentación, herramientas utilizadas, fuentes de información, código fuente, ejecutables e histórico de cambios están reunidos en el siguiente repositorio:

<http://code.google.com/p/blufeedme/>

También se dispone de un blog propio para la aplicación en <http://blufeedme.wordpress.com/>, donde se publicarán noticias sobre el proyecto, opiniones de invitados, etc.

## 2 INTRODUCCIÓN

El sistema BLUFEEDME para la publicación y divulgación de noticias a través de diferentes medios, Web y Bluetooth, desarrollado en este proyecto, nace con la idea de dar solución a la actual dispersión de información en el ámbito de la Universidad de Granada.

Los Sistemas de Información (SI) y las Tecnologías de Información (TI) han cambiado la forma en que se difundía la información. En la actualidad está claro que Internet es la mayor fuente de información existente y además la más accesible, por lo que los organismos y empresas estudian la forma de mejorar la divulgación de sus mensajes. El principal problema que encuentran hoy en día es la saturación de información que poseen los usuarios, por lo que la mejor forma de llamar su atención es ofrecérsela de forma personalizada, clara y a través de un mayor número de canales. De este modo el usuario captará con mayor interés la información que le sea más accesible, concisa, cómoda y atractiva visualmente.

La obligación de obtener información de un elevado número de fuentes hace el día a día del usuario más laborioso. En la actualidad, un usuario perteneciente a una universidad puede visitar diariamente alrededor de seis Webs diferentes.

Un ejemplo práctico, un estudiante de la UGR tiene que visitar diariamente la plataforma SWAD para buscar las nuevas notificaciones pertenecientes a las asignaturas que tiene registradas. A continuación visita alrededor de dos portales más para el resto de asignaturas ([decsai.ugr.es](http://decsai.ugr.es) , [electronica.ugr.es](http://electronica.ugr.es), [aulabd.ugr.es](http://aulabd.ugr.es), [tutor.ugr.es](http://tutor.ugr.es), [oficinavirtual.ugr.es](http://oficinavirtual.ugr.es), etc). Es probable que esté registrado en una o dos webs para asignaturas independientes que no están incluidas en ningún portal. Si a todo lo anterior le añadimos que el estudiante debe leer el correo, visitar la web de la universidad y la propia de su facultad para conocer las notificaciones de secretaría, cursos y ofertas culturales, perderá un tiempo valioso para poder estar al día de todo lo relacionado con su carrera. Este mismo caso práctico podría aplicarse a profesores de la universidad y personal adjunto.

Un posible sistema que resolvería el problema de la dispersión de información es la creación de una nueva plataforma que recoja en el caso de los estudiantes, por ejemplo, todas las asignaturas. Sin embargo ese nuevo sistema no dejaría de ser “otro más” en toda la red de información de la Universidad. A los usuarios que están habituados a un sistema no les sería fácil adaptarse de nuevo a otra plataforma más, con todos los cambios que ello conlleva. Otro motivo para rechazar la idea es la gran dificultad que supondría unificar toda la información que proviene de los sistemas existentes, donde cada uno de ellos posee una estructura y probablemente tecnología diferente. Es por ello que la

## 2 .Introducción

---

solución no pasaría por la creación de un nuevo sistema estático que los englobe a todos, sino en crear un sistema que sea compatible o transparente a todos ellos y que permita recopilar su información para cederla posteriormente a los usuarios u otros sistemas de la Web, esto es en definitiva, la metodología o el pensamiento de la Web 2.0.

Por todo lo anteriormente mencionado, es indudable que se necesitan herramientas que unifiquen la información ofrecida por las diferentes plataformas.

Hasta ahora era todo un reto conseguir que los distintos sistemas de información fueran “transparentes” entre sí tanto en el hardware como en el software, de tal manera que los usuarios de los servicios no estén “enterados” de la estructura sobre el que funcionan. En la actualidad está en auge los conceptos como cloud computing, web services [5]<sup>1</sup>, redes sociales o computación online que permiten que tanto los usuarios como otros sistemas accedan a distintos servicios de una forma estándar, transparente y accesible a todo el mundo a través de la web.

El sistema BLUFEEDME desarrollado en el presente documento tratará de obtener una solución válida en base a las ideas de integrabilidad, accesibilidad a la información y cercanía al usuario.

En lo referente a lo personal, el proyecto BLUFEEDME se hace cercano al usuario mediante el uso de tecnología Bluetooth. Hoy día un dispositivo móvil es un complemento diario más y en la mayoría de los casos necesario. BLUFEEDME aprovecha este elemento para ofrecer una información personalizada, filtrando la información, y acentuando la sensación en el usuario de que es una pieza más en todo el entramado de la organización del organismo.

Además, el hecho de que la información llegue al usuario de forma directa, sin que éste tenga que ir a buscarla, y que al almacenarla de forma local en su dispositivo móvil disponga de ella en cualquier momento, hace aún más atractivo al proyecto BLUFEEDME. Aportando un mayor radio de publicación de noticias, permitiendo que el usuario se convierta en un medio más de difusión de las mismas.

### ***2.1 Motivación del proyecto***

La elección del desarrollo de un proyecto cuyo tema central es la publicación de noticias a través de diferentes canales, surge tras valorar con los tutores diferentes alternativas sobre temas que pudiesen ser de interés para una entidad, en este caso la Escuela Técnica de Ingenierías Informática y Telecomunicación de la UGR, desde un punto de vista práctico y, porque no, quizás comercial de cara a otras entidades o empresas y que, además permitieran aplicar conocimientos y destrezas adquiridas durante los años

---

<sup>1</sup> Ref. Bibliografía: Web 2.0:Arquitectura Orientada a Servicios en Java.

## 2.1 .Motivación del proyecto

---

cursados de Ingeniería Informática.

Se trató de escoger un tema que además del uso de tecnologías actuales permitiese el uso de dispositivos periféricos. Respecto a esta cuestión el problema de publicación de noticias, tal y como se enfoca en el presente proyecto, emplea una tecnología muy extendida en la actualidad, tecnología Web 2.0, y además contempla un uso interesante de periféricos, incorporando pantallas y dispositivos Bluetooth; antenas Bluetooth y dispositivos móviles externos al sistema. La posibilidad de introducir estos dispositivos y tecnología hace que el problema adquiera un mayor atractivo.

Es por los motivos mencionados por los que se decide adoptar el problema de publicación e integración de noticias como tema central del Proyecto Informático detallado en el presente documento.

## 2.2 *Análisis de mercado*

Antes de realizar cualquier proyecto informático es muy recomendable comprobar si hay algún otro software que cumpla con los requerimientos solicitados. Esto puede ayudar a diseñar un software que siga los estándares de los programas que hay ya comercializados y además visualizar sus carencias e intentar imitar sus aciertos.

En lo que respecta a publicación de noticias los sistemas que actualmente pueden encontrarse en el mercado son sistemas web independientes y que no ofrecen posibilidad de integración en otros sistemas: *Fusion News* (<http://www.duamu.com/re/script/1227/id/5862/scripts-fusion-news.html>), *phpNewsManager* (<http://www.duamu.com/re/script/1227/id/5045/scripts-phpnewsmanager.html>), y multitud de sistemas con las mismas características y propósitos. Este tipo de sistemas no satisfacen una de las principales ideas sobre las que se sustenta la motivación para desarrollar el sistema propuesto en el presente documento, integrabilidad.

Por tanto, ese tipo de sistemas no son aptos para dar solución al problema planteado para la publicación y difusión de noticias.

En lo que respecta a la difusión de las noticias, en el mercado actual no existe ningún software que se acomode perfectamente a los requisitos deseados, sin embargo, hay multitud de programas que permiten el envío de ficheros e información a dispositivos móviles mediante Bluetooth. Hay pocos sistemas que estén relacionados con la educación porque la mayoría de estos programas están relacionados con el mundo del marketing y la publicidad. Estos programas permiten la divulgación de publicidad en entornos abiertos como pueden ser centros comerciales, estadios, bancos, monumentos etc.

Algunas de las empresas que realizan dicho software son:

- BluelInfo <http://www.blueinfo.net>
- Blue-Tooth <http://www.blue-tooth.es/>

## 2.2 .Análisis de mercado

---

- Bluemessenger <http://www.bluemessenger.es/>
- Blue-shot <http://www.blue-shot.com/>
- Bluehertz <http://www.bluehertz.es>

Se ha revisado las características de algunos de los programas anteriores y se ha visto como todos se basan en el lenguaje Java para sus aplicaciones. Esto es muy comprensible porque Java es un lenguaje que proporciona una gran portabilidad y que está muy extendido actualmente en la telefonía móvil, por lo que la aplicación resultante será compatible con la mayoría de los dispositivos móviles existentes en el mercado. Otra ventaja que posee java es que hay gran variedad de librerías para dispositivos móviles y que facilitan el desarrollo de las aplicaciones, además varios sistemas operativos para móviles ofrecen herramientas de desarrollo que están escritas en el lenguaje Java, como por ejemplo Android o Symbian (también está disponible para C++).

Algunos de los programas analizados se basan en una arquitectura de tipo cliente-servidor mientras que la gran mayoría está basada únicamente en el envío de ficheros ordinarios.

### **2.2.1 Proyectos basados en cliente-servidor**

En este tipo de proyectos es necesaria una aplicación cliente ubicada en el dispositivo móvil o portátil, mediante la cual se comunica con otra aplicación servidor que estará conectada a la antena o antenas que harán de emisores. Estas aplicaciones están orientadas a la comunicación en ambos sentidos, por lo que se utilizan en las aplicaciones que realizan una interacción directa con el usuario propietario del dispositivo móvil. Estos sistemas suelen ser más complejos y proporcionan un mayor número de posibilidades, por lo que permiten crear herramientas potentes.

Aunque poseen un gran inconveniente que es la desconfianza de los usuarios, ya que son más reacios a la hora de tener que instalar una aplicación en su móvil y que no probablemente no controlen sus operaciones. Además la aplicación cliente normalmente se ofrece de forma directa a través de Bluetooth utilizando envío de ficheros mediante protocolos como OBEX. Esto provoca que el usuario la mayoría de las veces rechace la recepción de la aplicación si no está lo suficientemente informado o tiene confianza en la empresa emisora.

Estos sistemas suelen trabajar sobre capas de más bajo nivel en la pila Bluetooth como son RFCOMM o incluso L2Cap, que están pensados principalmente para la transferencia de Bytes entre dos dispositivos.

### **2.2.2 Proyectos basados en transferencia de ficheros**

La gran mayoría de sistemas se basan únicamente en el envío unidireccional de información por parte del servidor Bluetooth hacia los distintos dispositivos clientes que se encuentran al alcance, por lo tanto, únicamente

## 2.2 .Análisis de mercado

---

necesitan enviar datos a un dispositivo destino sin la necesidad de recibir una respuesta o petición.

Por lo general todas las aplicaciones realizan envío de ficheros ordinarios (texto, video, imágenes, aplicaciones, html, vcards etc) mediante protocolos de más alto nivel en la pila Bluetooth como por ejemplo OBEX, pensado fundamentalmente para el intercambio de objetos y ficheros.

El protocolo OBEX permite a las aplicaciones realizar el envío de ficheros de una forma más sencilla que mediante RFCOMM, ya que se trata de un protocolo de más alto nivel orientado a la comunicación y que nos proporciona operaciones sencillas para la transferencia de ficheros.

## 2.3 *Objetivos del proyecto*

En base a lo comentado hasta el momento podemos decir que uno de los principales objetivos de este proyecto es el de implementar un sistema integrable en otros sistemas externos y que lleve a cabo la publicación de las noticias procedentes de dichos sistemas de forma conjunta, siguiendo para ello con la metodología y pensamiento de la Web 2.0; interoperatividad, aplicaciones web que facilitan el compartir información, un diseño centrado en el usuario y la colaboración en la World Wide Web son conceptos a los que comúnmente está asociado el término Web 2.0.

Para poder cumplir con este objetivo el sistema que se va a implementar dispondrá de un subsistema que permita que los usuarios lleven a cabo la gestión de la información desde otros sistemas externos. Este subsistema consistirá en un Web Service, facilitando la integración y uso en los sistemas externos desde cualquier plataforma, cualquier momento y cualquier lugar del mundo, y agrupar noticias procedentes de los múltiples sistemas externos.

Para un uso y funcionamiento adecuado, nuestro sistema debe contemplar mecanismos de control de usuarios, quién y a qué puede acceder cada usuario registrado en el sistema. El subsistema de gestión de la información debe ser quien implemente estos mecanismos, en nuestro caso el subsistema Web Service. Usando técnicas criptográficas para implementar métodos de autenticación basados en firma del mensaje.

Otro objetivo del proyecto es conseguir un sistema accesible. En el caso de los usuarios gestores el Web Service nos aportaría una elevada accesibilidad. Para los usuarios destinatarios de la información, se debe de tratar de conseguir un sistema que ponga a disposición del usuario múltiples medios para acceder a la información. Medios que sean lo más cercanos posibles al usuario y que acceder a ellos no suponga una tarea extra en el día a día.

Para conseguirlo en la publicación de las noticias se usarán dos mecanismos: vía Web, mediante pantallas colocadas por el edificio y vía Bluetooth; consiguiendo informar al usuario durante la actividad diaria dentro del edificio de la E.T.S.I.I.T y permitiendo almacenar la información en su dispositivo

### 2.3 .Objetivos del proyecto

móvil, facilitando el acceso en cualquier momento y a través de un elemento en la vida diaria de la sociedad actual.

Las noticias serían publicadas a través de pantallas colocadas en diferentes puntos y enviadas por Bluetooth a los dispositivos móviles registrados en el sistema. Con estos dos mecanismos conseguiríamos dotar al sistema de la cercanía al usuario y la accesibilidad indicada.

Las pantallas serían usadas para la publicación de noticias de carácter más general y las noticias personales serían enviadas al usuario mediante Bluetooth.

## ***2.4 Licencia y liberación de código***

Antes de adentrarnos en la licencia empleada y otros aspectos del proyecto aclaremos los conceptos principales que pueden encontrarse alrededor de la liberación de código.

El software libre (en inglés free software, aunque esta denominación también se confunde a veces con "gratis" por la ambigüedad del término en el idioma inglés) es la denominación del software que respeta la libertad de los usuarios sobre su producto adquirido y, por tanto, una vez obtenido puede ser usado, copiado, estudiado, modificado y redistribuido libremente. Según la Free Software Foundation, el software libre se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, modificar el software y distribuirlo modificado.

El software libre suele estar disponible gratuitamente, o al precio de costo de la distribución a través de otros medios; sin embargo no es obligatorio que sea así, por lo tanto no hay que asociar software libre a "software gratuito" (denominado usualmente freeware), ya que, conservando su carácter de libre, puede ser distribuido comercialmente ("software comercial"). Análogamente, el "software gratis" o "gratuito" incluye en ocasiones el código fuente; no obstante, este tipo de software no es libre en el mismo sentido que el software libre, a menos que se garanticen los derechos de modificación y redistribución de dichas versiones modificadas del programa.

Por tanto el «software libre» es una cuestión de libertad, no de precio. Para entender el concepto, debería pensar en «libre» como en «libre expresión», no como en «barra libre».

Tampoco debe confundirse software libre con "software de dominio público". Éste último es aquel software que no requiere de licencia, pues sus derechos de explotación son para toda la humanidad, porque pertenece a todos por igual. Cualquiera puede hacer uso de él, siempre con fines legales y consignando su autoría original. Este software sería aquel cuyo autor lo dona a la humanidad o cuyos derechos de autor han expirado, tras un plazo contado desde la muerte de este, habitualmente 70 años. Si un autor condiciona su uso bajo una licencia, por muy débil que sea, ya no es del dominio público.

## 2.4 .Licencia y liberación de código

---

Cuando hacemos referencia en la definición de software libre a libertad de los usuarios de ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software, significa que los usuarios de programas tienen las cuatro libertades esenciales.

- La libertad de ejecutar el programa, para cualquier propósito (libertad 0).
- La libertad de estudiar cómo trabaja el programa, y cambiarlo para que haga lo que usted quiera (libertad 1). El acceso al código fuente es una condición necesaria para ello.
- La libertad de redistribuir copias para que pueda ayudar al próximo (libertad 2).
- La libertad de distribuir copias de sus versiones modificadas a terceros (la 3<sup>a</sup> libertad). Si lo hace, puede dar a toda la comunidad una oportunidad de beneficiarse de sus cambios. El acceso al código fuente es una condición necesaria para ello.

Un programa es software libre si los usuarios tienen todas esas libertades.

Una vez situados en el ámbito, se puede decir que en el presente proyecto se desarrolla software libre. La licencia empleada es una de las más utilizadas, Licencia Pública General de GNU (GNU GPL) en su versión 3 [16]<sup>2</sup>.

Con esta licencia los autores conservamos los derechos de autor (copyright), y permite la redistribución y modificación bajo términos diseñados para asegurarse de que todas las versiones modificadas del software permanecen bajo los términos más restrictivos de la propia GNU GPL. Esto hace que sea imposible crear un producto con partes no licenciadas GPL: el conjunto tiene que ser GPL.

Es decir, la licencia GNU GPL posibilita la modificación y redistribución del software, pero únicamente bajo esa misma licencia. Por tanto, todas y cada una de las partes que componen el software desarrollado en el presente proyecto se encuentran licenciadas con GPL o con licencias compatibles.

La única parte del sistema que emplea una licencia diferente a GPL es la librería Bluecove [14]<sup>3</sup>, usada para implementar el subsistema de envío de noticias mediante Bluetooth. La versión utilizada, de dicha librería, cuenta con una licencia Apache v2.0.

La licencia Apache (Apache License o Apache Software License para versiones anteriores a 2.0) es una licencia de software libre creada por la Apache Software Foundation (ASF) compatible con GPL. La licencia (con versiones 1.0, 1.1 y 2.0) requiere la conservación del aviso de copyright y el disclaimer, pero no es una licencia copyleft, ya que no requiere la redistribución del código fuente cuando se distribuyen versiones modificadas.

Como cualquier otra de las licencias de software libre, la Licencia Apache

---

2 Ref. Bibliografía: Licenses GNU

3 Ref. Bibliografía: Bluecove

## 2.4 .Licencia y liberación de código

---

permite al usuario del software la libertad de usarlo para cualquier propósito, distribuirlo, modificarlo, y distribuir versiones modificadas de ese software.

Además de cumplir con lo necesario para liberar el software con licencia GPL (inclusión en cada uno de los archivos fuentes el texto con la licencia), se ha seguido durante el proceso de desarrollo los pasos necesarios para llevar a cabo, de la mejor manera posible, la liberación de código.

Todo el software empleado para el desarrollo del proyecto es software gratuito, abaratando los costes de desarrollo.

El software a sido desarrollado utilizando un repositorio de código abierto en GoogleCode. Para acceder tanto al código de los archivos fuentes del proyecto como a toda la documentación y algunas de las aplicaciones empleadas, basta con acceder a la dirección "<http://blufeedme.googlecode.com/svn/trunk/>".

Además de todo lo indicado, se ha creado un blog en Wordpress, <http://blufeedme.wordpress.com/> . De esta forma mostramos el software a toda la comunidad. A través del blog se podrán seguir los progresos y posibles cuestiones que surjan durante la vida del software desarrollado.

## 2.5 *Glosario de Términos*

**Webservice:** Forma estandarizada de integrar aplicaciones WEB mediante el uso de XML, SOAP, WSDL y UDDI [9]<sup>4</sup> sobre los protocolos de la Internet. XML es usado para describir los datos, SOAP se ocupa para la transferencia de los datos, WSDL se emplea para describir los servicios disponibles y UDDI se ocupa para conocer cuáles son los servicios disponibles. Este conjunto de protocolos y estándares permite intercambiar datos sin la necesidad de conocer los detalles de sus receptivos Sistemas de Información y realizado a través de aplicaciones software desarrolladas en diferentes lenguajes de programación y ejecutadas sobre cualquier plataforma a través de Internet.

**XML:** Abreviación de Extensible Markup Language. El XML es una especificación desarrollada por W3C. Permite a los desarrolladores crear sus propios tags o etiquetas que les permiten habilitar definiciones, validaciones, e interpretación de los datos entre aplicaciones y entre organizaciones.

**SOAP:** Abreviación de Simple Object Access Protocol , es un protocolo de mensajería construido en XML que se usa para codificar información de los requerimientos de los Web Services y para responder los mensajes antes de enviarlos por la red. Los mensajes SOAP son independientes de los sistemas operativos y pueden ser transportados por los protocolos que funcionan en la

---

4 Ref. Bibliografia: <http://www.w3.org/TR/wsdl>

## 2.5 .Glosario de Términos

---

Internet, como pueden ser: SMTP, MIME y HTTP.

**WSDL:** Abreviación de Web Services Description Language, es un lenguaje especificado en XML que se ocupa para definir los Web Service como colecciones de punto de comunicación capaces de intercambiar mensajes. El WSDL es parte integral de UDDI y parte del registro global de XML, en otras palabras es un estándar de uso público (no se requiere pagar licencias ni royalties para usarlo).

**UDDI:** Abreviación de Universal Description, Discovery and Integration. Es un directorio distribuido que opera en la Web y que permite a las empresas publicar sus Web Services, para que otras empresas los conozcan y utilicen. Funciona de manera análoga a las páginas amarillas pero en este caso de servicios.

**SHA-1:** Es una función de hash criptográfica diseñada por la Agencia de Seguridad Nacional y publicado por el NIST como un Estándar Federal de EE.UU. de procesamiento de información. SHA significa Secure Hash Algorithm.

**OBEX:** Abreviatura de OBject EXchange, intercambio de datos; también denominado IrOBEX) es un protocolo de comunicaciones que facilita el intercambio de objetos binarios entre dispositivos. Es mantenido por la Infrared Data Association (IrDA) pero ha sido adoptada también por el Bluetooth Special Interest Group y por la sección SyncML de la Open Mobile Alliance (OMA).

**Bluecove:** Bluecove es una implementación del estándar JSR-82 en J2SE que actualmente posee interfaces para MacOSX, WIDCOMM, BluSoleil y Microsoft stack, que son distintos tipos de pilas Bluetooth. Funciona bajo la Java Virtual Machine y está distribuida con las licencias Apache Software licence V 2.0 y la GPL V2.1.

**Hot-spot:** Un hotspot (punto caliente en inglés) es una zona de cobertura inalámbrica, en el que uno o varios puntos de acceso proporcionan servicios de red a través de un dispositivo físico o antena. Los hotspots se encuentran en lugares públicos, como aeropuertos, bibliotecas, centros de convenciones, facultades, etcétera. Este servicio puede brindarse de manera gratuita o pagando una suma que depende del proveedor.

**Pairing:** Procedimiento previo a una conexión bluetooth, por el cual dos dispositivos se identifican mutuamente mediante el intercambio de una clave conocida por ambos como medida de seguridad.

## 3 SOLUCIÓN PARA EL PROYECTO

---

La solución propuesta para resolver el problema de difusión de noticias planteado se basa en tres conceptos fundamentales: integración, accesibilidad e interacción personal con el usuario.

### 3.1 *Sistema Integrable*

Durante los inicios del desarrollo de las aplicaciones uno de los problemas en los que más empeño fue empleado para tratar de solucionarlo, fue tratar de que los sistemas sean independientes de la plataforma de hardware y sistema operativo, lo que de alguna forma se logra con la aparición de lenguajes de programación que permiten realizar esa diferencia, como por ejemplo Java.

Luego, el ambiente WEB abrió las puertas para generar aplicaciones muchos más independientes y accesibles desde cualquier parte del mundo, pero aun así existía un problema que se debía solucionar que era la heterogeneidad de los diferentes sistemas, los cuales estaban escritos en lenguajes distintos por lo que su interoperabilidad se hacía compleja. Es en este punto donde nace una solución que se empapa de diversas tecnologías que funcionan, y que de alguna forma solucionan este problema de comunicación entre aplicaciones que se encuentran en ambientes distribuidos (La WEB) y que se desarrollan absolutamente de forma distinta. Esta solución es conocida como Web Services, también en muchos textos se realiza una relación directa con SOA (Arquitectura Orientada a Servicios), esto ya que los Web Services son considerados los servicios bases para su funcionamiento, dejando claro que se podrían utilizar otros.

Desde el punto de vista más simple, se podría mencionar que los Web Services son Servicios Web, entendiendo por servicios “programas que pueden ser accedidos por la red” (En este caso la red sería la WEB). Desde el punto de vista de un desarrollador, un Web Service es un componente independiente que posee un conjunto de funcionalidades que pueden ser accedidas desde cualquier lugar y plataforma. Desde cualquier lugar, quiere decir que estarán disponibles dichos servicios a través de un medio común de comunicación (la WEB). Desde cualquier plataforma, quiere decir que los datos que se reciben y son enviados por los Web Services son independientes de la plataforma de origen o destino, esto se logra utilizando para la presentación de los datos el Lenguaje Extendido de Marcas (XML). Para obtener más información acerca del funcionamiento y estándares empleados por un Web Service se puede consultar el Anexo IV: Web Service.

En base a lo comentado se puede decir que los objetivos fundamentales de un Web Service son:

### 3.1 .Sistema Integrable

---

- Independencia del lenguaje y de la plataforma
  - Separación de especificación de la implementación
- Interoperabilidad
  - Utilización de estándares: XML, SOAP, WSDL, UDDI...
- Acoplamiento débil: Sistemas basados en mensajes
  - Interacciones síncronas y asíncronas
- A través de Internet
  - Sin control centralizado
  - Utilización de Protocolos establecidos
- Modularidad y Reusabilidad de servicios
  - Escalabilidad: Uno-a-uno frente a uno-a-muchos

El sistema desarrollado ofrece un Web Service, desarrollado en JAVA, a los usuarios registrados para llevar a cabo la gestión de toda la información que este dentro de su ámbito de acceso.

En la solución propuesta se trabaja con cuatro elementos significativos (clases conceptuales): noticias, categorías, gestores y dispositivos. Para el correcto funcionamiento y tratamiento de la información contenida en nuestro sistema se realiza un control de acceso de usuarios. El Web Service permite un acceso controlado de usuarios mediante el uso de técnicas de cifrado, concretamente mediante autentificación con firma del mensaje obtenida mediante el algoritmo de encriptación SHA-1. Todas y cada una de las operaciones requieren que el usuario firme el mensaje para poder llevar a cabo la operación.

La forma de estructurar la información permitiendo un acceso controlado a la misma es la agrupación de las noticias en categorías. Cada una de las categorías existentes está controlada solo y exclusivamente por un gestor. De esta forma un gestor solo tendrá privilegios sobre la información relacionada con las categorías que gestione. Todas las operaciones necesarias para la gestión adecuada de dicha información están disponibles a través del Web Service. Las operaciones relacionadas con la gestión de categorías y noticias son:

- ✓ *addNoticia*: mediante esta operación el gestor podrá crear noticias asociadas a una de las categorías gestionadas por él.
- ✓ *deleteNoticia*: mediante esta operación el gestor podrá eliminar una de las noticias creadas en el sistema, perteneciente, claro está, a una de las categorías gestionadas por él.

### 3.1 .Sistema Integrable

---

- ✓ *updateNoticia*: mediante esta operación el gestor podrá modificar la información de una de las noticias creadas por él.
- ✓ *getNoticias*: mediante esta operación el gestor puede obtener todas las noticias pertenecientes a las categorías gestionadas por él. Permite filtrar por categoría las noticias a obtener.
- ✓ *GetCategorias*: mediante esta operación el gestor puede acceder a la información referente a cada una de las categorías que gestiona.

Para una mayor seguridad y un mayor control el sistema no permite que se registren usuarios por sí mismos, es decir, solo el administrador del sistema puede registrar usuarios gestores. Igual ocurre con las categorías y la asignación de gestores a categorías. De esta forma evitamos que usuarios externos a la entidad, en nuestro caso la E.T.S.I.I.T, puedan registrarse en el sistema y publicar noticias ajenas al fin del sistema, y que los usuarios puedan auto asignarse la gestión de categorías de forma incontrolada.

El sistema consta de una parte de publicación de noticias mediante el envío vía Bluetooth a dispositivos móviles, que será detallada más adelante. El Web Service ofrece, además de las funciones antes mencionadas, operaciones para la gestión de dispositivos móviles. Al igual que con los gestores, la agrupación en categorías de las noticias permite que cada dispositivo reciba solamente las noticias adecuadas mediante la asociación de éstos a las categorías deseadas. En este caso el Web Service proporciona a los gestores la capacidad de poder registrar dispositivos móviles en el sistema. Todos y cada uno de los dispositivos registrados en nuestro sistema recibirán las noticias publicadas mediante el subsistema de publicación de noticias por pantallas, pero además, existe la posibilidad de asociar los dispositivos a las categorías deseadas. Las operaciones disponibles en el Web Service relacionadas con la gestión de dispositivos móviles son:

- ✓ *addDispositivoMovil*: esta operación permite al usuario registrar un dispositivo móvil en el sistema.
- ✓ *DeleteDispositivoMovil*: mediante esta operación el usuario podrá dar de baja un dispositivo móvil en el sistema. Para ello el dispositivo no podrá estar asociado a ninguna categoría.
- ✓ *updateDispositivoMovil*: mediante esta operación el usuario podrá modificar la información relativa a un dispositivo registrado en nuestro sistema. Solo podrá llevar a cabo la operación si el dispositivo móvil está asociado a una de las categorías que gestiona el gestor.
- ✓ *asociar*: esta operación permite al usuario asociar un dispositivo móvil a una de las categorías gestionadas por él.

### 3.1 .Sistema Integrable

---

- ✓ *desasociar*: mediante esta operación el gestor puede desasociar un dispositivo móvil de una categoría. La categoría a desasociar debe de ser una categoría gestionada por él.
- ✓ *getDispositivos*: permite al gestor obtener información de todos los dispositivos existentes en el sistema.
- ✓ *getMyDispositivos*: mediante esta operación el gestor puede consultar información de los dispositivos que están asociados a alguna de las categorías gestionadas por él.
- ✓ *getDispositivosWhere*: permite al usuario obtener los dispositivos móviles de forma filtrada asociados a una de las categorías gestionadas por él.

Toda la información del sistema es almacenada en una base de datos centralizada. De esta forma todos y cada uno de los subsistemas que componen nuestro sistema accederán a información actualizada, siendo la base de datos la encargada de gestionar la concurrencia en el acceso a los datos.

Dotando de un Web Service a la solución para el problema de publicación de noticias conseguimos satisfacer la necesidad actual de obtener sistemas de fácil integración.

## 3.2 *Accesibilidad*

Para cualquier sistema de publicación de información es imprescindible conseguir poner a disposición del usuario destinatario medios suficientes para acceder a dicha información. La solución propuesta en el presente documento ofrece la posibilidad de emplear medios próximos al usuario y mediante elementos que forman parte del día a día del destinatario.

Como un primer medio de publicación se utilizan pantallas distribuidas por el edificio de la E.T.S.I.I.T. Con ese medio de publicación el usuario puede obtener la información de forma rápida y en el transcurso de la actividad desarrollada en el edificio de la escuela. La tarea de adquirir la información no supondría una tarea extra en la interacción diaria entre la entidad, Escuela Técnica Superior de Ingenierías Informática y de Telecomunicaciones, y el usuario. Usar este tipo de medio de publicación evita al usuario la necesidad de disponer de algún recurso en el momento que desee acceder a la información: PC, PDA, acceso a internet, etc.

La publicación a través de las pantallas consiste en una web dinámica, constituida en su totalidad por un carrusel. El carrusel mostrará de forma consecutiva todas y cada una de las noticias de interés general. Para ello se dispondrá de una categoría asociada a la publicación por este medio, de forma que, todas las noticias pertenecientes a dicha categoría serán consideradas de

### 3.2 .Accesibilidad

---

carácter general y publicadas por las pantallas.

La web se implementa usando algunas de las tecnologías que permiten dotar de dinamismo la web: PHP [6]<sup>5</sup> y JavaScript. Se emplean librerías de ambas tecnologías que facilitan la implementación de la web: PHPLib de PHP y JQuery [10]<sup>6</sup> en JavaScript. Con el uso de las citadas librerías se consigue que la web se actualice de forma dinámica, utilizando AJAX, y muestre información totalmente actualizada accediendo a la información existente en la base de datos centralizada.

Como se ha comentado para dotar al sistema de mayor accesibilidad es necesaria la utilización de múltiples medios de publicación de la información. La solución propuesta ofrece un segundo medio para la publicación de la información, noticias en este caso. Este segundo medio consiste en el envío a través de Bluetooth de las noticias, esta vez de carácter personal, a los dispositivos móviles de los usuarios. Este sistema de publicación permite que la información llegue al usuario sin necesidad de que este tenga que ir a buscarla, utilizando un elemento cotidiano y personal. La publicación mediante este medio da la posibilidad al usuario de almacenar su información, accediendo a ella en cualquier momento y en cualquier lugar, y porque no, divulgar dicha información; convirtiendo al usuario en un elemento más del sistema de publicación de noticias.

El subsistema encargado de publicar noticias vía Bluetooth consiste en una aplicación, implementada en JAVA, que mediante el uso de antenas Bluetooth, detectará todo aquel dispositivo móvil que, en disposición de tecnología Bluetooth, entre dentro del rango de cobertura de dichas antenas. Esta aplicación controlará la información a enviar y a qué usuarios mediante el acceso a la base de datos centralizada, en la cual existirá un historial y la información necesaria para llevar a cabo dicha tarea.

Como se indicó en el apartado anterior, el sistema permite el registro de dispositivos móviles. Esta tarea se lleva a cabo a través del Web Service disponible.

Independientemente de si el dispositivo está o no registrado en nuestro sistema, una vez detectado le serán enviadas, si acepta el envío claro está, las noticias de carácter general que en el momento del envío están siendo publicadas por las pantallas colocadas en el edificio. De esta forma permitimos que mediante este medio se obtenga toda la información disponible, sin necesidad de acceder a las pantallas colocadas por la facultad.

En el caso de dispositivos registrados asociados a alguna de las categorías existentes, además de recibir la información mencionada anteriormente, recibirán todas y cada una de las noticias pertenecientes a las categorías a las que se encuentran asociados. Con esta medida personalizamos la información que el usuario recibe.

---

5 Ref. Bibliografía: Programación de Servicios Web interactivos con PHP y MySQL.

6 Ref. Bibliografía: <http://jquery.com/>

### ***3.3 Interacción cercana al usuario***

Lograr un sistema de publicación de noticias que tenga una interacción cercana con el usuario ha sido otra de las ideas base para obtener la solución propuesta para el presente proyecto. Con la integración de pantallas como medio de publicación se consigue transmitir la información durante la actividad diaria desarrollada en el centro. Pero es con la publicación mediante Bluetooth cuando se consigue una mayor proximidad al usuario. Actualmente para cualquier persona, o sino para la gran mayoría, el dispositivo móvil es un complemento personal y laboral más. Permitir que el usuario acceda a la información, noticias en este caso, a través de su dispositivo móvil, acentúa el hecho de que forma parte de la estructura de la entidad, la E.T.S.I.I.T. Resulta muy útil para cualquier usuario poder almacenar la información de forma local, y acceder y hacer un uso personal de dicha información. Además, el poder almacenar la información en los dispositivos móviles, convierte a estos en un medio más para extender la divulgación de las noticias, al poder transmitirse esta información entre diferentes usuarios.

Es por ese motivo por el que la solución propuesta ofrece los medios de divulgación o publicación de noticias mencionados.

### ***3.4 Escenario***

A grandes rasgos, el escenario en el que se desarrollará nuestro proyecto consistirá en la publicación de noticias en una o varias facultades de la Universidad de Granada. Para simplificar el problema se tomará como ejemplo únicamente la Escuela Técnica Superior de Ingenierías Informática y Telecomunicación.

En la ETSIIT se imparten, desde el curso 2010-2011, el Grado en Ingeniería Informática y el Grado en Ingeniería de Tecnologías de Telecomunicación, aunque aún se imparte docencia de los antiguos títulos de Informática y Telecomunicación hasta su extinción definitiva de forma gradual.

Se trata de una entidad que cuenta con múltiples departamentos que imparten docencia en las titulaciones antes mencionadas. En la actualidad todos ellos disponen de plataformas propias para la gestión de su información y por tanto, para la publicación de información asociada al departamento, [decsai.ugr.es](http://decsai.ugr.es), [tstc.ugr.es](http://tstc.ugr.es), [lsi.ugr.es](http://lsi.ugr.es), [atc.ugr.es](http://atc.ugr.es), etc. Algunos departamentos emplean las plataformas propias para la gestión de sus asignaturas, másteres, proyectos de fin de carrera, etc. En otros casos, se dispone de plataformas para la docencia, comunes a toda la comunidad universitaria y usadas por diferentes departamentos para la gestión y el desarrollo de algunas de las asignaturas impartidas. Es el caso de [swad.ugr.es](http://swad.ugr.es).

Además de las citadas plataformas la escuela cuenta con una web, [etsiit.ugr.es](http://etsiit.ugr.es), para la gestión y publicación de toda la información de carácter general referente a la escuela.

### 3.4 .Escenario

---

En todas las plataformas web mencionadas se produce publicación de noticias a las que el usuario debe acceder de forma independiente para obtener toda la información necesaria en su día a día.

En cuanto a la arquitectura del edificio, éste cuenta con dos módulos, un módulo para biblioteca, secretaría, comedores, cafetería y despachos, y otro módulo para aulas de docencia, teóricas y prácticas. Ambos módulos cuentan con 6 plantas.

## ***3.5 Arquitectura Física o Hardware***

Tras la especificación del escenario anterior se puede realizar una implantación de la infraestructura desarrollada para el caso concreto de la ETSIIT. La aplicación utiliza en mayor parte una arquitectura de tipo cliente/servidor, dado que la mayoría de los servicios que se ofrecen son a través de la Web. En este tipo de arquitecturas, una aplicación se modela como un conjunto de servicios que son proporcionados por la máquina servidor para que los distintos clientes hagan uso de ellos. De esta forma, los distintos programas cliente únicamente necesitan conocer la existencia del servidor para poder enviarle las peticiones que deseen. Una vez recibidas dichas peticiones por parte del servidor, éste las llevará a cabo y enviará la respuesta correspondiente de vuelta al cliente. A causa de dicha arquitectura, la aplicación necesita cuatro tipos de servidores distintos, servidor de base de datos, servidor web, servidor para los Web Services y servidores Bluetooth, que podrán estar o no en un mismo servidor físico, aunque como veremos más adelante se ha optado por separar físicamente cada uno de ellos.

A continuación pasamos a comentar el diseño de toda la infraestructura necesaria por el sistema Blufeedme. En primer lugar realizaremos una enumeración de todos los dispositivos físicos que formarán parte de dicha infraestructura:

- Servidores físicos:
  1. se necesita un servidor que contendrá al sistema de gestión de base de datos y en el cuál se almacenará la base de datos de nuestro software (sería conveniente que estuviera optimizado para ello y que tuviera un sistema óptimo de copias de seguridad).
  2. Es necesaria otra máquina para contener nuestro servidor de aplicaciones que será el encargado de recibir y contestar a las peticiones mediante Web Services.
  3. También se necesita un servidor web en el que estará alojada la página web que visualizarán todas las pantallas, así como la aplicación web de gestión que será un cliente de nuestros propios Web Services.
  4. Pcs de sobremesa o estaciones de trabajo por cada uno de los Hot Spot Bluetooth. Los Hot Spot Bluetooth estarán conectados a estas

### 3.5 .Arquitectura Física o Hardware

máquinas y tendrán instalada tanto la plataforma Java como la aplicación Bluetooth.

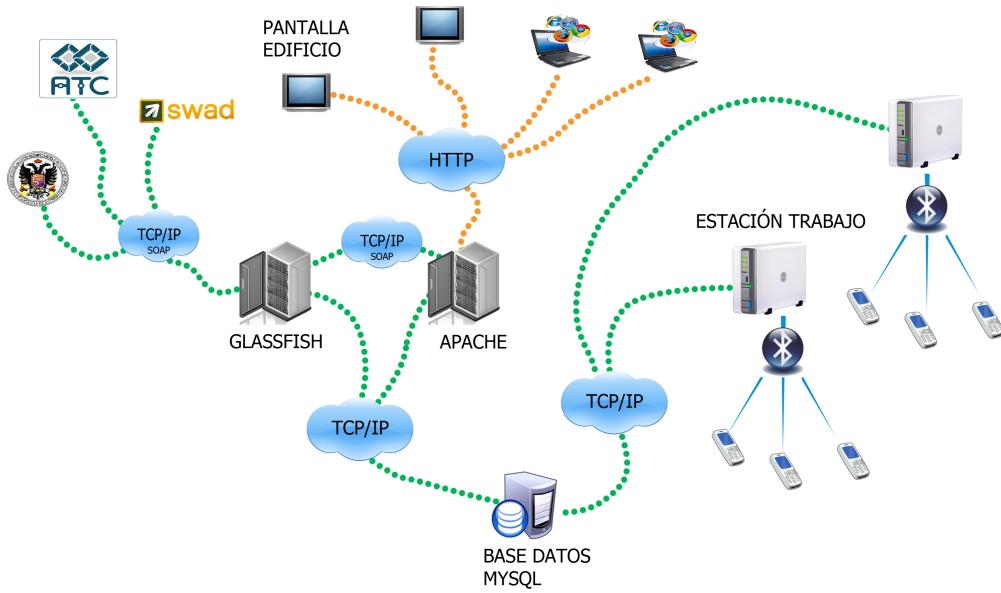
Se ha decidido separar el servidor web del servidor de aplicaciones para aumentar la fiabilidad de nuestro sistema. Si se hace de esta manera, la aplicación encargada de visualizar las noticias en las pantallas será independiente del servidor que contendrá nuestros WebServices, en caso de que en alguno de los dos se caiga el servicio.

- Hot Spot Bluetooth o antena Bluetooth: Dispositivo físico capaz de emitir y recibir señales Bluetooth a grandes distancias (200 metros) frente a los dispositivos Bluetooth convencionales. Dicho dispositivo deberá poseer una pila Bluetooth compatible con la librería Bluecove y que disponga del servicio OBEX PUSH.
- Repetidor de señal Bluetooth: Un repetidor Bluetooth es un dispositivo electrónico que recibe una señal débil o de bajo nivel y la retransmite con una mayor potencia o nivel más alto, de esta forma podemos cubrir mayores distancias disminuyendo la degradación de la señal Bluetooth.
- Televisión o monitor: Se necesita una serie de televisores que estarán colocados en distintos sitios estratégicos y que tendrán la función de mostrar las noticias más importantes en tiempo real. Para que esto sea posible podría implementarse de dos formas distintas:
  1. Que los televisores posean un navegador integrado, de este modo solamente tendrían que estar conectados a internet mediante un cable de red o por wifi.
  2. Que los televisores estén conectados mediante un cable de video a la salida de video a uno o varios Pcs, que dispondrán de conexión a internet y tendrán un navegador web instalado.

Una vez que se han listado todos los dispositivos que formarán el sistema es necesario establecer la ubicación de cada uno de ellos y su interconexión. Para ello mostramos un esquema genérico donde mostramos toda la información:

### 3.5 .Arquitectura Física o Hardware

---



*Ilustración 3.5.1: Arquitectura Hardware*

Como se aprecia en el esquema anterior, se deben establecer varios servidores Bluetooth para ofrecer una mayor cobertura a los estudiantes a lo largo de toda la escuela y además reducir el tiempo de servicio, ya que cada servidor Bluetooth posee un número máximo de conexiones simultáneas. Para reducir el coste de la infraestructura se ha optado por reducir el número de servidores Bluetooth y sustituirlos por repetidores de señal que proporcionarán una mayor cobertura y tienen un coste más reducido.

Con la infraestructura comentada anteriormente se puede garantizar que cualquier persona que esté en el interior del centro esté informada de todas las noticias que sean publicadas al instante. Del mismo modo, también se logra que cualquier editor que desee publicar una noticia pueda realizar dicha operación desde cualquier lugar, con el simple hecho de tener una conexión a internet. Además mediante este sistema se proporciona una futura integración con las distintas plataformas ya existentes, ya que únicamente tendrán que implementar una serie de peticiones mediante Web Services contra el servidor para la publicación de sus propios contenidos a través del sistema.

Un ejemplo práctico de utilización del sistema en la escuela ocurriría de la siguiente forma:

Una persona que está encargada de escribir las noticias de una o varias categorías se dispone a publicar una noticia para que todo el centro tenga conocimiento de ella. Lo primero que hace es acceder a mediante un navegador

### 3.5 .Arquitectura Física o Hardware

a la interfaz web de la aplicación Blufeedme. Desde dicha interfaz introducirá la noticia a través de los distintos formularios e inmediatamente ésta quedará almacenada en el sistema. En ese mismo momento cualquier persona que se encuentre en el centro y esté al alcance de cobertura Bluetooth, recibirá en su dispositivo móvil la información que acaba de ser publicada mediante la recepción de un fichero con todas las noticias de las que no tiene conocimiento. Además, si se trata de una noticia de carácter general, ésta será publicada en todos los monitores que están ubicados en la escuela, por lo que también podrá ser leída por aquellas personas que no dispongan de un dispositivo móvil o simplemente tengan el Bluetooth desactivado.

Ese sería el ejemplo en el que una persona publica una noticia en la escuela, sin embargo, también podría darse el caso de que cualquier plataforma que hay actualmente en la escuela realice las mismas operaciones pero de forma automática. Supongamos que SWAD ha implementado las funciones necesarias para poder enviar peticiones mediante Web Services al sistema. En este caso, el ejemplo sería distinto porque la persona que introduce noticias en SWAD no necesita tener conocimiento ni acceso a la plataforma para publicar las noticias a través del sistema, ya que sería la propia plataforma SWAD la que enviaría las peticiones mediante los servicios web que se oferta en el sistema para toda la gestión de las noticias.

## **3.6 Arquitectura Software**

Una vez enumerada toda la infraestructura física de que se dispone para el sistema, es necesario definir que software estará instalado en cada uno de los dispositivos con los que se cuenta.

A continuación se realiza un listado con todo el software necesario y la ubicación en la que debe estar instalado:

- *Servidor de Base de datos:* En dicho servidor estará instalado un sistema de gestión de base de datos, en concreto MySQL Server. Este SGBD ofrece la posibilidad de ser instalado sobre varios sistemas operativos y su elección será libre porque no influirá en el funcionamiento de la aplicación. Ésta máquina debe estar conectada a internet y debe tener las reglas adecuadas para autorizar el acceso del resto de servidores para las peticiones contra el SGBD. En esta máquina se ubicará la base de datos de la aplicación.
- *Servidor de aplicaciones:* En esta máquina se deberá instalar el servidor de aplicaciones, que en este caso se trata de Glassfish. GlassFish implementa las tecnologías definidas en la plataforma Java EE y tiene como base al servidor *Sun Java System Application Server (un derivado de Apache Tomcat)*. Como está basado en Java Platform Enterprise Edition (o Java EE) que es multiplataforma, puede ser instalado en diferentes sistemas operativos. La elección del sistema operativo para

### 3.6 .Arquitectura Software

este servidor tampoco afecta al funcionamiento de la aplicación, por lo que su elección es libre. En este servidor estarán todos los servicios Web que ofrece la aplicación.

- *Servidor Web:* Como comentamos en el apartado de Arquitectura Física o Hardware, se ha decidido separar físicamente el servidor de aplicaciones del servidor web para mejorar los servicios prestados en caso de fallo por parte de alguno de los dos servidores. El servidor de aplicaciones Glassfish nos ofrece la posibilidad de correr bajo él la aplicación web implementada en PHP, sin embargo, es un servidor que pese a su gran potencia es bastante pesado en cuanto a necesidad de recursos. Por este motivo, se ha aprovechado la separación física de ambos servidores para instalar un servidor más liviano para los propósitos del sistema como es Apache. Apache es un servidor web HTTP de código abierto multiplataforma (Unix, BSD, Linux, Microsoft Windows, Macintosh etc) que implementa el protocolo HTTP/1.12 y el concepto de sitio virtual. Por tanto, la elección del sistema operativo para el servidor Web quedará en manos del usuario.

En este servidor se almacenará tanto la aplicación cliente de servicios Web que se encarga de ofrecer una interfaz gráfica para dichas operaciones, como la página web que muestra todas las noticias más importantes y que serán mostradas en los distintos televisores instalados.

- *Servidor Bluetooth:* En los distintos servidores Bluetooth se instalará cualquier sistema operativo que se desee con la única restricción de que sea compatible con la JVM (Java Virtual Machine). Sobre él se instalará la J2SE 6 Update21 junto con la aplicación Bluetooth que está realizada en Java. Para que la aplicación funcione se deben copiar las librerías correspondientes de Bluecove y las de conexión con la base de datos JDBC (para más detalles ver apartado de 7.1.1 , Instalación/Configuración).

#### **3.6.1 Modelo de Arquitectura Software**

Cada uno los subsistemas que forman parte de la aplicación Blufeedme está basado en la arquitectura Cliente-servidor y por tanto sigue el mismo patrón que la mayoría de aplicaciones orientadas a la Web.

Cada subsistema de la aplicación lo podríamos dividir en tres capas lógicas diferenciables:

- *Capa de presentación:* es la que ve el usuario (también se la denomina "capa de usuario"), presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). También es conocida como interfaz gráfica y debe tener la característica de ser amigable (entendible y fácil de usar) para el usuario. Esta capa se

### 3.6 .Arquitectura Software

comunica únicamente con la capa de negocio.

- *Capa de negocio:* es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse y que han sido establecidas. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y a su vez con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él.
- *Capa de datos:* es donde residen los datos y es la encargada de acceder a los mismos. Está formada por un gestor de bases de datos que realiza todo el almacenamiento de datos y recibe las solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

Como se ha comentado, los subsistemas observados de forma independiente, interactúan con el usuario siguiendo una arquitectura cliente-servidor y construidos siguiendo una arquitectura en capas, sin embargo, todos los subsistemas de Blufeedme se relacionan entre sí para ofrecer una visión de conjunto. Esto se consigue mediante una arquitectura de repositorio en la que cada uno de los subsistemas se toma como independiente y todos los datos se ubican en una base de datos central a la que acceden todos. Esto nos ofrece la ventaja de tener un mayor desacoplamiento entre los distintos subsistemas, de tal forma que cada uno de ellos puede ser modificado por separado.

También se ha tratado de seguir en los distintos subsistemas una arquitectura modelo-vista-controlador (MVC) para desacoplar la lógica de aplicación en cada uno de los subsistemas de su visualización. Por ejemplo, para la aplicación Bluetooth se han separado el modelo de datos, la aplicación que lleva a cabo todo el proceso y la interfaz que únicamente se encarga de recoger los datos y mostrarlos. En el caso del subsistema de gestión Web Service y del subsistema de publicación de noticias a través de pantallas.

En resumen, se podría decir que los subsistemas siguen una arquitectura cliente-servidor para responder a las distintas peticiones que les son solicitadas (Bluetooth, Web Services, http). Interaccionan entre todos ellos a través de la base de datos, siguiendo por tanto una arquitectura de repositorio y en cada uno de ellos se separan el modelo, la vista que se ofrece y el controlador.

#### **3.6.2 Lenguajes de programación**

##### **3.6.2.1 PHP**

Para dotar de dinamismo el contenido de las diferentes plataformas Web implementadas en la solución ofrecida para el problema planteado en el presente proyecto, se emplea PHP.

### 3.6 .Arquitectura Software

---

PHP es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. Es usado principalmente para la interpretación del lado del servidor (server-side scripting) pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica usando las bibliotecas Qt o GTK+.

En nuestro caso se ha utilizado para la interpretación del lado del servidor. Fundamentalmente se ha empleado en el acceso a Bases de datos en los casos necesarios (Control de sesiones en el cliente del servicio web y acceso a la información de las noticias en el caso de publicación a través de pantallas) y para crear un cliente de nuestro servicio web.

#### **A) Software libre**

Publicado bajo la PHP License, la Free Software Foundation considera esta licencia como software libre.

Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno. El lenguaje PHP se encuentra instalado en más de 20 millones de sitios web y en un millón de servidores, el número de sitios en PHP ha compartido algo de su preponderante sitio con otros nuevos lenguajes no tan poderosos desde agosto de 2005.

#### **B) Similitud con lenguajes estructurados**

El gran parecido que posee PHP con los lenguajes más comunes de programación estructurada, como C y Perl, permiten a la mayoría de los programadores crear aplicaciones complejas con una curva de aprendizaje muy corta. También les permite involucrarse con aplicaciones de contenido dinámico sin tener que aprender todo un nuevo grupo de funciones.

#### **C) Otras ventajas**

- Es un lenguaje multiplataforma.
- Orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos.
- El código fuente escrito en PHP es invisible al navegador web y al cliente ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador. Esto hace que la programación en PHP sea segura y confiable.
- Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y PostgreSQL.
- Capacidad de expandir su potencial utilizando módulos (llamados ext's o extensiones).

### 3.6 .Arquitectura Software

---

- Posee una amplia documentación en su sitio web oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite aplicar técnicas de programación orientada a objetos.
- Biblioteca nativa de funciones sumamente amplia e incluida.
- No requiere definición de tipos de variables aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución.
- Tiene manejo de excepciones (desde PHP5).
- Si bien PHP no obliga a quien lo usa a seguir una determinada metodología a la hora de programar (muchos otros lenguajes tampoco lo hacen), aun haciéndolo, el programador puede aplicar en su trabajo cualquier técnica de programación o de desarrollo que le permita escribir código ordenado, estructurado y manejable. Un ejemplo de esto son los desarrollos que en PHP se han hecho del patrón de diseño Modelo Vista Controlador (MVC), que permiten separar el tratamiento y acceso a los datos, la lógica de control y la interfaz de usuario en tres componentes independientes.

#### 3.6.2.2 JAVA

Analizando las aplicaciones del mercado actual, se ha comprobado que la gran mayoría está realizada en el lenguaje de programación Java. Los componentes Web Service y publicación mediante Bluetooth se implementan empleando el lenguaje Java.

##### A) Lenguaje simple

Java posee una curva de aprendizaje muy rápida. Resulta relativamente sencillo escribir aplicaciones interesantes desde el principio. Como estamos familiarizados con C++, nos encontramos que aprender Java es mucho más sencillo, ya que se han eliminado ciertas características, como los punteros. Otra ventaja es que se pueden migrar muy rápidamente fragmentos de código escritos previamente desde C++ a Java y ser productivos en poco tiempo.

##### B) Orientado a objetos

Java fue diseñado como un lenguaje orientado a objetos desde el principio. Los objetos agrupan en estructuras encapsuladas tanto sus datos como los métodos (o funciones) que manipulan esos datos. La tendencia del futuro, a la que Java se suma, apunta hacia la programación orientada a objetos, especialmente en entornos cada vez más complejos y basados en red.

**C) Distribuido**

Java proporciona una colección de clases para su uso en aplicaciones de red, que permiten abrir sockets y establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas.

**D) Interpretado y compilado a la vez**

Java es compilado, en la medida en que su código fuente se transforma en una especie de código máquina, los bytecodes, semejantes a las instrucciones de ensamblador.

Por otra parte, es interpretado, ya que los bytecodes se pueden ejecutar directamente sobre cualquier máquina a la cual se hayan portado el intérprete y el sistema de ejecución en tiempo real (run-time).

**E) Robusto**

Java fue diseñado para crear software altamente fiable. Para ello proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. Sus características de memoria liberan a los programadores de una familia entera de errores (la aritmética de punteros), ya que se ha prescindido por completo los punteros, y la recolección de basura elimina la necesidad de liberación explícita de memoria.

**F) Seguro**

Dada la naturaleza distribuida de Java, donde las applets se bajan desde cualquier punto de la Red, la seguridad se impuso como una necesidad de vital importancia. A nadie le gustaría ejecutar en su ordenador programas con acceso total a su sistema, procedentes de fuentes desconocidas. Así que se implementaron barreras de seguridad en el lenguaje y en el sistema de ejecución en tiempo real.

**G) Indiferente a la arquitectura**

Java está diseñado para soportar aplicaciones que serán ejecutadas en los más variados entornos de red, desde Unix a Windows Nt, pasando por Mac y estaciones de trabajo, sobre arquitecturas distintas y con sistemas operativos diversos. Para acomodar requisitos de ejecución tan variopintos, el compilador de Java genera bytecodes: un formato intermedio indiferente a la arquitectura diseñada para transportar el código eficientemente a múltiples plataformas hardware y software. El resto de problemas los soluciona el intérprete de Java.

**H) Portable**

La indiferencia a la arquitectura representa sólo una parte de su portabilidad. Además, Java especifica los tamaños de sus tipos de datos básicos y el comportamiento de sus operadores aritméticos, de manera que los

### 3.6 .Arquitectura Software

programas son iguales en todas las plataformas.

Estas dos últimas características se conocen como la Máquina Virtual Java (JVM).

#### **I) Multihebra**

Hoy en día ya se ven como terriblemente limitadas las aplicaciones que sólo pueden ejecutar una acción a la vez. Java soporta sincronización de múltiples hilos de ejecución (*multithreading*) a nivel de lenguaje, especialmente útiles en la creación de aplicaciones de red distribuidas. Así, mientras un hilo se encarga de la comunicación, otro puede interactuar con el usuario mientras otro presenta una animación en pantalla y otro realiza cálculos.

#### **J) Dinámico**

El lenguaje Java y su sistema de ejecución en tiempo real son dinámicos en la fase de enlazado. Las clases sólo se enlazan a medida que son necesitadas. Se pueden enlazar nuevos módulos de código bajo demanda, procedente de fuentes muy variadas, incluso desde la Red.

## 4 METODOLOGÍA DE DESARROLLO Y HERRAMIENTAS

El primer paso a llevar a cabo antes de adentrarse de lleno en el desarrollo de cualquier proyecto es necesario estudiar y determinar la metodología de desarrollo que se va a emplear y, consecuentemente, el conjunto de herramientas necesarias para cada una de las tareas que componen dicha metodología, y que nos llevará a la obtención del sistema final. La metodología nos permitirá alcanzar una solución de calidad. Como calidad software entenderemos la concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados, y con las características implícitas que se espera de todo software. La calidad depende del proceso de desarrollo seguido para la construcción del software por lo que es de vital importancia estudiar las posibles alternativas para comprobar sus características y determinar cuál de ellas es la que mejor se adapta a nuestro proyecto.

El problema que estamos abordando es un problema bien definido, basado en unos estándares existentes bien documentados. Además, no existe la figura de un cliente real como tal, por lo que aumenta la certeza de que el propósito del sistema no va a cambiar durante el proceso de desarrollo software. Estas características nos permiten determinar claramente el propósito del sistema desde un principio sin riesgos de grandes cambios a lo largo del desarrollo del producto. Por tanto, se necesita una metodología con un proceso de desarrollo que enfatice el uso de documentación generada frente al *cara a cara* con el cliente.

Dado lo anterior, se descarta una metodología ágil ya que enfatizan las comunicaciones cara a cara en vez de la documentación. Los métodos ágiles también enfatizan que el software funcional es la primera medida del progreso. El cliente interviene de forma activa durante el desarrollo produciéndose continuos cambios en los requisitos, para adaptar el producto a sus necesidades, a partir de prototipos (software funcional) que se entregan en cada una de las etapas en las que se divide el desarrollo. Combinado con la preferencia por las comunicaciones cara a cara, generalmente los métodos ágiles son criticados y tratados como "indisciplinados" por la falta de documentación técnica. Nos vemos pues, obligados a descartar metodologías ágiles como DSDM (Método de desarrollo de sistemas dinámicos), AUP (Agile Unified Process), XP (Programación Extrema) o Scrum.

Descartada la posibilidad de emplear una metodología ágil se hace necesaria la elección de una metodología tradicional o también conocidas como

## 4 .Metodología de desarrollo y herramientas

---

“pesadas”. Frente a la “indisciplina” de los métodos ágiles, los métodos “pesados” son muy estructurados y estrictos, extraídos directamente del modelo de desarrollo en cascada. Dentro de estas metodologías podemos encontrar, desarrollo en cascada, desarrollo basado en prototipos, incremental y proceso de desarrollo en espiral.

De todas las propuestas se opta por usar una metodología con un proceso de desarrollo incremental. Este proceso consiste en una serie de iteraciones (mini-proyecto) de corta duración fijada cuyo resultado es un sistema que puede ser probado, integrado y ejecutado [3]<sup>7</sup>.

Determinado el proceso de desarrollo, dado que se va a seguir una metodología de desarrollo orientada a objetos, se debe determinar un lenguaje de modelado. Como lenguaje de modelado se usará UML [4]<sup>8</sup>, ya que:

- Propone una metodología universal.
- Es un lenguaje que nos permite representar los modelos que se obtienen a partir de la aplicación de cualquier método OO.
- No fuerza a utilizar un modelo concreto de proceso del software.

### **4.1 Metodología General**

Como se ha indicado el proceso de desarrollo seleccionado es el desarrollo iterativo o incremental. Dado que es el proceso seleccionado definitivamente, vamos a dedicar esta sección a profundizar en los fundamentos de la ya citada metodología y determinar cómo adaptarla al desarrollo de este proyecto.

La principal idea del proceso incremental a la hora de afrontar el desarrollo de un proyecto de cierta envergadura es la división en iteraciones, de corta duración, siguiendo en cada una un proceso similar al seguido en el desarrollo en cascada. Cada una de estas iteraciones, mini-proyectos, darán lugar a un sistema o subsistema que podrá ser probado, ejecutado e integrado con lo obtenido en iteraciones anteriores. Cada una de estas iteraciones consta de las siguientes etapas:

---

7 Ref. Bibliografía: Ingeniería del Software II

8 Ref. Bibliografía: El lenguaje Unificado de modelado

#### 4.1 .Metodología General

---



*Ilustración 4.1.1: Proceso de desarrollo  
OO iterativo simplificado*

Donde cada una de las tareas o etapas consiste en:

- Modelado de requisitos: define el propósito del sistema. Dentro de esta tarea encontramos las siguientes sub-tareas:
  - Identificación de los requisitos funcionales del sistema.
  - Modelo funcional. Diagramas de casos de uso.
  - Requisitos no funcionales
  - Operaciones del sistema. Diagrama de secuencia del sistema.
  - Revisión
- Fase de análisis del proyecto: Se formaliza la especificación de requisitos obtenida en el modelado de requisitos en modelos que describen completamente el sistema. Dentro de esta tarea encontramos las sub-tareas:
  - Modelado estático: Identificación de clases conceptuales, atributos y relaciones: Diagrama de clases.
  - Modelado del comportamiento externo. Contratos. Diagramas de secuencia.
- Fase de diseño del proyecto: transformación del modelo de análisis a un modelo de diseño en el dominio de la solución seleccionando estrategias adecuadas para la construcción del sistema. Dentro de esta tarea encontramos las sub-tareas:
  - Diseño del sistema. Estilo arquitectónico.
  - Diseño de objetos. Diagrama de clases de diseño.
  - Storyboard de la aplicación.

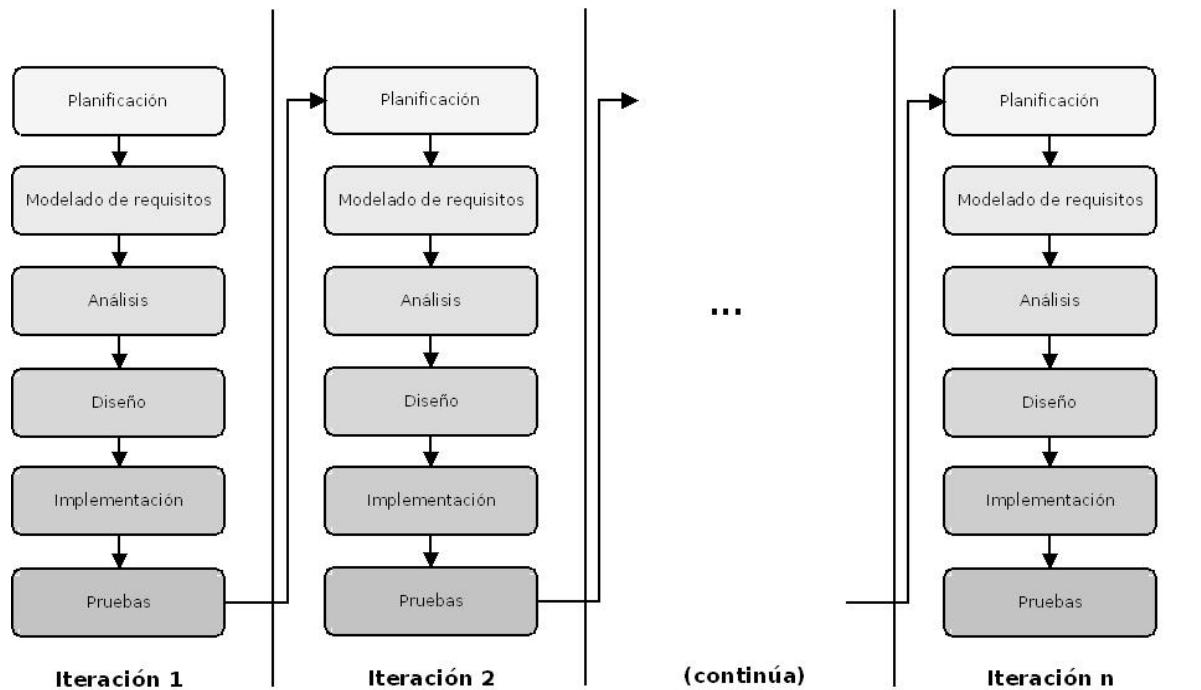
#### 4.1 .Metodología General

- Diseño de la BD.
- Diseño de la interfaz.
- Revisión.
- Implementación: transformación del modelo de diseño en código fuente. Se encontrarán sub-tareas como:
  - Implementación del modelo, arquitectura.
  - Implementación de la BD.
  - Implementación de interfaces...
- Pruebas y comprobación: en esta tarea se intentará encontrar diferencias entre el comportamiento esperado del sistema y el comportamiento observado en la ejecución del sistema implementado. Además, se debe revisar que los modelos de análisis se corresponden con la implementación. Formada básicamente por dos subtareas
  - Diseño de las pruebas.
  - Realización de las pruebas.

Cada iteración da soporte a la siguiente. Cuando se inicia una nueva iteración se utiliza el conocimiento adquirido en la anterior/es para el desarrollo de la misma:

#### 4.1 .Metodología General

---



*Ilustración 4.1.2: Desarrollo OO iterativo o incremental*

Como se indicó anteriormente cada iteración da lugar a un sistema o subsistema, de forma que el usuario final podría hacer uso del sistema en el estado en que se encuentre y, si así lo desea, modificar los requisitos que debe cumplir el sistema. Esta es una ventaja más del desarrollo iterativo, aunque podría suponer un problema si el usuario no estuviese dispuesto a invertir el tiempo que supondría estar involucrado en ese grado durante el desarrollo del proyecto.

Dado el problema a resolver en este proyecto, se trata de un problema bien definido, por lo que los requisitos alcanzados en cada iteración no cambiarán. Del mismo modo, al tener esta característica, puede que existan iteraciones donde no aparezcan todas las etapas mencionadas anteriormente debido a que ya se haya efectuado en el desarrollo de la iteración anterior.

La duración de cada una de las iteraciones debe de ser lo más ajustada posible a los objetivos que se van a desarrollar, ya que se trabaja de forma más eficiente si los objetivos a cumplir son a corto plazo. Se tratará de ajustar entre 3-5 semanas en función de los objetivos de la iteración. La planificación temporal de cada una de las iteraciones se puede consultar en el apartado 5.1, Planificación

## **4.2 *Determinación de recursos***

En esta sección vamos a citar y describir cada una de las herramientas usadas para llevar a cabo el desarrollo del presente proyecto. Las agruparemos en dos grandes grupos: software y hardware.

### **4.2.1 Herramientas Software**

En este apartado vamos a citar y describir las herramientas software usadas en las etapas de las iteraciones del proceso de desarrollo. Las agruparemos en función del tipo de herramienta software.

#### **4.2.1.1 *Lenguajes de programación.***

Para el desarrollo del subsistema Web Service y el subsistema Bluetooth se emplea como lenguaje de programación Java.

- **Java:** Java es un lenguaje de programación orientado a objetos, de propósito general e independiente de la plataforma en la que se ejecuta. Se trata de un lenguaje con un modelo de objetos más simple que otros lenguajes como C++, y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria. Para el desarrollo del proyecto se hace uso de la versión 6.

Sitio web oficial: [www.oracle.com/us/sun/index.html](http://www.oracle.com/us/sun/index.html)

Para el desarrollo del subsistema Web para mostrar noticias a través de las pantallas y el desarrollo del cliente del Web Service se hace uso del lenguaje de programación PHP5 y del lenguaje de scripting JavaScript.

- **PHP:** PHP es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. Es usado principalmente para la interpretación del lado del servidor (server-side scripting). Cuando el cliente hace una petición al servidor para que le envíe una página web, el servidor ejecuta el intérprete de PHP. Éste procesa el script solicitado que generará el contenido de manera dinámica (por ejemplo obteniendo información de una base de datos). El resultado es enviado por el intérprete al servidor, quien a su vez se lo envía al cliente. Para el desarrollo del proyecto se hace uso de la versión 5.1.3.

Sitio web oficial: [www.php.net](http://www.php.net)

- **JavaScript:** se trata de un lenguaje de programación interpretado. Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas, aunque existe una forma de JavaScript del lado del servidor (Server-side JavaScript o SSJS).

## 4.2 .Determinación de recursos

---

Manual online: [www.w3schools.com/js/default.asp](http://www.w3schools.com/js/default.asp)

### 4.2.1.2 IDEs (entornos de desarrollo)

Para el desarrollo de las partes del proyecto en las que se emplea el lenguaje de programación JAVA se emplea el entorno de desarrollo NetBeans. La utilización de dicho entorno se basa en la simplicidad a la hora de desarrollar un Web Service en dicho entorno: integración y comunicación con Glassfish, generación de WebServices a partir de WSDL y creación de operaciones del Web Service con asistente entre otras. Además, el usar dicho entorno no tendría apenas costes de uso en cuanto a aprendizaje, ya que ambos miembros del grupo disponen de conocimientos avanzados del IDE.

- **NetBeans:** NetBeans es un entorno de desarrollo, hecho principalmente para el lenguaje de programación Java aunque existe un número importante de módulos para extender el NetBeans IDE para su uso con otros lenguajes. NetBeans IDE es un producto libre y gratuito sin restricciones de uso. Se ha usado la versión 6.7.

Sitio web oficial: [netbeans.org](http://netbeans.org)

Durante el desarrollo del Web Service se ha empleado SOAPUI. Este entorno nos permite testear WebServices de forma fácil y rápida.

- **SOAPUI:** es un software libre de código abierto para testear WebServices. Permite comprobar la respuesta a cada una de las operaciones del Web Service de forma cómoda. Para el desarrollo del proyecto se usa la versión 1.0.1.

Sitio web oficial: [www.soapui.org](http://www.soapui.org)

- **Notepad++:** Notepad++ es un editor gratuito de código fuente para Microsoft Windows que soporta el desarrollo en varios numerosos lenguajes de programación, de entre los cuales PHP, JavaScript, XHTML y los archivos CSS, resultan especialmente interesantes para el desarrollo de este proyecto. Presenta un entorno de desarrollo altamente configurable y extensible mediante extensiones, de forma que cada usuario puede configurar el entorno de la forma que le resulte más cómoda para trabajar.

Sitio web oficial: [notepad-plus-plus.org](http://notepad-plus-plus.org)

### 4.2.1.3 Base de Datos

Como sistema de base de datos se utiliza para el proyecto MySQL. Para su administración se utiliza PHPMyAdmin. De ambos se posee conocimientos avanzados por lo que supone una ventaja decantarnos por dicha elección.

- **MySQL:** es un sistema de gestión de bases de datos relacional, multihilo y multiusuario. Existe una versión bajo la licencia GNU GPL. MySQL es

## 4.2 .Determinación de recursos

---

muy utilizado en aplicaciones web, como Drupal o phpBB, en plataformas (Linux/Windows-Apache-MySQL-PHP/Perl/Python). Su popularidad como aplicación web está muy ligada a PHP, que a menudo aparece en combinación con MySQL. MySQL es una base de datos muy rápida en la lectura cuando utiliza el motor no transaccional MyISAM, pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación. Existe la alternativa de usar InnoDB como motor de almacenamiento transaccional, con capacidades de commit (confirmación), rollback (cancelación) y recuperación de fallas. Además soporta restricciones de clave externa FOREING KEY. La versión usada para el desarrollo del proyecto a sido 5.5.8.

Sitio web oficial: [www.mysql.com](http://www.mysql.com)

- **PHPMyAdmin:** herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de páginas web, utilizando Internet. Actualmente puede crear y eliminar Bases de Datos, crear, eliminar y alterar tablas, borrar, editar y añadir campos, ejecutar cualquier sentencia SQL, administrar claves en campos, administrar privilegios y exportar datos en varios formatos. Se encuentra disponible bajo la licencia GPL.

Sitio web oficial: [www.phpmyadmin.net/home\\_page/index.php](http://www.phpmyadmin.net/home_page/index.php)

### 4.2.1.4 Librerías.

Para los subsistemas web se emplea la librería JQuery, una biblioteca de JavaScript, y PHPLib, librería del lenguaje de programación PHP.

- **JQuery:** se trata de una biblioteca o framework de JavaScript. Permite simplificar en gran medida la manera de interactuar con los documentos HTML, manipular el árbol DOM (Document Object Model; Modelo de objetos del documento), manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX (**A**synchronous **J**ava**S**cript **A**nd **X**ML) a páginas web, técnica que permite realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

Sitio web oficial: [jquery.com](http://jquery.com)

- **PHPLib:** librería que permite de forma sencilla poder trabajar con diferentes bases de datos de forma transparente, gestionar sesiones de usuarios y llevar un control de acceso basado en bases de datos.
- **Bluecove:** biblioteca Java para Bluetooth (implementación para la JSR-82) para las distintas interfaces de pilas Bluetooth en la actualidad como, Mac OS X, WIDCOMM, BlueSoleil, Microsoft Windows XP, Vista, 7 y WIDCOMM para Windows Mobile. Posee una gran portabilidad, ya que está implementada en Java y se puede integrar en cualquier programa que se base en la J2SE. También provee soporte para gran variedad de pilas Bluetooth existentes en la actualidad y al proveer una

## 4.2 .Determinación de recursos

---

implementación para el estándar JSR-82 funcionará con cualquier dispositivo que soporte dicho estándar. La utilización de esta librería no supondrá ningún coste adicional, únicamente se deben cumplir las cláusulas de su licencia dual [Apache License, Version 2.0](#) y [GNU General Public License V3](#).

Sitio web oficial:

<http://bluecove.org/>

<http://code.google.com/p/bluecove/>

### 4.2.1.5 Navegadores web

A medida que se desarrollan las partes web del sistema que se desarrolla en el presente proyecto es necesario ir comprobando su funcionamiento. Para esa tarea es necesario emplear un navegador web. Se puede utilizar cualquiera de los navegadores web modernos existentes actualmente, si bien se aconseja el uso del navegador Firefox con el complemento Firebug instalado para poder modificar, al vuelo, el contenido de la página mostrada en él.

- **Mozilla Firefox:** Mozilla Firefox es un navegador web libre y de código abierto y el segundo más usado en la actualidad. Es un navegador multiplataforma y como característica más destacada presenta la posibilidad de configurarlo a medida del usuario mediante la gran cantidad de complementos existentes para él. En la prueba Acid3 de estándares web, Firefox, en su versión actual, obtiene una puntuación de 97 sobre 100.

Sitio web oficial: [www.mozilla-europe.org/es](http://www.mozilla-europe.org/es)

### 4.2.1.6 Servidores

El presente proyecto consta de dos subsistemas, publicación de noticias vía web a través de las pantallas y el cliente del Web Service, ambos residentes en la Web, por lo que se hace necesaria la elección de un servidor web. Este servidor será el que se encargue de procesar las peticiones HTTP enviadas por los clientes, a través de su navegador web, y responder a dichas peticiones con el código HTML de la pagina web; el cliente, una vez recibido el código, lo interpreta y lo exhibe en pantalla.

- **Servidor HTTP Apache:** El servidor HTTP Apache es un servidor HTTP de código libre desarrollado para los sistemas operativos modernos por la Fundación de Software Apache (The Apache Software Foundation). Apache es el servidor web más utilizado en el mundo desde el año 1996. Para el presente proyecto se hace uso del servidor Apache en su versión 2.2.17

Sitio web oficial: [www.apache.org](http://www.apache.org)

## 4.2 .Determinación de recursos

---

Además del citado servidor web el presente proyecto consta de un Web Service. Para poder ejecutar el Web Service es necesario disponer de un servidor de aplicaciones. Como servidor de aplicaciones vamos a emplear Glassfish en el desarrollo del presente proyecto. Al igual que con el resto de herramientas ambos componentes del grupo poseen conocimientos, por lo que no supone un coste adicional seleccionar este servidor, sino que será una ventaja.

- **Servidor de aplicaciones Glassfish:** Se trata de un servidor de aplicaciones desarrollado por Sun Microsystems para la plataforma Java EE. Como características se puede destacar la funcionalidad como servidor web, servidor de JEE 5 y de JEE6 en su versión 3.2, servidor de WebServices, Contenedor de EJB, JBI, balanceo de carga y colas de espera y una infinidad de funciones más. Este servidor permite subdividir las aplicaciones que están residentes en dominios; agrupaciones de aplicaciones que pueden compartir librerías, logs y configuraciones. Se trata de un “gran” servidor de aplicaciones web, por lo que es un alto consumidor de CPU y la administración es complicada. Para el desarrollo del presente proyecto se usa la versión 3.1.

Sitio web oficial: [glassfish.java.net](http://glassfish.java.net)

### 4.2.1.7 Software Planificación, Modelado, Análisis y Diseño

Durante cada una de las etapas de las iteraciones se emplean programas para llevar a cabo la planificación, modelado, análisis y diseño.

En todas las etapas se hace uso de software que nos permita aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo en nuestro caso. Este tipo de herramientas son conocidas como herramientas CASE(Computer Aided Software Engineering; Ingeniería de Software Asistida por Computadora). Estas herramientas nos pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras. En el presente proyecto se hace uso de la herramienta EnterpriseArchitect en su versión de evaluación. Se trata de una herramienta muy potente que puede ser usada durante todo el proceso de desarrollo software. Como complemento para la realización de diagramas de menor envergadura o de una complejidad menor se ha empleado el software DIA.

- **EnterpriseArchitect:** Enterprise Architect 8 es una plataforma de diseño, visualización y modelado de alto rendimiento basada en el estándar de UML 2.3. Con trazabilidad completa desde mapas conceptuales, a través de requisitos hasta el diseño y despliegue de software y negocios, Enterprise Architect 8 provee el tipo de visualización y colaboración robusta y eficiente requerida en los entornos de modelado grandes y demandantes de hoy en día. Para el desarrollo del presente proyecto se

## 4.2 .Determinación de recursos

---

ha empleado la versión 7.1.

Sitio web oficial: [www.sparxsystems.com.ar/products/index.html](http://www.sparxsystems.com.ar/products/index.html)

- **DIA:** DIA es una aplicación de código libre para el desarrollo de todo tipo de diagramas (de flujo de información, eléctricos, etc.), basado en gtk+e inspirada en el software comercial Visio de Microsoft. Los diagramas generados pueden ser exportados a una gran cantidad de formatos diferentes para su utilización.

Sitio web oficial: [live.gnome.org/Dia](http://live.gnome.org/Dia)

Para llevar a cabo la etapa de planificación de cada una de las iteraciones se hace necesario el uso de herramientas de planificación. La herramienta seleccionada para este proyecto es Gantt.

- **Gantt:** GanttProject es una herramienta de escritorio multiplataforma para la programación y gestión de proyectos. Se ejecuta bajo Windows, Linux y MacOSX, es libre y su código es opensource. Permite la realización de diagramas de Gantt de forma gráfica, la distribución de recursos humanos para trabajar en las tareas que se planifiquen y obtener el diagrama de PERT de forma automática a partir del diagrama de Gantt. Ofrece la posibilidad de guardar en una amplia variedad de formatos de imagen los diagramas elaborados.

Sitio web oficial: [www.ganttproject.biz](http://www.ganttproject.biz)

### 4.2.1.8 Control de versiones

Mediante la utilización de un Sistema de control de versiones se automatiza la tarea de creación y actualización de archivos en un servidor central de archivos creados o modificados en el área de trabajo local. Además permite recuperar los archivos situados en el servidor central bien porque se hayan perdido en el área de trabajo local o bien porque la copia almacenada en el servidor sea más reciente. Dado que el proyecto es realizado por un grupo formado por dos personas, es prácticamente imprescindible esta herramienta. Como sistema de control de versiones se hace uso de Subversion, mientras que como cliente suyo se utiliza TortoiseSVN.

- **Subversion:** Subversion es un sistema de control de versiones basado en software libre. Con él, de forma simplificada, se pueden llevar a cabo las tareas antes descritas permitiendo, además, volver a versiones anteriores de los archivos y comprobar las distintas modificaciones llevadas a cabo sobre ellos en el tiempo.
- **TortoiseSVN:** TortoiseSVN es un cliente para Subversion, para el Sistema Operativo Windows, que permite la interacción con dicho sistema de control de versiones de forma simplificada y desde el propio entorno de

## 4.2 .Determinación de recursos

---

Windows ya que se integra en él. También es software libre.

Sitio web oficial: [tortoisesvn.tigris.org](http://tortoisesvn.tigris.org)

### 4.2.1.9 Paquetes ofimáticos y documentación

Para redactar todos los documentos de cada una de las etapas del proceso de desarrollo software se emplea un procesador de textos, OpenOffice.

- **OpenOffice:** es una suite ofimática libre (código abierto y distribución gratuita) que incluye herramientas como procesador de textos, hoja de cálculo, presentaciones, herramientas para el dibujo vectorial y base de datos.<sup>4</sup> Está disponible para varias plataformas, tales como Microsoft Windows, GNU/Linux, BSD, Solaris y Mac OS X. Para redactar la memoria se emplea la versión 3.2.

Sitio web oficial: [www.openoffice.org](http://www.openoffice.org)

### 4.2.1.10 Sistemas Operativos

- **Linux:**

- *OpenSuse 11.3*
  - *Ubuntu 10.10 Maverick Meerkat*

- **Windows:**

- *Microsoft Windows XP (Service Pack 3).*
  - *Microsoft Windows 7*
  -

## 4.2.2 Herramientas Hardware

Las distintas herramientas hardware con las que se ha contado a lo largo del proceso de desarrollo del proyecto son las siguientes:

- Pcs Sobremesa:
  - AMD Athlon64 3700+, 2GBytes de RAM DDR, 1000 Gbytes HD en Raid 0. El equipo cuenta con un dispositivo Bluetooth Conceptronic nano.
  - Intel Pentium4 3.4 GHz 650 HT, 2 Gbytes de RAM DDR2, 560 Gbytes de disco duro. El equipo cuenta con un dispositivo Bluetooth Conceptronic nano.
- Portátiles:
  - Intel Centrino Duo a 1.83 GHz 2Mb Cache, 1Gb de memoria Ram, disco duro 80 Gbytes.

#### 4.2 .Determinación de recursos

---

- Intel Core 2 Duo a 2 GHz 2Mb Cache, 2 GB de Ram DDR2, disco duro 80 Gbytes 7200rpm.

Todos los equipos anteriores cuentan con todo el software necesario para el desarrollo del proyecto.

- Móviles:
  - Sony Ericsson Xperia X10 mini. Sistema operativo Android 2.1. Bluetooth v2.0
  - HUAWEI U7510. Bluetooth v2.0
  - Motorola EM25 . Bluetooth v2.0

#### 4.3 *Estándares seguidos*

- **Web Services Protocol Stack:** es una colección de protocolos y estándares para redes de Computadores que son utilizados para definir, localizar, implementar y hacer que un Servicio Web interactúe con otro. La Pila de Protocolos para servicios esta comprendida principalmente por cuatro áreas:
  - *Servicio de Transporte:* responsable del transporte de mensajes entre las Aplicaciones de red y los protocolos en los cuales se incluyen protocolos tales como HTTP, SMTP, FTP, así como también el más reciente Blocks Extensible Exchange Protocol (BEEP).
  - *Mensajería XML:* responsable por la codificación de mensajes en un formato común XML así que ellos puedan ser entendidos en cualquier extremo de una conexión de red. Actualmente, esta área incluye protocolos tales como XML-RPC, SOAP y REST.
  - *Descripción del Servicio:* usado para describir la interfaz pública de un Servicio Web específico. El formato de interfaz Web Services Description Language - WSDL es típicamente usado para este propósito.
  - *Descubrimiento de servicios:* centraliza servicios en un registro común tal que los servicios Web de la red puedan publicar su localización y descripción, y hace que sea fácil descubrir que servicios están disponibles en la red. Actualmente, la API Universal Description Discovery and Integration - UDDI se utiliza normalmente para el descubrimiento de servicios.
- **SOAP:** (siglas de Simple Object Access Protocol) es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. Este protocolo deriva de un protocolo creado por David Winer en 1998, llamado XML-

#### 4.3 .Estándares seguidos

---

RPC. SOAP fue creado por Microsoft, IBM y otros y está actualmente bajo el auspicio de la W3C. Es uno de los protocolos utilizados en los servicios Web. Para más detalles consultar Anexo IV: Web Service.

- **XML:** XML, siglas en inglés de eXtensible Markup Language ('lenguaje de marcas extensible'), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML). Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades[cita requerida]. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG, MathML.

XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.

XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

- **UDDI:** UDDI son las siglas del catálogo de negocios de Internet denominado Universal Description, Discovery and Integration. El registro en el catálogo se hace en XML. UDDI es una iniciativa industrial abierta (sufragada por la OASIS) entroncada en el contexto de los servicios Web. El registro de un negocio en UDDI tiene tres partes:

- Páginas blancas - dirección, contacto y otros identificadores conocidos.
- Páginas amarillas - categorización industrial basada en taxonomías.
- Páginas verdes - información técnica sobre los servicios que aportan las propias empresas.

UDDI es uno de los estándares básicos de los servicios Web cuyo objetivo es ser accedido por los mensajes SOAP y dar paso a documentos WSDL, en los que se describen los requisitos del protocolo y los formatos del mensaje solicitado para interactuar con los servicios Web del catálogo de registros.

Para más detalles consultar Anexo IV: Web Service.

- **HTML:** HTML, siglas de HyperText Markup Language (Lenguaje de Marcado de Hipertexto), es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con

#### 4.3 .Estándares seguidos

---

objetos tales como imágenes. HTML se escribe en forma de «etiquetas», rodeadas por corchetes angulares (<,>). HTML también puede describir, hasta un cierto punto, la apariencia de un documento, y puede incluir un script (por ejemplo Javascript), el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML.

- **XHTML:** XHTML, acrónimo en inglés de eXtensible Hypertext Markup Language (lenguaje extensible de marcado de hipertexto), es el lenguaje de marcado pensado para sustituir a HTML como estándar para las páginas web.

El estándar XHTML indica en un apéndice informativo una manera de escribir XHTML de modo tal que los navegadores actuales que sólo entienden HTML, lo procesen como si fuera éste. Para esto se deberá crear un documento con algunas restricciones y consideraciones, y servirlo con el «content-type» text/html, en vez del correcto para XHTML.

- **CSS3:** El nombre hojas de estilo viene del inglés Cascading Style Sheets, del que toma sus siglas. CSS es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). El W3C (World Wide Web Consortium) es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los agentes de usuario o navegadores.
- **UML:** Lenguaje Unificado de Modelado (LUM o UML, por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables.

Es importante resaltar que UML es un "lenguaje de modelado" para especificar o para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo.

Se puede aplicar en el desarrollo de software entregando gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado Racional o RUP), pero no especifica en sí mismo qué metodología o proceso usar.

UML no puede compararse con la programación estructurada, pues UML significa Lenguaje Unificado de Modelado, no es programación, solo se diagrama la realidad de una utilización en un requerimiento. Mientras que, programación estructurada, es una forma de programar como lo es la

#### 4.3 .Estándares seguidos

---

orientación a objetos, sin embargo, la programación orientada a objetos viene siendo un complemento perfecto de UML, pero no por eso se toma UML sólo para lenguajes orientados a objetos.

UML cuenta con varios tipos de diagramas, los cuales muestran diferentes aspectos de las entidades representadas.

- **Bluetooth:** Bluetooth es una especificación industrial para Redes Inalámbricas de Área Personal (WPANs) que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia en la banda ISM de los 2,4 GHz. Los principales objetivos que se pretenden conseguir con esta norma son:
  - Facilitar las comunicaciones entre equipos móviles y fijos.
  - Eliminar cables y conectores entre éstos.
  - Ofrecer la posibilidad de crear pequeñas redes inalámbricas y facilitar la sincronización de datos entre equipos personales.

Los dispositivos que con mayor frecuencia utilizan esta tecnología pertenecen a sectores de las telecomunicaciones y la informática personal, como PDA, teléfonos móviles, computadoras portátiles, ordenadores personales, impresoras o cámaras digitales.

- **JSR82:** Se trata de una API para Java para el manejo de conexiones mediante tecnología Bluetooth. Esta API está dividida en dos partes: el paquete javax.bluetooth y el paquete javax.obex.

Los dos paquetes son totalmente independientes. El primero de ellos define clases e interfaces básicas para el descubrimiento de dispositivos, descubrimiento de servicios, conexión y comunicación. La comunicación a través de javax.bluetooth es a bajo nivel: mediante flujos de datos o mediante la transmisión de arrays de bytes. Por el contrario el paquete javax.obex permite manejar el protocolo de alto nivel OBEX (OBject EXchange). Este protocolo es muy similar a HTTP y es utilizado sobre todo para el intercambio de archivos. El protocolo OBEX es un estándar desarrollado por IrDA y es utilizado también sobre otras tecnologías inalámbricas distintas de Bluetooth.

La plataforma principal de desarrollo del API JSR-82 es J2ME. El API ha sido diseñada para depender de la configuración CLDC. Sin embargo existen implementaciones para poder hacer uso de este API en J2SE.

Para más detalles ver Anexo V: Bluetooth:JSR 82

## 5 DESARROLLO SOFTWARE DEL PROYECTO

---

### 5.1 *Planificación*

Tal y como se comentó en el capítulo 4, Metodología de desarrollo y herramientas, para el desarrollo del proyecto se decide usar una metodología de desarrollo iterativo ya que se ajusta bastante bien a nuestro problema. La aplicación de esta metodología implica la división del problema en una serie de iteraciones, mini-proyectos, cuya envergadura debe seleccionarse de forma que estas iteraciones tengan una duración de entre 3 y 5 semanas para forzar la obtención de objetivos a corto plazo.

Por tanto, para llevar a cabo la planificación se hace necesario definir cada una de las iteraciones de las que constará el proceso de desarrollo. Cada una de estas iteraciones estará formada por un conjunto de etapas, organizadas en la estructura indicada en el capítulo 4, Metodología de desarrollo y herramientas. Según dicha estructura cada iteración constará de planificación, modelado de requisitos, análisis, diseño, implementación y pruebas.

En el presente capítulo vamos a tratar de albergar todo lo referente a la planificación de cada una de las iteraciones en la que se divide el proceso de desarrollo software del presente proyecto.

Los recursos de los que se dispone, software y hardware, para realizar la planificación son los mencionados en el capítulo 4, Metodología de desarrollo y herramientas. En lo que respecta a recursos humanos para el desarrollo del proyecto se cuenta con dos personas. Ambos miembros poseen experiencia en la materia, y poseen capacidad de trabajo en equipo y de organización.

#### 5.1.1 Declaración de alcance

El proceso de desarrollo software que vamos a llevar a cabo será un proceso basado en 4 iteraciones. Como ya se ha mencionado en capítulos anteriores, se establecen iteraciones de 3-5 semanas. Se trata de un proceso del software iterativo y evolutivo, en el que cada iteración dará soporte al proceso siguiente. En cada iteración se trata de llevar a cabo los objetivos planteados pero en caso de no ser así la siguiente cubre los retrasos en los objetivos de la anterior iteración. Para todo el proceso se usa una metodología de desarrollo orientada a objetos. Para representar los modelos durante el proceso de desarrollo software se usará como lenguaje de modelado UML.

##### 5.1.1.1 **Objetivo general**

El objetivo general es construir un sistema para la integración y

## 5.1 .Planificación

---

publicación de noticias.

Este sistema dispondrá de un subsistema para gestión de noticias a través de un Web Service. Los usuarios, con permisos para poder efectuar las operaciones, podrán crear, modificar, eliminar, etc, las noticias efectuando una llamada a la operación correspondiente del Web Service. Con esta técnica el sistema podrá ser usado de forma sencilla en otras plataformas, y éste integrar noticias procedentes de múltiples sistemas externos.

Para la publicación de las noticias se usaran dos mecanismos: vía Web (mediante pantallas) y vía Bluetooth. Las noticias serán publicadas a través de pantallas colocadas en diferentes puntos y enviadas por Bluetooth a los dispositivos móviles de los usuarios registrados en el sistema.

Para poder llevar a cabo las operaciones ofrecidas por el Web Service se llevará un control de usuarios. Todas las noticias registradas en el sistema deberán pertenecer a una de las categorías existentes. Dicha categoría será gestionada por un determinado usuario establecido por el administrador del sistema. Será ese usuario el que podrá llevar a cabo operaciones sobre el sistema usando el Web Service (Solo podrá efectuar operaciones sobre la información relacionada con la categoría que gestiona). Por tanto, solo podrán interactuar con el sistema aquellos usuarios registrados en él (gestores de categorías).

Cada dispositivo móvil que sea detectado recibirá vía Bluetooth las noticias que en ese momento se están mostrando por las pantallas. Además de dichas noticias, si el dispositivo ha sido registrado, a través del Web Service, y se encuentra asociado a otras categorías, recibirá las noticias pertenecientes a las categorías personales.

Por las pantallas, distribuidas por el edificio de la E.T.S.I.I.T. solo aparecerán noticias pertenecientes a una categoría que se usara con ese fin.

### 5.1.1.2 Objetivos iteraciones

El desarrollo del proyecto se divide en cuatro iteraciones cuyos objetivos deben estar bien delimitados y definidos. Por tanto es necesario establecer los objetivos a cumplir en cada una de las 4 iteraciones.

El desarrollo del proyecto comienza el 01 de Noviembre de 2010 y se realiza la planificación de las distintas etapas teniendo como fecha objetivo para la finalización del mismo el día 01 de Mayo de 2011.

En los siguientes apartados se muestran los objetivos a cumplir en cada uno de las iteraciones en las que se ha dividido el proyecto, junto con las restricciones en cuanto a rendimiento, fiabilidad, disponibilidad y restricciones temporales.

### **A) Objetivos iteración 1**

En esta primera iteración el primer punto va a ser tratar de llevar a cabo el estudio inicial del problema, análisis y diseño del sistema completo desde una visión global. Una vez llevado a cabo el punto indicado el objetivo será tratar de modelar, analizar, diseñar e implementar por completo el subsistema de almacenamiento permanente (Base de datos) y llevar a cabo el desarrollo por completo el modelo de datos del sistema y el Web Service parcialmente. Del Web Service solamente se va a desarrollar la parte relacionada con la publicación de noticias vía Web, sin llegar a desarrollar la parte relacionada con dispositivos móviles (operaciones para la gestión de envío de noticias mediante Bluetooth)

Para poder llevar a cabo el proceso de desarrollo será necesario disponer del software para poder realizar todas las tareas. Preparar tanto el subsistema a desarrollar como los recursos (software y hardware), será tarea de esta primera iteración.

Dada la importancia de esta primera iteración la duración estimada será la más extendida de las iteraciones en las que se divide el proceso incremental.

En lo que respecta a rendimiento la parte del sistema que se desarrollará en esta iteración no requerirá cómputo elevado por lo que no será necesario un hardware avanzado. En cuanto al software que se va a desarrollar, tampoco se requiere una eficiencia de respuesta en “tiempo real”, por lo que se dedicará el solo el tiempo estrictamente necesario y razonable para la optimización del código.

Sin embargo, se deberá tener en cuenta la forma en la que se reparte la ocupación del dispositivo Bluetooth encargado de enviar noticias a los distintos dispositivos porque existe un número máximo de conexiones simultáneas. Para ello se deberá estudiar sobre el tamaño máximo de la información enviada en cada sesión para evitar la inanición del resto de dispositivos móviles.

También se tratará de llevar a cabo una configuración e implementación del subsistema de Base de datos óptima para las operaciones más utilizadas en el sistema.

La parte a desarrollar en esta iteración debe ser robusta ante posibles errores porque un fallo puede originar problemas en la gestión de los distintos elementos. Entre los fallos que se pueden originar en esta iteración están los que afectan a la base de datos, ya sea por corrupción de los datos internos, perdiendo así su integridad, o por fallo global de la base de datos, por eliminación total de ésta o por fallo del sistema de gestión de base de datos. Para evitar estos posibles fallos se tratará de llevar a cabo un diseño tolerante a fallos de la BD (copias de seguridad, roles, atomicidad, etc).

El resto de errores de fiabilidad pueden venir por un mal desarrollo del software. Se debe llevar a cabo un desarrollo acorde con lo obtenido en el proceso de desarrollo software para garantizar que el software es fiable.

## 5.1 .Planificación

---

La disponibilidad vendrá determinada por ambos servidores: Servidor de aplicaciones y servidor de base de datos. Se tratará de seleccionar en la fase de desarrollo software los servidores que más seguridad nos den en cuanto al aspecto que estamos tratando.

En cuanto a restricciones, la principal que existe es la temporal, ya que se establece una fecha límite para la iteración y debe ser cumplida. El plazo de finalización de la primera iteración concluye el día 01/12/2010.

### **B) Objetivos iteración 2**

En esta segunda iteración se va a proceder a la finalización del Web Service y la implementación del subsistema de publicación de noticias en las pantallas (Vía Web). El objetivo será tratar de modelar, analizar, diseñar e implementar por completo el Web Service. Del Web Service ya se ha desarrollado la parte relacionada con la publicación de noticias vía Web, sin llegar a desarrollar la parte relacionada con dispositivos móviles (operaciones para la gestión de envío de noticias mediante Bluetooth) que será realizada en la siguiente iteración.

El sistema ya se encuentra preparado, tanto el subsistema a desarrollar como los recursos (software y hardware).

Esta iteración será de menor duración ya que se tratará de una iteración para finalizar por completo un subsistema del sistema que está en un 70% desarrollado.

La parte del sistema que se desarrollará en esta iteración no requerirá cómputo elevado por lo que no será necesario un hardware avanzado. En cuanto a el software que se va a desarrollar, tampoco se requiere una eficiencia de respuesta en “tiempo real”, por lo que se dedicará el solo el tiempo estrictamente necesario y razonable para la optimización del código.

La parte a desarrollar en esta iteración debe ser robusta ante posibles errores porque un fallo puede originar problemas en la gestión de los distintos elementos. Entre los fallos que se pueden originar en esta iteración están los que afectan a la base de datos, ya sea por corrupción de los datos internos, perdiendo así su integridad, o por fallo global de la base de datos, por eliminación total de ésta o por fallo del sistema de gestión de base de datos. Para evitar estos posibles fallos ya se trató de llevar a cabo un diseño tolerante a fallos de nuestra Base de Datos (copias de seguridad, roles...) en la iteración uno.

El resto de errores de fiabilidad pueden venir por un mal desarrollo del software. Se debe llevar a cabo un desarrollo acorde con lo obtenido en el proceso de desarrollo software para garantizar que el software es fiable.

La disponibilidad vendrá determinada por ambos servidores: Servidor de aplicaciones y servidor de base de datos. En la iteración uno se trató de seleccionar los servidores que más seguridad nos den en cuanto al aspecto que

## 5.1 .Planificación

---

estamos tratando.

La principal restricción que existe es, al igual que para el resto de iteraciones la temporal, ya que se fijará una fecha límite para la iteración y debe ser cumplida. El plazo de finalización de la primera iteración concluye el día 23 de Diciembre de 2010.

### C) Objetivos iteración 3

En esta tercera iteración se diseñará y se implementará el subsistema Bluetooth propiamente dicho, encargado del reconocimiento de dispositivos móviles, composición y envío de noticias. Este subsistema tendrá como base el modelo implementado en la primera iteración y funcionará conjuntamente con el subsistema de Web Services.

El subsistema que se desarrollará en esta iteración no requerirá cómputo elevado por lo que no será necesario un hardware avanzado. En cuanto al software que se va a desarrollar, tampoco se requiere una eficiencia de respuesta en “tiempo real”, por lo que se dedicará el solo el tiempo estrictamente necesario y razonable para la optimización del código.

Sin embargo, se deberá tener en cuenta la forma en la que se reparte la ocupación del dispositivo Bluetooth encargado de enviar noticias a los distintos dispositivos porque existe un número máximo de conexiones simultáneas. Para ello se deberá estudiar sobre el tamaño máximo de la información enviada en cada sesión para evitar la inanición del resto de dispositivos móviles.

También se tratará de llevar a cabo una configuración e implementación del subsistema de Base de datos óptima para las operaciones más utilizadas en el sistema.

La parte a desarrollar en esta iteración debe ser robusta ante posibles errores porque un fallo puede originar problemas en la gestión de los distintos elementos. Entre los fallos que se pueden originar en esta iteración están los que afectan a la base de datos, ya sea por corrupción de los datos internos, perdiendo así su integridad, o por fallo global de la base de datos, por eliminación total de ésta o por fallo del sistema de gestión de base de datos. Para evitar estos posibles fallos se tratará de llevar a cabo un diseño tolerante a fallos de nuestra BD (copias de seguridad, roles, atomicidad ...).

El resto de errores de fiabilidad pueden venir por un mal desarrollo del software. Se debe llevar a cabo un desarrollo acorde con lo obtenido en el proceso de desarrollo software para garantizar que el software es fiable.

En esta iteración será muy importante tener en cuenta la itinerancia de los dispositivos móviles así como los posibles fallos de cobertura Bluetooth que podrán provocar transacciones incompletas y que deberán repetirse en caso de fallo.

La disponibilidad vendrá determinada por ambos servidores: Servidor de aplicaciones y servidor de base de datos. Se tratará de seleccionar en la fase de

## 5.1 .Planificación

---

desarrollo software los servidores que más seguridad nos den en cuanto al aspecto que estamos tratando.

La principal restricción que existe es la temporal, ya que se establece una fecha límite para la iteración y debe ser cumplida. El plazo de finalización de la tercera iteración concluye el día 11 de Marzo de 2011.

En cuanto a hardware se deberá implementar el subsistema de forma que sea compatible con la mayoría de los dispositivos móviles de la actualidad. También se tendrán que utilizar dispositivos Bluetooth que soporten las pilas implementadas por las librerías Bluetooth que se elijan. Se deberá utilizar librerías en Java y que sigan los estándares más utilizados en los distintos sistemas operativos móviles y de ordenadores portátiles.

### D) Objetivos iteración 4

En esta cuarta iteración se va a proceder a la finalización del subsistema de publicación de noticias vía Bluetooth y la implementación de un cliente del Web Service como un ejemplo de posibilidad de uso del sistema. El objetivo será tratar de finalizar el subsistema Bluetooth (diseño de interfaz) y modelar, analizar, diseñar e implementar por completo un cliente sencillo de Web Service. Del Bluetooth ya se ha desarrollado gran parte del subsistema en la iteración anterior.

El sistema ya se encuentra preparado, tanto el subsistema a desarrollar como los recursos (software y hardware).

Esta iteración será de menor duración ya que se tratará de una iteración para finalizar por completo un subsistema del sistema que está en un 70% desarrollado.

La parte del sistema que se desarrolla en esta iteración no requerirá computo elevado por lo que no será necesario un hardware avanzado. En cuanto a el software que se va a desarrollar, tampoco se requiere una eficiencia de respuesta en “tiempo real”, por lo que se dedicará el solo el tiempo estrictamente necesario y razonable para la optimización del código.

El subsistema a desarrollar en esta iteración debe ser robusta ante posibles errores porque un fallo puede originar problemas en la gestión de los distintos elementos. Entre los fallos que se pueden originar en esta iteración están los que afectan a la base de datos, ya sea por corrupción de los datos internos, perdiendo así su integridad, o por fallo global de la base de datos, por eliminación total de ésta o por fallo del sistema de gestión de base de datos. Para evitar estos posibles fallos ya se trató de llevar a cabo un diseño tolerante a fallos de nuestra BD (copias de seguridad, roles...) en la iteración uno.

El resto de errores de fiabilidad pueden venir por un mal desarrollo del software. Se debe llevar a cabo un desarrollo acorde con lo obtenido en el proceso de desarrollo software para garantizar que el software es fiable.

La disponibilidad vendrá determinada por los dispositivos Bluetooth y por

## 5.1 .Planificación

---

el servidor de base de datos. En la iteración uno se trató de seleccionar los servidores que más seguridad nos den en cuanto al aspecto que estamos tratando. En cuanto a los dispositivos de Bluetooth a usar se analizarán en esta iteración.

La principal restricción que existe para cumplir los objetivos de la iteración es la temporal ya que, como para el resto de iteraciones, se establece una fecha límite para la finalización de la iteración y debe ser cumplida. El plazo de finalización de la primera iteración concluye el día 24/03/2011.

### **5.1.2 Planificación temporal y organizativa**

Como primera tarea de cada iteración se lleva a cabo una planificación temporal y organizativa de recursos. Para ello se debe realizar una estimación temporal para a continuación llegar a establecer la planificación temporal de la iteración. La planificación temporal nos aportará la organización a lo largo del tiempo de la realización de las tareas que componen el proyecto. Existen dos posibilidades para llevar a cabo las estimaciones temporales en cada una de las 4 iteraciones de las que consta el proceso de desarrollo software:

- Basándose en hechos históricos.
- Descomponiendo en subtareas.

En función de la iteración en la que nos encontremos se podrá usar una o ambas posibilidades.

Para estimar la duración de cada una de las subtareas y en definitiva de cada iteración, se va a considerar que se trabaja un determinado número de horas al día, concretamente 3 horas diarias.

A continuación se muestra la planificación de cada una de las iteraciones del proceso de desarrollo.

#### **A) Iteración 1**

En esta iteración no es posible basarse en la experiencia de las iteraciones anteriores para realizar una estimación sobre las tareas que se han de realizar y el tiempo que conlleva cada una. Por tanto nos basaremos en una descomposición en subtareas para realizar una planificación más simple. Para estimar los costes temporales mediante esta división de tareas lo que se hace es prever el número de horas necesarias para cada subtarea. La planificación de esta primera iteración se realiza de forma concurrente con el modelado de requisitos.

La iteración 1 comienza el día 1 de Noviembre de 2010, teniendo como fecha de fin de iteración el día 30 de Noviembre de 2010. Para cada una de las tareas se cumplen las fechas mostradas en el siguiente diagrama de Gantt.

## 5.1 .Planificación

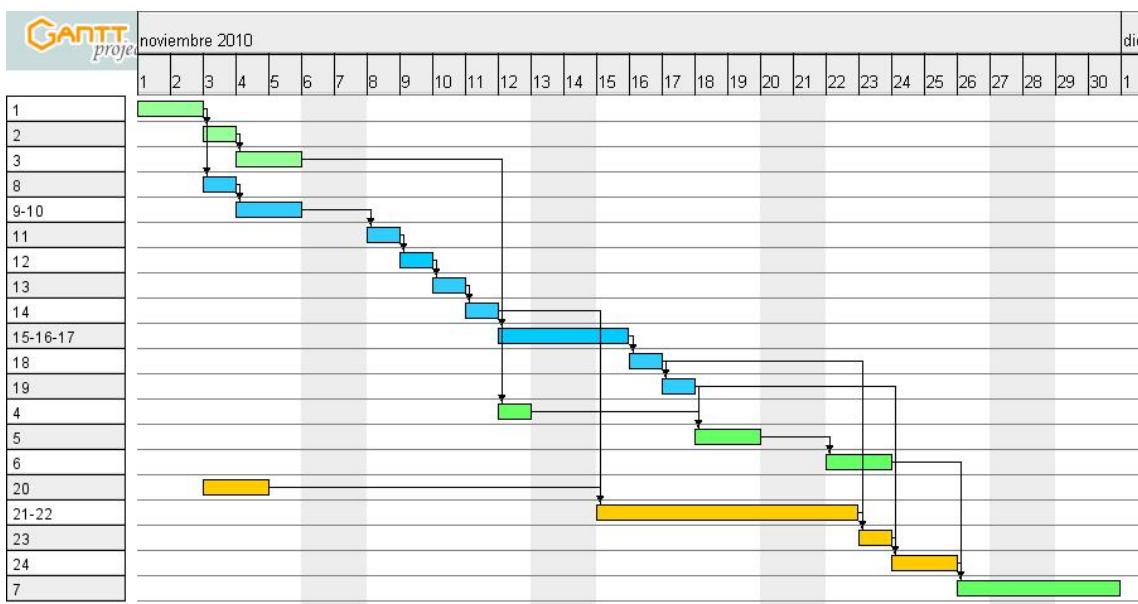


Ilustración 5.1.1: Planificación temporal iteración 1

Teniendo en cuenta lo comentado sobre la consideración de trabajo de 3 horas diarias, la estimación temporal resultante es:

Número	Tarea	Duración (días)	Personas	Duración Total (h/(dia)persona * 2persona * dia)
<b>PLANIFICACIÓN</b>				
1	Planificación temporal y organizativa	2	2	<b>12h</b>
2	Diagrama de Gantt	0.5	1	<b>1,5h</b>
3	Ánálisis de riesgos	2	2	<b>12h</b>
4	Revisión documento modelado de requisitos y documento de análisis	1	2	<b>6h</b>
5	Revisión del documento de diseño	1	2	<b>6h</b>
6	Diseño de las pruebas	2	2	<b>12h</b>

## 5.1 .Planificación

Número	Tarea	Duración (días)	Personas	Duración Total (h/(dia)persona * 2persona * dia)
7	Realización de las pruebas	3	2	<b>18h</b>
<b>MODELADO, ANALISIS Y DISEÑO</b>				
8	Identificación de requisitos funcionales.	1	1	<b>3h</b>
9	Diagramas de casos de uso	0.5	2	<b>3h</b>
10	Descripción detallada casos de uso	0.5	2	<b>3h</b>
11	Identificar subsistemas (Diagrama de paquetes).	0.5	2	<b>3h</b>
12	Identificación de los requisitos no funcionales.	0.5	2	<b>3h</b>
13	Modelado estático (Diagrama de clases)	1	2	<b>6h</b>
14	Determinación Arquitectura	1	2	<b>6h</b>
15	Diagrama paquetes (Diseño)	1	2	<b>6h</b>
16	Diagrama de componentes	1	2	<b>6h</b>
17	Diagrama de despliegue	1	2	<b>6h</b>
18	Diseño de la base de datos (Diagrama E-R)	1	2	<b>6h</b>
19	Diseño interfaz acceso BD (Clases)	1	2	<b>6h</b>
<b>IMPLEMENTACIÓN</b>				
20	Instalación/configuración software necesario.	2	2	<b>12h</b>

## 5.1 .Planificación

Número	Tarea	Duración (días)	Personas	Duración Total (h/(dia)persona * 2persona * dia)
21	Implementación de modelo	1	2	<b>6h</b>
22	Implementación del Web Service	5	2	<b>30h</b>
23	Implementación de la BD	1	2	<b>6h</b>
24	Implementación interfaz acceso BD	2	2	<b>18h</b>
<b>TOTAL</b>				<b>190,5 horas</b>

Tabla 1: Estimación temporal iteración 1

La planificación organizativa establecida queda reflejada en la tabla anterior para cada una de las tareas de la iteración en el campo de la tabla *Personas*. Aún así veamos de forma gráfica la planificación de los recursos humanos en esta iteración:

## 5.1 .Planificación

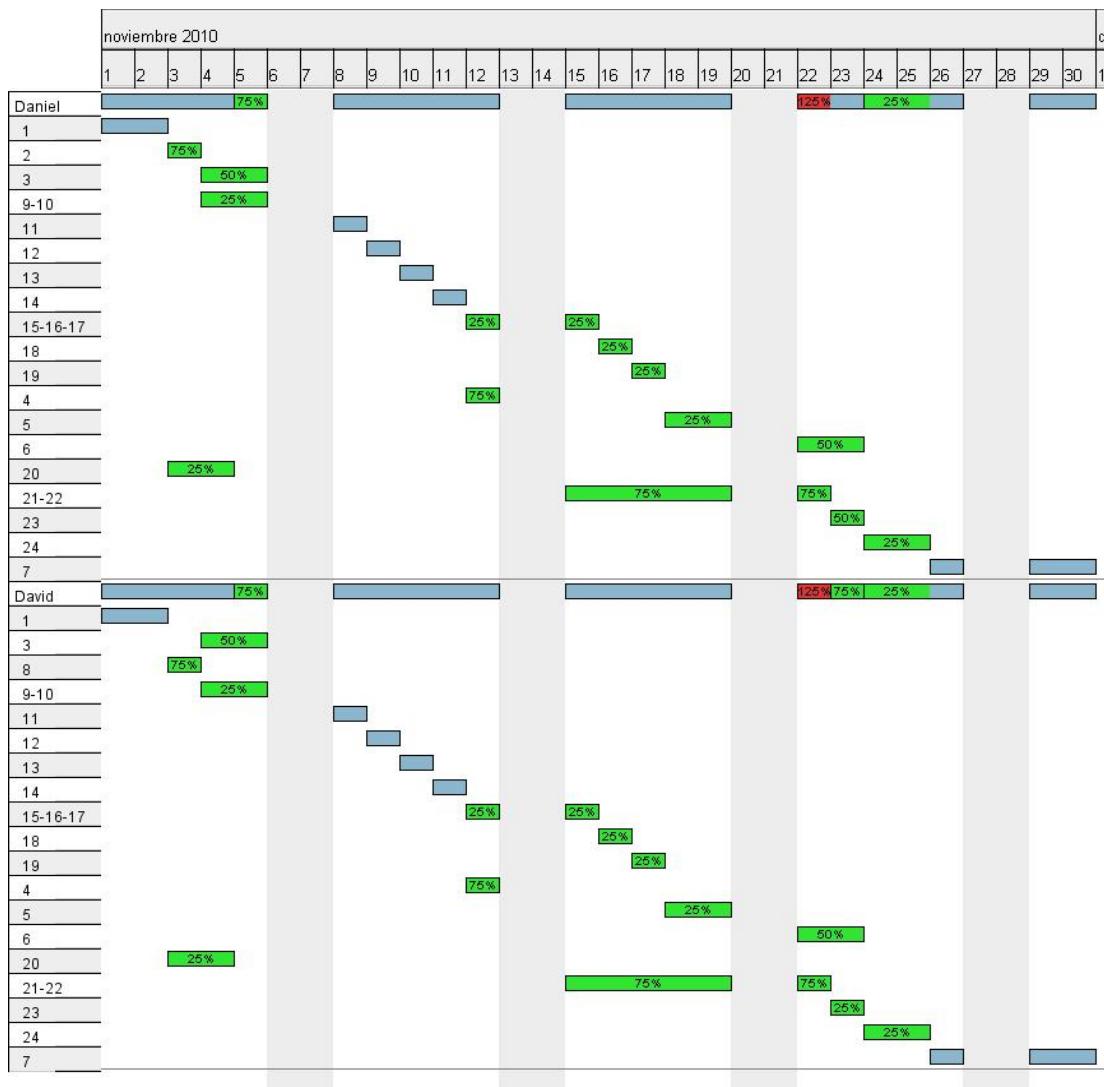


Ilustración 5.1.2: Planificación recursos humanos iteración 1

El emplear un recurso en más de un 100% supondrá el hecho de que existe la posibilidad de que el recurso, humano en este caso, puede superar esas tres horas de trabajo diarias. Por ese motivo la tabla anterior muestra una estimación, no son las horas exactas empleadas.

Ambos miembros poseen todos los conocimientos necesarios para llevar a cabo los objetivos de esta iteración. Por ese motivo no se ha introducido dentro de la planificación ningún periodo de aprendizaje o búsqueda de información.

### B) Iteración 2

En esta iteración ya es posible basarse en la experiencia de las iteraciones anteriores por lo que es más sencillo cumplir con las fechas fijadas. Nos basaremos en la composición de sub-tareas realizada en la primera

## 5.1 .Planificación

iteración. Además, se combinará la previsión temporal de la anterior iteración con el tiempo empleado final; de esta manera se pretende obtener una mejor estimación en esta etapa.

Como en la anterior iteración, la planificación de esta segunda se ha realizado de forma concurrente con el modelado de requisitos. También se ha tenido en cuenta al planificar temporalmente la iteración que, aunque ya existe una base que nos facilitará el trabajo, que la iteración 2 sea lo más corta posible. Por tanto, se trata de hacer concurrente todo el trabajo posible.

La iteración 2 abarca el periodo de tiempo comprendido entre el día 1 de Diciembre de 2010 y el día 22 de Diciembre de 2010. La distribución temporal de las tareas de la iteración es la mostrada en el siguiente diagrama de Gantt.

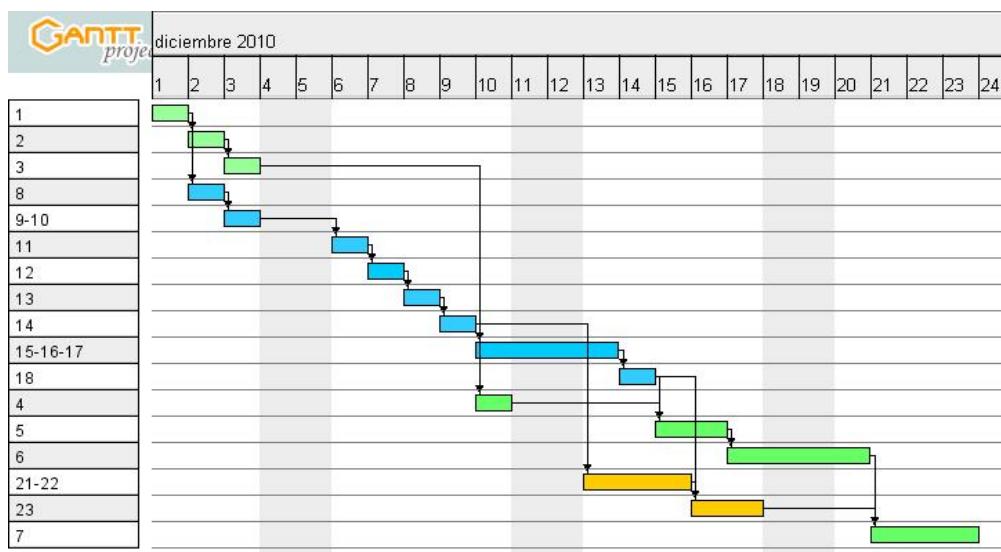


Ilustración 5.1.3: Planificación temporal iteración 2

Teniendo en cuenta lo comentado sobre la consideración de trabajo de 3 horas diarias, la estimación temporal resultante es:

Número	Tarea	Duración (días)	Personas	Duración Total (h/(dia)persona * 2persona * dia)
<b>PLANIFICACIÓN</b>				
1	Planificación temporal y organizativa	1	2	<b>6</b>

## 5.1 .Planificación

Número	Tarea	Duración (días)	Personas	Duración Total (h/(dia)persona * 2persona * dia)
2	Diagrama de Gantt	0.5	1	<b>1,5</b>
3	Análisis de riesgos	1	1	<b>3</b>
4	Revisión documento modelado de requisitos y documento de análisis	1	1	<b>3</b>
5	Revisión del documento de diseño	1	2	<b>6</b>
6	Diseño de las pruebas	2	2	<b>12</b>
7	Realización de las pruebas	3	2	<b>18</b>
8	Identificación de requisitos funcionales.	1	2	<b>6</b>
9	Diagramas de casos de uso	0.5	2	<b>3</b>
10	Descripción detallada casos de uso	0.5	2	<b>3</b>
11	Identificar subsistemas (Diagrama de paquetes).	0.5	2	<b>3</b>
12	Identificación de los requisitos no funcionales.	0.5	2	<b>3</b>
13	Modelado estático (Diagrama de clases)	0,5	2	<b>3</b>
14	Determinación Arquitectura	0,5	2	<b>3</b>
15	Diagrama paquetes (Diseño)	0,5	1	<b>1,5</b>
16	Diagrama de componentes	0,5	1	<b>1,5</b>

## 5.1 .Planificación

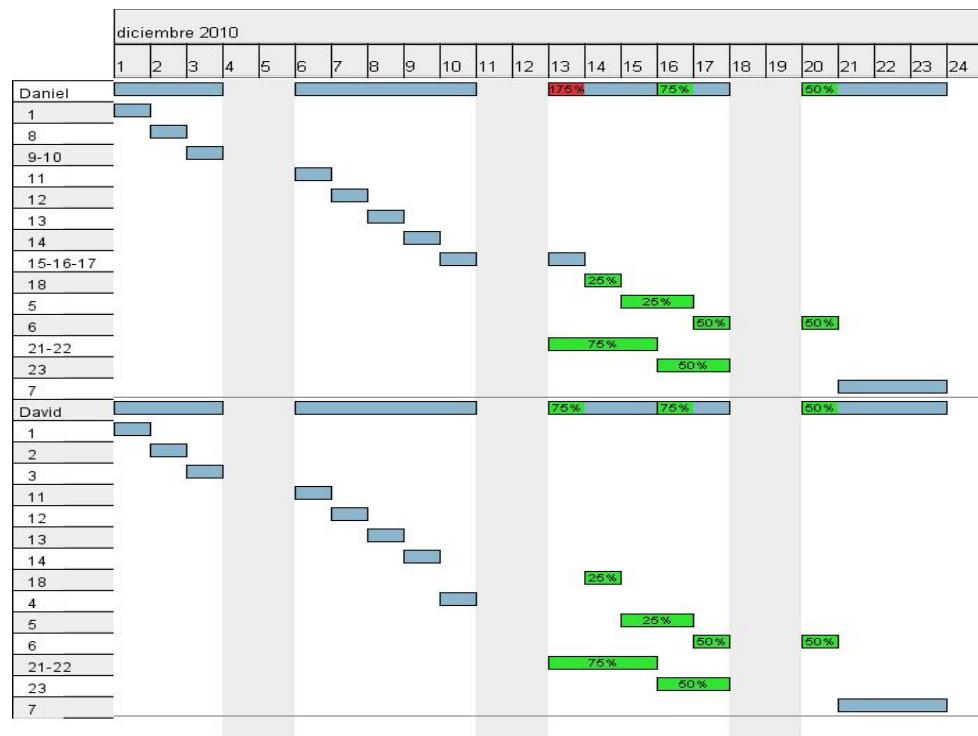
Número	Tarea	Duración (días)	Personas	Duración Total (h/(dia)persona * 2persona * dia)
17	Diagrama de despliegue	0,5	1	<b>1,5</b>
18	Diseño de la Interfaz de pantallas	1	2	<b>6</b>
<b>IMPLEMENTACIÓN</b>				
21	Implementación de modelo	1	2	<b>6</b>
22	Implementación del Web Service	3	2	<b>18</b>
23	Implementación de la interfaz	1	2	<b>6</b>
<b>TOTAL</b>				<b>114 horas</b>

Tabla 2: *Estimación temporal iteración 2*

La planificación organizativa establecida queda reflejada en la tabla anterior para cada una de las tareas de la iteración en el campo de la tabla *Personas*. Aún así veamos de forma gráfica la planificación de los recursos humanos en esta iteración:

## 5.1 .Planificación

---



*Ilustración 5.1.4: Planificación recursos iteración 2*

El emplear un recurso en más de un 100% supondrá el hecho de que existe la posibilidad de que el recurso, humano en este caso, puede superar esas tres horas de trabajo diarias. Por ese motivo la tabla anterior muestra una estimación, no son las horas exactas empleadas.

Ambos miembros poseen todos los conocimientos necesarios para llevar a cabo los objetivos de esta iteración. Por ese motivo no se ha introducido dentro de la planificación ningún periodo de aprendizaje o búsqueda de información.

### C) Iteración 3

En esta iteración es posible basarse en la experiencia de las iteraciones anteriores. Se basará en la composición de sub-tareas realizada en iteraciones anteriores. Además, se combinará la previsión temporal de la anterior iteración con el tiempo empleado final; de esta manera se pretende obtener una mejor estimación en esta etapa.

Como en la anterior iteración, la planificación de esta segunda se ha realizado de forma concurrente con el modelado de requisitos.

La iteración 3 abarca el periodo de tiempo comprendido entre el día 28 de Enero de 2011 y el día 11 de Marzo de 2011. La distribución temporal de las tareas de la iteración es la mostrada en el siguiente diagrama de Gantt.

## 5.1 .Planificación

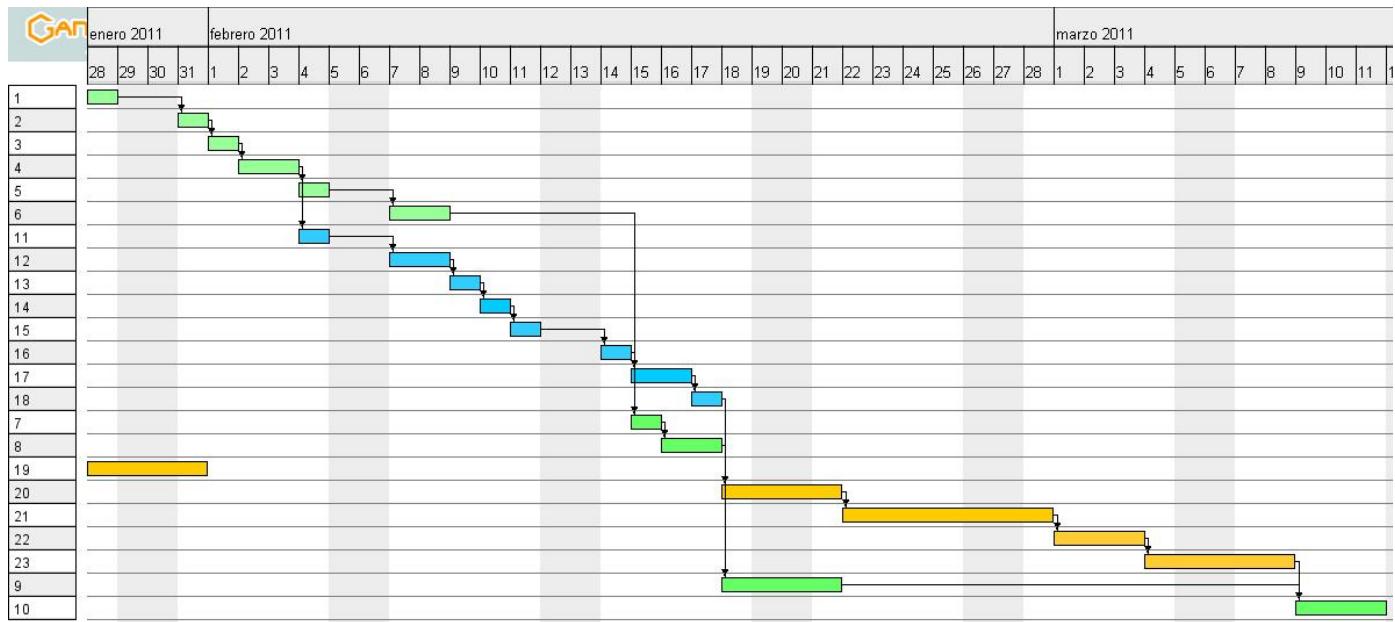


Ilustración 5.1.5: Planificación temporal iteración 3

Teniendo en cuenta lo comentado sobre la consideración de trabajo de 3 horas diarias, la estimación temporal resultante es:

Número	Tarea	Duración (días)	Personas	Duración Total (h/(dia)persona * 2persona * dia)
<b>PLANIFICACIÓN</b>				
1	Investigación sobre tecnología Bluetooth	1	2	<b>6h</b>
2	Búsqueda de proyectos parecidos	1	2	<b>6h</b>
3	Análisis de riesgos. Elección entre las distintas librerías de Bluetooth	1	2	<b>6h</b>
4	Planificación temporal y organizativa	2	2	<b>12h</b>
5	Diagrama de Gantt	1	2	<b>6h</b>

## 5.1 .Planificación

Número	Tarea	Duración (días)	Personas	Duración Total (h/(dia)persona * 2persona * dia)
6	Análisis de riesgos	2	2	<b>12h</b>
7	Revisión documento modelo de requisitos y documento de análisis	1	2	<b>6h</b>
8	Revisión del documento de diseño	2	2	<b>12h</b>
9	Diseño de las pruebas	2	2	<b>12h</b>
10	Realización de las pruebas	3	2	<b>18h</b>
<b>MODELADO, ANALISIS Y DISEÑO</b>				
11	Identificación de requisitos funcionales	1	2	<b>6h</b>
12	Diagrama casos uso y descripción	2	2	<b>12h</b>
13	Identificación subsistemas	1	2	<b>6h</b>
14	Identificación requisitos no funcionales	1	2	<b>6h</b>
15	Diagrama de clases	1	2	<b>6h</b>
16	Determinación Arquitectura	1	2	<b>6h</b>
17	Diagrama de Paquetes y componentes,Despliegue (Diseño)	2	2	<b>12h</b>
18	Adaptación de la base de datos	1	2	<b>6h</b>
<b>IMPLEMENTACIÓN</b>				

## 5.1 .Planificación

Número	Tarea	Duración (días)	Personas	Duración Total (h/(dia)persona * 2persona * dia)
19	Instalación/configuración software necesario.	2	2	12h
20	Estudio y aprendizaje de la librería escogida	2	2	12h
21	Implementación subsistema Detección de dispositivos	5	2	30h
22	Implementación composición de datos para el envío	3	2	18h
23	Implementación subsistema de Envío de datos	3	2	18h
<b>TOTAL</b>				<b>246 horas</b>

La planificación organizativa establecida queda reflejada en la tabla anterior para cada una de las tareas de la iteración en el campo de la tabla *Personas*. Aún así veamos de forma gráfica la planificación de los recursos humanos en esta iteración:

## 5.1 .Planificación

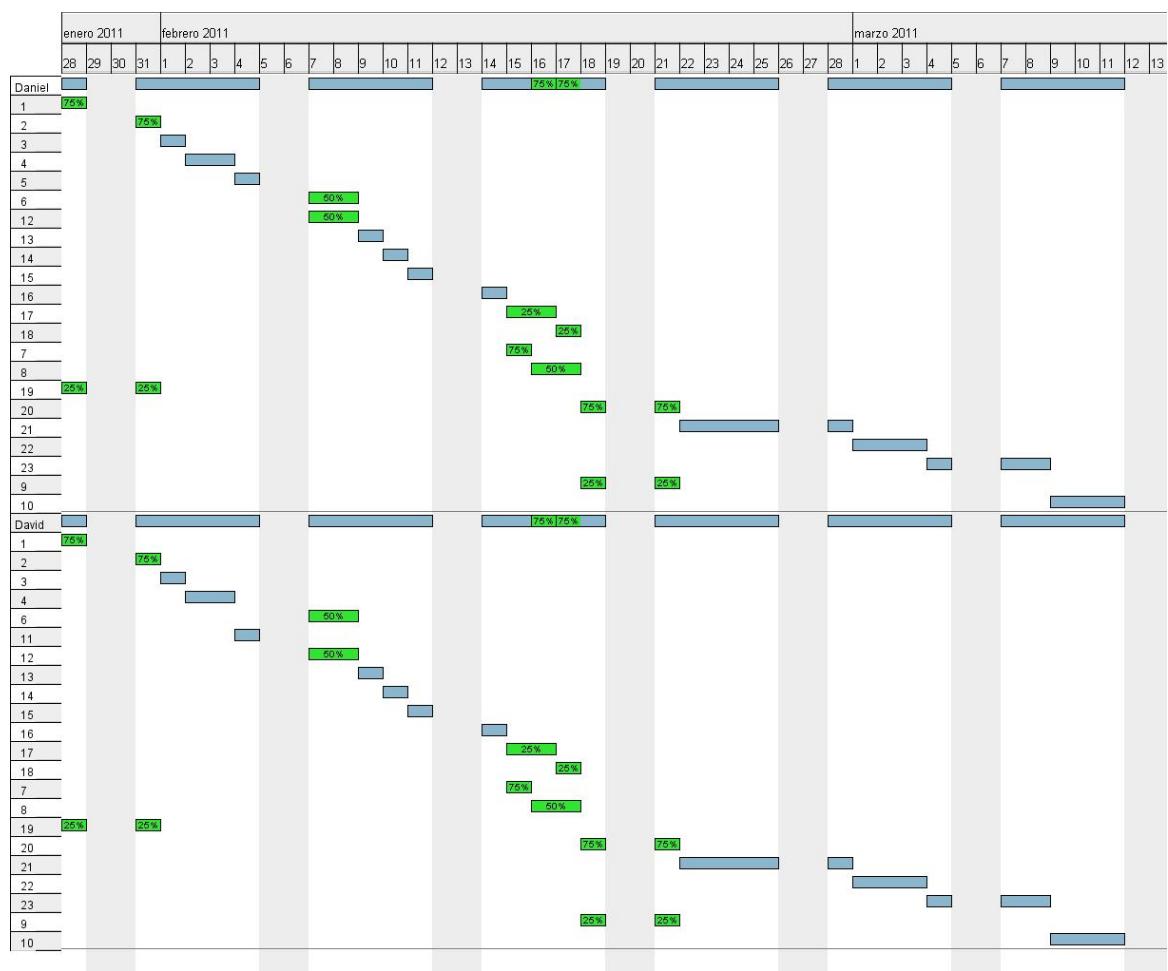


Ilustración 5.1.6: Planificación recursos iteración 3

El emplear un recurso en más de un 100% supondrá el hecho de que existe la posibilidad de que el recurso, humano en este caso, puede superar esas tres horas de trabajo diarias. En caso de ser menor al 100% supondrá una jornada diaria menor a esas 3 horas. Por ese motivo la tabla anterior muestra una estimación, no son las horas exactas empleadas.

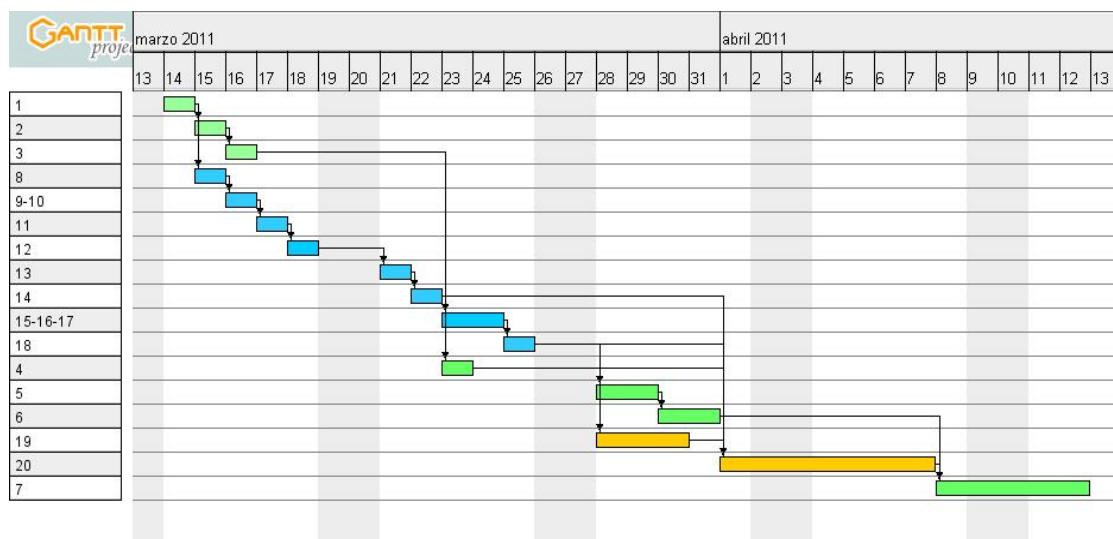
## 5.1 .Planificación

---

### D) Iteración 4

En esta iteración es posible basarse en la experiencia de las iteraciones anteriores. Se basará en la composición de sub-tareas realizada en iteraciones anteriores. Además, se combinará la previsión temporal de la anterior iteración con el tiempo empleado final; de esta manera se pretende obtener una mejor estimación en esta etapa.

La iteración tiene comienzo el día 14 de Marzo de 2011 y finaliza el día 13 de Abril de 2011.



*Ilustración 5.1.7: Planificación temporal iteración 4*

Teniendo en cuenta lo comentado sobre la consideración de trabajo de 3 horas diarias, la estimación temporal resultante es:

### 5.1 .Planificación

Núm	Tarea	Duraci ón (días)	Personas	Duración Total (h/(dia)persona * 2persona * dia)
<b>PLANIFICACIÓN</b>				
1	Planificación temporal y organizativa	1	2	<b>6</b>
2	Diagrama de Gantt	0.5	1	<b>1,5</b>
3	Ánálisis de riesgos	1	1	<b>3</b>
4	Revisión documento modelado de requisitos y documento de análisis	1	1	<b>3</b>
5	Revisión del documento de diseño	1	2	<b>6</b>
6	Diseño de las pruebas	2	2	<b>12</b>
7	Realización de las pruebas	3	2	<b>18</b>
<b>ANALISIS Y DISEÑO</b>				
8	Identificación de requisitos funcionales.	1	1	<b>3</b>
9	Diagramas de casos de uso	0.5	1	<b>1,5</b>
10	Descripción detallada casos de uso	0,5	1	<b>1,5</b>
11	Identificar subsistemas (Diagrama de paquetes).	0.5	1	<b>1,5</b>
12	Identificación de los requisitos no funcionales.	1	1	<b>3</b>
13	Modelado estático (Diagrama de clases)	1	1	<b>3</b>

## 5.1 .Planificación

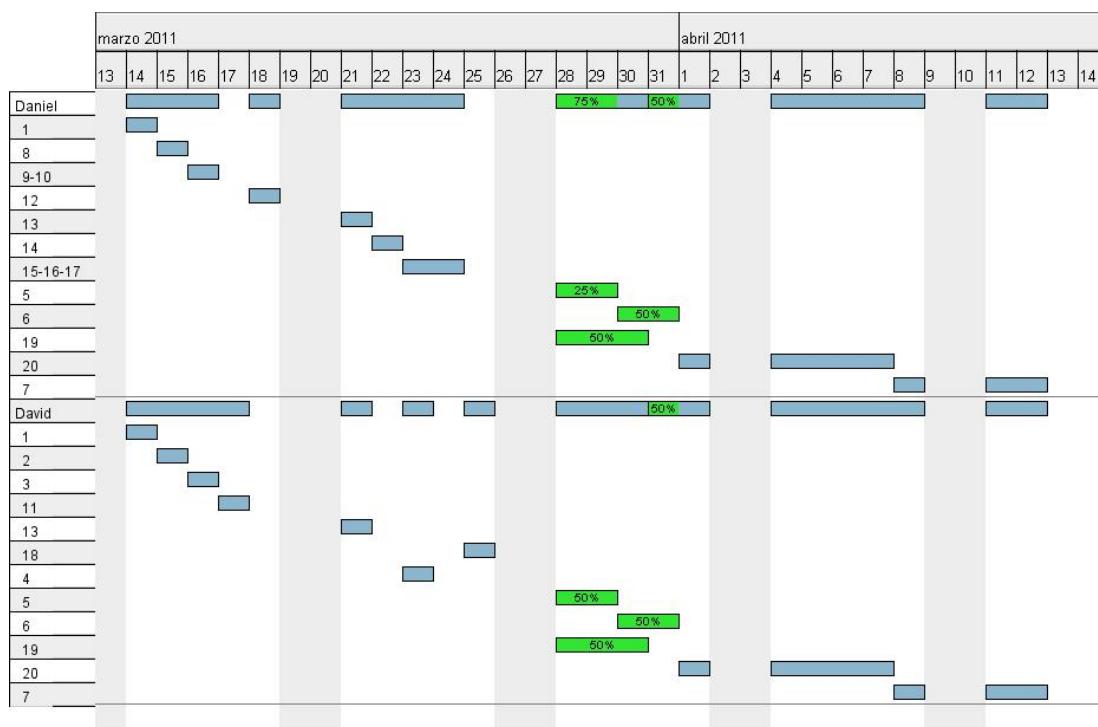
Núm	Tarea	Duraci ón (días)	Personas	Duración Total (h/(dia)persona * 2persona * dia)
14	Determinación Arquitectura	0,5	1	<b>1,5</b>
15	Diagrama paquetes (Diseño)	0,5	1	<b>1,5</b>
16	Diagrama de componentes	0,5	1	<b>1,5</b>
17	Diagrama de despliegue	0,5	1	<b>1,5</b>
18	Diseño de la Interfaces	2	2	<b>12</b>
<b>IMPLEMENTACIÓN</b>				
19	Implementación de interfaz de usuario Bluetooth	3	2	<b>18</b>
20	Implementación del Cliente de Web Service	5	2	<b>30</b>
<b>TOTAL</b>				<b>129 horas</b>

Tabla 3: Estimación temporal iteración 4

La planificación organizativa establecida queda reflejada en la tabla anterior para cada una de las tareas de la iteración en el campo de la tabla *Personas*. Aún así veamos de forma gráfica la planificación de los recursos humanos en esta iteración:

## 5.1 .Planificación

---



*Ilustración 5.1.8: Planificación recursos humanos iteración 4*

## **5.2 Modelado del sistema**

En esta etapa se define el propósito del sistema. Se analiza *qué va debe hacer* el sistema obteniendo una visión contextualizada del mismo. Se tratará de identificar y delimitar y se determinarán las características, cualidades y restricciones que debe de satisfacer. Dado que no existe una figura como tal de cliente que tenga unas determinadas necesidades a partir de las cuales extraer el propósito del sistema, deberemos de extraer dicho propósito a partir del estudio y análisis del problema llevado a cabo en el capítulo 3 ,Solución para el proyecto junto con las reuniones realizadas entre tutor de proyecto y alumno.

### **5.2.1 Especificación de requisitos**

Desde el punto de vista de la ingeniería del software un requisito es una característica del sistema o una descripción de algo que el sistema debe ser capaz de hacer con objeto de satisfacer el propósito del sistema. Los requisitos se pueden dividir en dos tipos: *funcionales* y *no funcionales*.

#### **5.2.1.1 Funcionales**

Los requisitos funcionales se encargan de realizar una descripción entre el sistema y su entorno proporcionando servicios que proveerá el sistema o indicando la manera en que éste reaccionará ante determinados estímulos.

Debido a que en el desarrollo no existe la figura de un cliente que tenga unas determinadas necesidades a partir de las cuales extraer los requisitos del sistema, deberemos de extraer dichos requisitos a partir del exhaustivo estudio y análisis del problema y a partir de las charlas mantenidas entre los alumnos y el tutor del proyecto.

Vamos a proceder a la identificación de los objetivos funcionales para el presente proyecto.

#### **A) Requisitos relativos a los subsistemas de publicación de noticias**

- El sistema a desarrollar contará con un subsistema de publicación de noticias vía Web (Pantallas colocadas por el centro).
- El subsistema de publicación mediante pantallas mostrará las noticias de forma continuada.
- El sistema contará con un mecanismo de envío a móviles para publicación de noticias.
- Cada uno de estos dispositivos recibirá a través del subsistema (Bluetooth) de envío a móviles todas las noticias.
- Además de las categorías a las que está asociado un dispositivo móvil, este recibirá las noticias publicadas a través del subsistema de publicación de noticias vía Web (Pantallas del centro).

## 5.2 .Modelado del sistema

---

- Se debe crear un sistema que sea capaz de detectar todos los dispositivos Bluetooth (móviles, portátiles, PDAs, etc.) que estén visibles desde los dispositivos emisores Bluetooth.
- Se comprobará si los dispositivos detectados (a través de su dirección física que es única) ya están registrados en la base de datos o no.
- En caso de que el dispositivo no esté registrado, el sistema deberá registrarla y, por tanto, enviarle todas las noticias que se muestren en ese momento en las pantallas, es decir, a partir de ese momento recibirá las noticias de las pantallas solamente mientras ese dispositivo no se asocie a ninguna categoría.
- En caso de que el dispositivo ya esté registrado se enviarán todas las noticias que posean alguna de las categorías asociadas para ese usuario en concreto y que no hayan sido enviadas previamente para dicho dispositivo.
- Para el envío de ficheros por Bluetooth se deberá añadir algún sistema de seguridad. (Ver requisitos no funcionales)
- No se debe repetir el envío de las mismas noticias tanto si el dispositivo está registrado como si no lo está.

### **B) Requisitos relativos a usuarios**

- El sistema deberá permitir un mecanismo de identificación de usuarios. Los usuarios que podrán interactuar con el sistema serán:
  - Gestor Categoría: Se trata de un usuario encargado de gestionar todo lo relativo a una categoría de las existentes en el sistema.
- Un gestor deberá tener login y password.
- El lógin de un gestor debe de ser único en el sistema.
- Los usuarios no podrán darse de alta por sí mismos en el sistema.
- El administrador del sistema será el único con capacidad para dar de alta usuarios mediante acceso local a la base de datos.
- El administrador del sistema podrá introducir/modificar/eliminar gestores del sistema. Por tanto el login y el password serán inicialmente elegidos por el administrador.
- No se permitirá operaciones de gestión de login y password propios a los gestores. Para ello deberán ponerse en contacto con el administrador del sistema.
- El sistema deberá permitir la realización de operaciones sobre las partes del sistema acorde con el usuario indicado.
- Un gestor podrá crear noticias en el sistema asociadas a las categorías

## 5.2 .Modelado del sistema

---

gestionadas por él.

- Un gestor podrá eliminar cualquier noticia asociada a una de las categorías gestionadas por él.
- Un gestor podrá modificar cualquier campo de una noticia que pertenezca a una de sus categorías.
- Un gestor podrá obtener/consultar todas las noticias publicadas por él, es decir, todas las noticias que pertenecen a sus categorías.
- De igual forma un gestor podrá obtener/consultar las categorías que gestiona en un momento determinado.
- Un gestor no podrá crear/modificar/eliminar una categoría. Será el administrador del sistema el que realice estas operaciones a través de un sistema independiente.
- Los gestores pueden controlar varias categorías. Podrán existir gestores que no gestionen ninguna categoría.
- Los gestores podrán dar de alta, eliminar, modificar dispositivos en el sistema.
- Al dar de alta un dispositivo el gestor podrá indicar una, o varias, de las categorías que él gestiona para que sean asociadas al dispositivo.
- Cualquier gestor puede dar de baja un dispositivo móvil. Para eliminar un dispositivo móvil del sistema, éste no podrá estar asociado a ninguna categoría.
- Un gestor podrá asociar o desasociar categorías a un dispositivo móvil. Solo podrá realizar estas operaciones con categorías que son gestionadas por él.
- Un gestor podrá consultar todos los dispositivos existentes en el sistema, así como los que pertenecen a categorías gestionadas por él.
- Un gestor solo podrá ser dado de baja en el sistema si no existen categorías gestionadas por él, es decir, no existen ni categorías ni noticias pertenecientes a esas categorías.

### **C) Requisitos relativos a noticias, categorías y dispositivos móviles**

- Las noticias deben de tener una fecha a partir de la cual serán publicadas a través de los distintos sistemas de publicación.
- Las noticias deben de tener una fecha a partir de la cual dejarán de ser publicadas en los diferentes subsistemas de publicación de noticias.
- En lo que respecta a la información de una noticia será:
  - Título: Título de la noticia.

## 5.2 .Modelado del sistema

---

- Subtítulo: Subtítulo de la noticia.
- Autor: Nombre y/o apellidos del autor de la noticia.
- Cuerpo de la noticia: texto con toda la información de la noticia.
- Fecha de publicación: Fecha en la que la noticia ha sido o será publicada.
- Fecha de caducidad: Fecha en la que la noticia dejará de aparecer publicada.
- Las categorías del sistema estarán formadas por:
  - Nombre: Nombre de la categoría.
  - Descripción: Breve descripción de la categoría: nombre completo, web, localización, información de contacto, etc.
- Todas las noticias del sistema deberán estar asociadas a una categoría. Un gestor no podrá publicar una noticia sin asociarle una categoría.
- Todas las categorías existentes en el sistema deberán tener un gestor asociado.
- Cada categoría solo podrá ser gestionada por un único gestor.
- Existirá una categoría usada exclusivamente para la publicación de noticias a través del subsistema de publicación de pantallas.
- Todo los dispositivos del sistema estarán asociados a la categoría de pantalla.
- El sistema debe permitir el registro de dispositivos móviles.
- Cuando se registra un dispositivo en el sistema automáticamente se asocia con la categoría seleccionada para albergar las noticias mostradas a través del subsistema de publicación vía pantallas.
- Los dispositivos móviles podrán estar asociados a categorías, de forma que recibirán las noticias que estén publicadas para dichas categorías.
- Para un dispositivo se almacenará:
  - MAC: dirección ethernet del dispositivo Bluetooth.
  - Pin: código que se usará para establecer la comunicación entre el dispositivo y el sistema.
  - URL: dirección del servicio para el envío de archivos. Es el servicio que usará el sistema para enviar las noticias correspondientes al dispositivo.

### **5.2.1.2 No Funcionales**

Los requisitos no funcionales describen cualidades o restricciones del sistema que no se relacionan de forma directa con el comportamiento funcional del mismo. Podemos clasificarlos según el modelo FURPS+:

#### **A) Facilidad de uso (Usability)**

- Aunque los usuarios que harán uso del sistema dispondrán de conocimientos informáticos, sobre todo para la parte del sistema que permitirá la integración del sistema en otras plataformas, se deben de implementar todas las interfaces, gráficas o de servicios, de la forma más simple posible.
- Los usuarios tendrán acceso al manual de uso del sistema, el cual les servirá como ayuda para aprender cómo realizar las operaciones necesarias y disponibles para desempeñar su labor.
- Existirá documentación de código accesible por los usuarios.

#### **B) Fiabilidad (Reliability)**

- El acceso y uso del sistema debe ser independiente de las características software y hardware del equipo desde el que se acceda.
- El sistema constará de un mecanismo de gestión de noticias, es decir una interfaz, que sea integrable de forma sencilla en otras plataformas.
- El sistema debe permitir el acceso simultáneo a los usuarios.
- Se debe establecer un mecanismo de seguridad para el envío de información desde los sistemas externos al sistema en lo que respecta a autenticación. Se deberá usar un sistema de firma del mensaje.
- Los datos registrados deberán ser accesibles los usuarios del sistema, sea cual sea su ubicación, desde el momento en que estos se almacenan en la base de datos.
- Los datos almacenados en la base de datos deberán estar siempre en un estado consistente.
- Se ha de permitir la realización de copias de seguridad de la base de datos completa del sistema.
- Los servidores web y de base de datos deberán estar situados en sistemas de altas prestaciones ya que deben atender las peticiones de todos los usuarios del sistema.
- El servidor de base de datos deberá tener suficiente capacidad de almacenamiento para poder almacenar todas las noticias generadas desde los distintos sistemas externos y poder almacenar todo el historial de noticias enviadas a los dispositivos móviles.

## 5.2 .Modelado del sistema

---

- El subsistema de envío de noticias a móviles debe de ser robusto ante posibles errores en la conexión, reenviando la información necesaria cuando el sistema se recupere del error.
- Se deberá realizar un pairing entre el emisor y el dispositivo detectado tanto si está registrado como si no lo está. El pin utilizado para los dispositivos registrados será el indicado en su creación y que estará recogido en la base de datos, mientras que para los dispositivos sin registrar será un pin que se elija por defecto y que se deberá dar a conocer al resto de personas mediante algún método.
- La base de datos deberá estar protegida convenientemente para evitar accesos no autorizados.
- El sistema de envío de noticias debe ser fiable y detectar si se ha producido algún error durante la transferencia del fichero con las noticias. En caso de detectar algún problema en la transmisión se deberá abortar la transferencia para volver a iniciarla posteriormente.
- Se comprobará si las transacciones han sido completadas con éxito para que no se repitan de nuevo y realizar el proceso de confirmación de forma atómica.

### C) Rendimiento (Performance)

- Los tiempos de respuesta de la aplicación tras una interacción de un usuario determinado deberán ser lo más reducidos posibles.
- El subsistema de publicación mediante pantallas deberá estar lo más actualizado posible en lo que a alta/baja de noticias se refiere.
- Debe de mostrar las noticias en función del tamaño de pantalla y de la cantidad de noticias disponibles. Actuando en caso necesario como un carrusel, en el que las noticias van apareciendo con el paso del tiempo.
- El subsistema de envío de noticias a dispositivos móviles deberá tener el mayor grado de concurrencia posible para poder atender al mayor número posible de dispositivos al mismo tiempo. Para ello deberá aprovechar al máximo la tecnología Bluetooth y utilizar todas las posibles conexiones simultáneas que se permitan.
- Se deberá establecer un tamaño máximo para los datos que se envíen en cada transacción. Con ello se evitará el problema de inanición para el resto de dispositivos, sin embargo, el tamaño no podrá ser demasiado pequeño para que no se reduzca la eficiencia.
- Se deberán establecer timeouts para las distintas operaciones Bluetooth como por ejemplo la obtención del nombre de dispositivo, búsqueda de dispositivos, espera en la introducción del pin, espera para el envío de información etc. Con ello se asegurará que no haya ningún dispositivo que ocupe indefinidamente un recurso.

## 5.2 .Modelado del sistema

---

- Se deberán introducir en nuestra base de datos los dispositivos encontrados y la dirección de conexión del servicio para el envío de datos. Con ello evitamos tener que buscar de nuevo la URL de conexión para un dispositivo cada vez que se le envíen noticias. De este modo aumentamos la eficiencia del dispositivo Bluetooth que realizará únicamente búsquedas para aquellos dispositivos que no conozcamos.

### **D) Soporte (Supportability)**

- Los equipos desde los que se haga uso de la plataforma web deberán tener instalado un navegador web moderno: Internet Explorer, Mozilla Firefox, Google Chrome, Safari, Opera, etc., aunque se aconseja el uso de Mozilla Firefox.
- El sistema debe funcionar con el mayor número de dispositivos móviles existentes en la actualidad. Para ello se debe realizar un estudio previo de las diferentes implementaciones de la pila Bluetooth del sistema operativo de los dispositivos. Por ejemplo se verificará si funciona en sistemas operativos como Android, Symbian, IOS, Bada o sistemas operativos propios de las distintas empresas.

### **E) Implementación**

- Las contraseñas de acceso de los usuarios del sistema deberán almacenarse cifradas en la base de datos por motivos de seguridad.
- Para la implementación de la plataforma web se utilizará el lenguaje de programación interpretado PHP.
- Para la implementación del servicio web y del subsistema de envío de noticias a dispositivos móviles se empleará como lenguaje de programación JAVA.

### **F) Interfaz**

- El subsistema web de publicación de noticias a través de pantallas deberá mostrar como información principal los títulos de las noticias, mostrando y ocultando, tras un tiempo razonable de lectura, el cuerpo de las mismas
- El subsistema de envío de noticias a móviles mostrará todo lo que esté ocurriendo en cada instante a través de una interfaz simple y clara.
- A través de la interfaz que proporciona el sistema de envío de noticias a móviles se podrá ver los dispositivos detectados, sus propiedades.
- También se deberá cuidar el formato de redacción de las noticias, intentando distinguir encabezados y títulos del resto del cuerpo de la noticia para su mejor comprensión, incluso se podrían añadir distintos colores y formato.

## 5.2 .Modelado del sistema

---

- Las noticias enviadas a los dispositivos móviles deberán de tener un formato adecuado a este tipo de hardware para que el usuario pueda acceder a la información de forma eficaz y clara.
- El sistema deberá generar ficheros que sean legibles con facilidad desde cualquier dispositivo móvil o portátil, independientemente del sistema operativo.
- Se tendrá en cuenta las distintas propiedades físicas de los dispositivos como por ejemplo la resolución de pantalla, por lo tanto se debe conseguir que el tamaño de letra y el aspecto general de las noticias sea el mismo independientemente del dispositivo.
- Los subsistemas del sistema que dispongan de interfaz, con las que los usuarios interactúan, deberán ser lo más elementales posibles, con el objetivo de que puedan realizar su labor de la forma sencilla y directa.
- Los posibles mensajes de error mostrados o devueltos al usuario deberán ser sencillos y concisos para que éste detecte de forma inmediata cuál es el problema que se ha producido.
- El subsistema de envío de noticias por Bluetooth deberá mostrar información de todo el proceso de detección y envío de noticias.
- La interfaz debe de adaptarse a los diferentes navegadores existentes. Para ello se deberán usar correctamente los estándares.

### **G) Legales**

- El desarrollo de la aplicación se llevará a cabo con herramientas y tecnologías gratuitas, por lo que en principio no es necesario obtener licencias de ningún software. Además se deberán utilizar librerías de código abierto y con licencias compatibles con nuestra propia licencia del sistema.

### **5.2.2 Modelo funcional y subsistemas funcionales**

A partir de los requisitos funcionales establecidos en el apartado 5.2.1, Especificación de requisitos, podemos proceder a desarrollar el modelo funcional. Para ello debemos identificar los actores, identificar los posibles casos de uso y las relaciones entre ellos y finalmente realizar una descripción en lenguaje natural de la secuencia de pasos que se llevarán a cabo para completar su ejecución.

La división en subsistemas se realiza posteriormente al modelado funcional, pero, para facilitar la compresión y la lectura, se ha separado el estudio del modelo funcional en dos apartados, acorde con los dos grandes subsistemas: parte Web (con subsistemas Web Service junto con cliente Web del Web Service y web pantallas, ) y el subsistema Bluetooth (envío de noticias a dispositivos móviles).

#### **5.2.2.1 Web**

Como se ha comentado anteriormente dentro de este apartado se abarcará el desarrollo del modelo funcional abarcando los subsistemas Web Service, web para pantallas y cliente web del Web Service.

##### **A) Identificación de actores**

Los actores identifican un rol que cierta entidad externa adopta cuando interactúa con su sistema directamente. Puede representar un rol de usuario, o un rol desempeñado por otro sistema o hardware que toca el límite de su subsistema.

Para identificar los actores necesitamos considerar quién y qué utiliza el sistema, y que rol desempeñan. Se han identificado los siguientes actores:

- *Gestor categoría*: Se trata de un usuario que podrá llevar a cabo publicación/modificación/eliminación/consulta de noticias y consulta de categorías (solo las gestionadas por él). En este caso, este actor será un sistema informático externo, ya que al tratarse de un Web Service la forma de interactuar con el sistema será a través de algún otro sistema que haga uso del mismo.
- *Temporizador*. Este actor simplemente ejecuta cada cierto tiempo el caso de uso mostrar noticias pantalla. Formará parte del subsistema web para mostrar noticias por pantallas.

##### **B) Identificación de casos de uso**

Un caso de uso se define como “una especificación de secuencias de acciones, incluidas secuencias variantes y secuencias de error que un sistema, subsistema o clase puede realizar al interactuar con actores externos”, es decir, algo que el actor quiere que el sistema haga. A la hora de identificar los casos de

## 5.2 .Modelado del sistema

---

uso no se debe confundir un evento entre actor y sistema con un caso de uso que agrupa a una secuencia de eventos para producir un beneficio a alguno de los actores. Cuando se identifican casos de uso se puede caer en una granularidad excesiva del caso de uso. No se deben definir casos de uso de muy bajo nivel que definan cómo se va a hacer algo, en lugar de comunicar lo que el sistema hace para el actor.

Teniendo en cuenta las consideraciones anteriores se procede a la identificación de los casos de uso para el sistema. A continuación citamos los diferentes casos de uso que se han identificado. Las relaciones se describirán en el diagrama de casos de uso del siguiente apartado. La explicación detallada de cada uno de los casos de uso puede ser consultada en el Anexo I Subsistema Web. En dicho anexo se muestra la secuencia de eventos que proporcionan el resultado del caso de uso.

Los casos de uso identificados son:

- Crear noticia.
- Modificar noticia.
- Eliminar noticia.
- Consultar noticias.
- Consultar categorías.
- Dar de alta dispositivo móvil.
- Dar de baja dispositivo móvil.
- Modificar dispositivo móvil.
- Asociar categoría a dispositivo.
- Desasociar categoría a dispositivo.
- Obtener dispositivos.
- Mostrar noticias pantalla.

## 5.2 .Modelado del sistema

### C) Diagrama de casos de uso

Los diagramas de casos de uso nos permiten mostrar de forma gráfica los actores, los casos de uso y la relación entre ellos.

En la siguiente figura se muestran el diagrama de casos de uso.

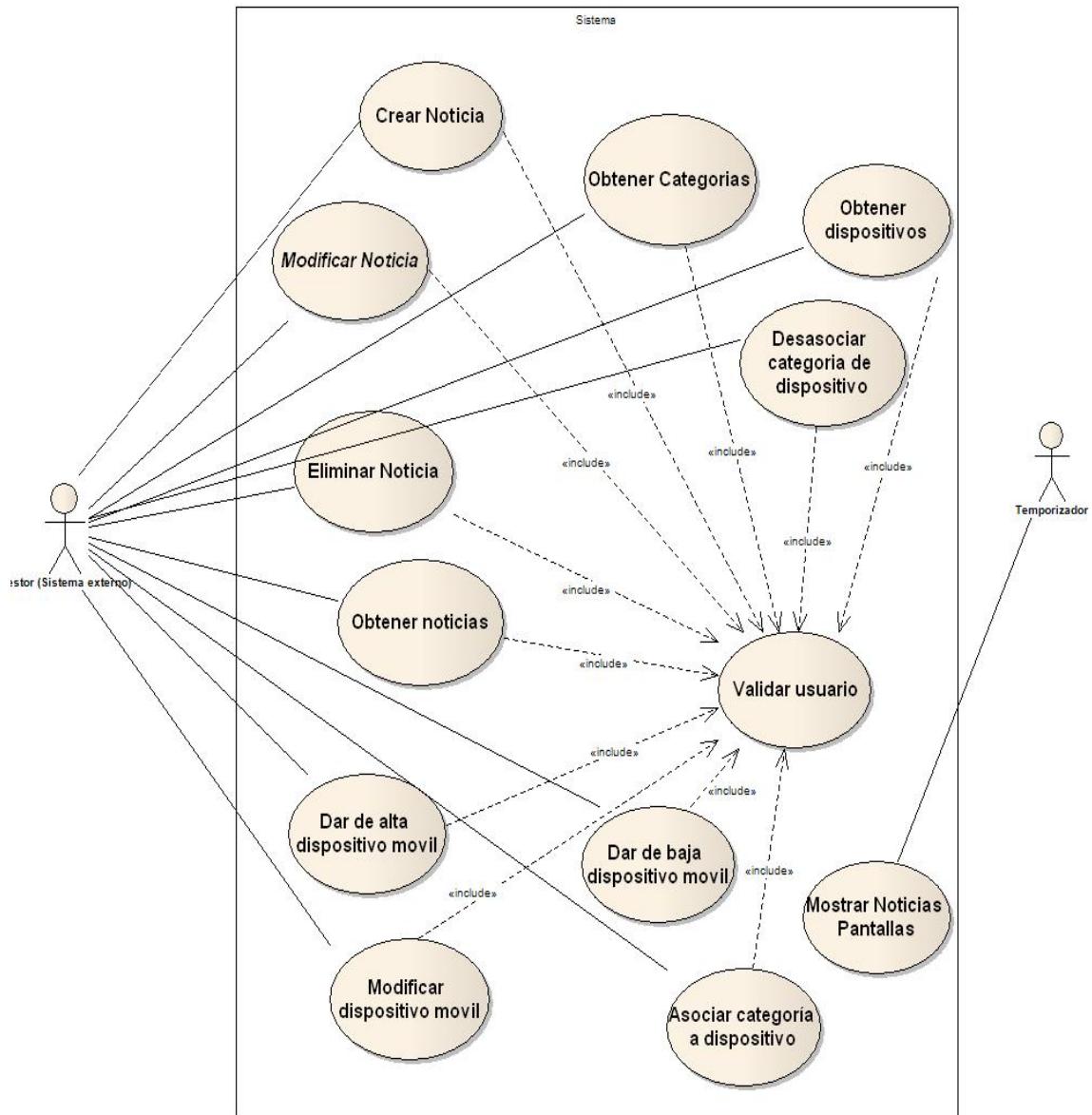


Ilustración 5.2.1: Diagrama Casos de Uso subsistema Web

Como se puede observar el actor Gestor, un sistema externo, se relaciona con prácticamente todos los casos de uso, excepto con el caso de uso MostrarNoticiasPantallas. En este caso, el actor temporizador, que está relacionado con el caso de uso, actúa de forma independiente sobre el sistema, no depende de la acción de ningún usuario.

### 5.2.2.2 Bluetooth

Siguiendo el mismo procedimiento explicado anteriormente en el modelado funcional del subsistema Web, procedemos al modelado del subsistema de envío de noticias mediante Bluetooth.

#### A) Identificación de actores

Para identificar los actores es necesario considerar quién y qué utiliza el sistema, y que rol desempeñan. Para el subsistema de envío de noticias del proyecto se han identificado los siguientes actores:

- *Temporizador*: Se trata de una entidad abstracta que lanzará operaciones periódicamente.
- *Dispositivo registrado*: Es un dispositivo que está registrado en nuestra base de datos y que por tanto tiene asociadas categorías de noticias.
- *Dispositivo no registrado*: Es un dispositivo que se encuentra en el radio de acción del emisor Bluetooth y que no está registrado en la base de datos.

#### B) Identificación de casos de uso

A continuación citamos los diferentes casos de uso que se han identificado en el subsistema Bluetooth. Las relaciones entre cada caso de uso y los actores se describirán en el apartado posterior. La explicación detallada de cada uno de los casos de uso puede ser consultada en el Anexo I Subsistema Bluetooth

- Inicializar Pila Bluetooth
- Escanear dispositivos
- Escanear servicios
- Realizar Pairing
- Enviar noticias
- Generar noticias Dispositivo.

#### C) Diagrama de casos de uso

En la siguiente imagen se muestran el diagrama de casos de uso correspondiente al subsistema Bluetooth.

## 5.2 .Modelado del sistema

---

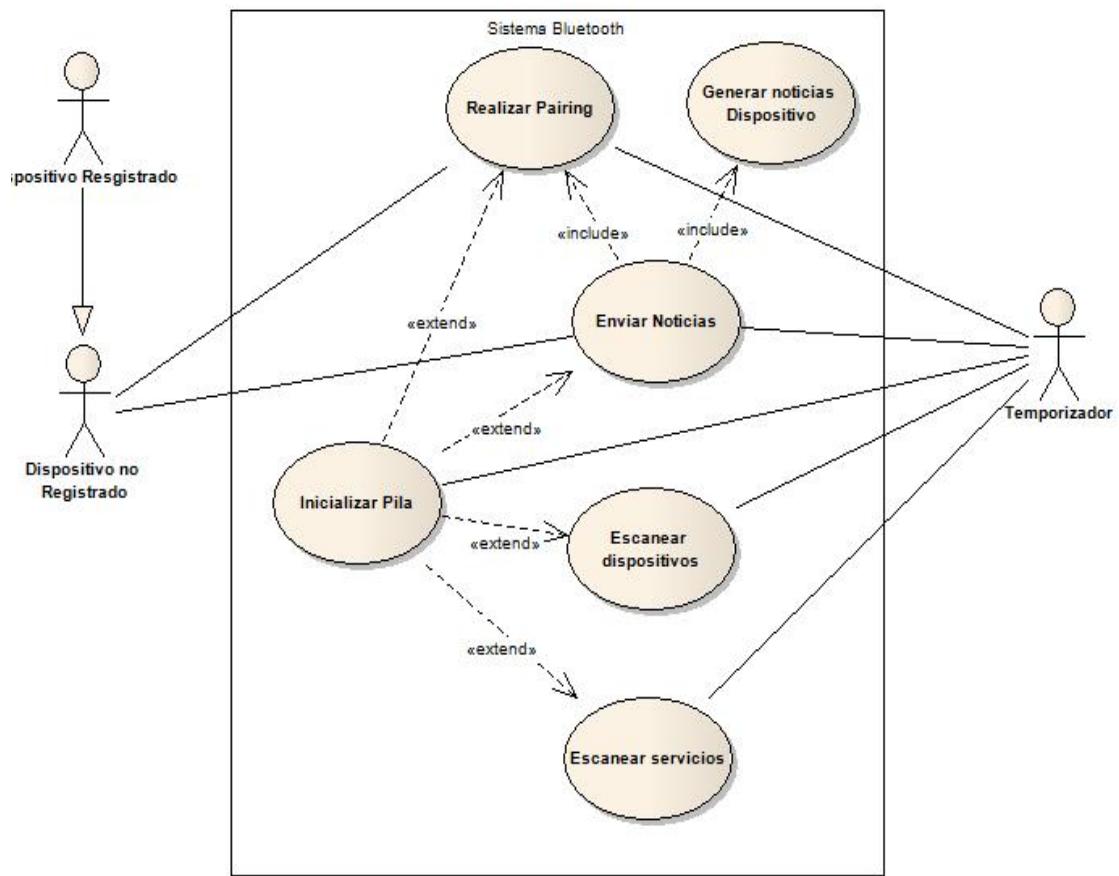


Ilustración 5.2.2: Diagrama Casos de Uso subsistema Bluetooth

## ***5.3 Análisis y Diseño del sistema***

A partir del modelo funcional obtenido en el apartado anterior llevamos a cabo la formalización de los requisitos en modelos que describirán completamente el sistema.

Una vez obtenido el modelo funcional se puede estudiar la posibilidad de descomponer el sistema en subsistemas funcionales más simples de un sistema complejo.

En el presente proyecto ya se ha efectuado una división en subsistemas. En el capítulo 4, Metodología de desarrollo y herramientas se hacía mención a los cuatro subsistemas, que finalmente, conformarán el sistema. Como se especifica al inicio del presente apartado los subsistemas a desarrollar son:

- Web Service: gestor principal, subsistema encargado de toda la gestión de información existente en el sistema.
- Web pantallas: subsistema encargado de mostrar a través de pantallas las noticias correspondientes.
- Cliente Web Service: subsistema que usará a modo de ejemplo práctico todas y cada una de las funcionalidades ofrecidas por el Web Service.

Los subsistemas Web Service, junto con el cliente, y el subsistema web para pantallas comprenderían funcionalidad plasmada en el apartado Web del capítulo 5.2.2, Modelo funcional y subsistemas funcionales y un aplicación independiente abarcaría la funcionalidad relativa al apartado Bluetooth del mismo capítulo. El cliente Web implementado sería un ejemplo del actor hasta el momento conocido como Gestor.

Al igual que en el modelado de requisitos, se realizará el análisis y diseño separándolo en dos apartados: web (Web Service, web pantallas, cliente Web del Web Service) y Bluetooth (subsistema de envío de noticias a dispositivos móviles).

### **5.3.1 Web**

El principal objetivo del análisis es comprender y describir el sistema en el dominio del problema. Se estructuran y se formalizan los requerimientos obtenidos en el modelado de requisitos trabajando con abstracciones del mundo real, obteniendo objetos, situaciones o procesos del mundo real. El análisis está compuesto por dos modelos, modelos estáticos y modelos dinámicos. Los primeros reflejan la estructura estática a partir de los conceptos, clases u objetos, y sus relaciones. Los modelos dinámicos reflejan el comportamiento del sistema y de sus componentes. En los siguientes apartados vamos a centrarnos en los modelos estáticos.

### 5.3.1.1 Análisis: Modelo estático

La primera y más complicada tarea a realizar es la identificación de clases, atributos y relaciones. Para ello se buscan abstracciones significativas que identifiquen los aspectos claves del dominio del problema. Cada una de estas clases estará caracterizada por una serie de rasgos que almacenarán la información relevante de datos. Estas clases conceptuales deben interrelacionarse entre sí a través de las relaciones para poder satisfacer las necesidades de información o de comportamiento que demandan los casos de uso obtenidos en 5.2.2, Modelo funcional y subsistemas funcionales.

El conjunto de clases que se considera necesario para poder desarrollar el subsistema web, es el siguiente:

- **Dispositivo:** representa a cada uno de los dispositivos registrados en el sistema, dispositivos móviles en este caso.
- **Categoría:** representa a todas aquellas categorías registradas en el sistema como tal que nos permiten agrupar las noticias de forma lógica.
- **Usuario:** representa el conjunto de usuarios que pueden hacer uso del sistema.
- **Gestor:** representa el conjunto de usuarios con la caracterización de gestor de una de las categorías.
- **Noticia:** representa el conjunto de noticias que pueden existir en el sistema.

La estrategia llevada a cabo para identificar las clases conceptuales mencionadas anteriormente es la extracción de los sustantivos más relevantes de los que aparecen en el modelado de requisitos. Es una forma sencilla y natural de obtener el diagrama de clases.

En la figura 5.3.1, Diagrama clases análisis se muestra el diagrama de clases estático conceptual. En el podemos apreciar cada una de las relaciones entre las clases identificadas anteriormente.

En el diagrama no aparecen ninguno de los atributos ni métodos de las clases. El motivo ha sido no cargar excesivamente el diagrama ya que sería ilegible. Los atributos de cada una de estas clases se pueden consultar en el Anexo II: Diccionario de datos análisis y diseño.. Por su parte los métodos de cada una de las clases, por ser muy numerosos pueden consultarse directamente en el código Java implementado para cada una de ellas que se adjunta al presente documento.

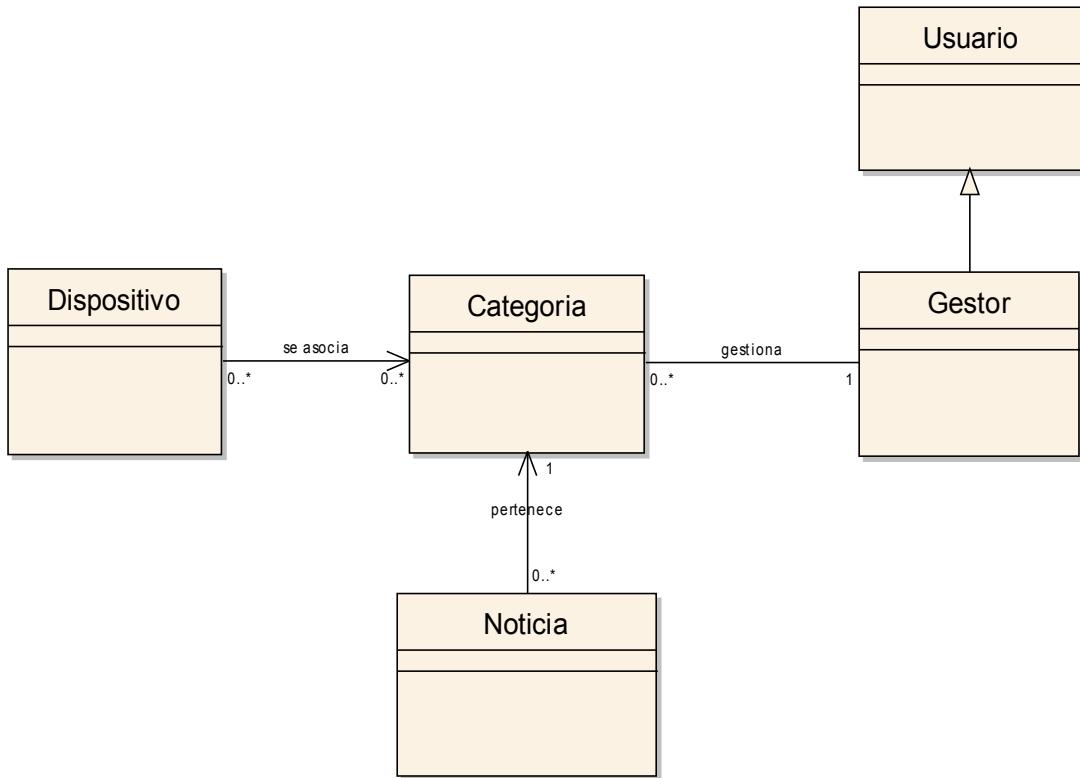
Si bien no vamos a listar los métodos de las clases, destacar que todas ellas incorporan métodos de acceso y asignación para cada uno de los atributos.

Las clases obtenidas conforman el modelo del sistema y serán usadas en todos los subsistemas para manejar toda la información que necesaria para el

### 5.3 .Análisis y Diseño del sistema

---

funcionamiento del mismo.



*Ilustración 5.3.1: Diagrama clases análisis*

Las relaciones establecidas entre cada una de las clases identificadas nos permitirán satisfacer los requisitos impuestos en el apartado 5.2, Modelado del sistema. Veamos las implicaciones de cada una de las relaciones reflejadas en el diagrama de clases estático.

- **se asocia:** relaciona las clases Dispositivo y Categoria. La cardinalidad de la relación es de muchos a muchos, por lo que el sistema permitirá que un dispositivo pueda asociarse a múltiples categorías y a cada categoría podrán estar asociados varios dispositivos móviles.
- **pertenece:** relaciona las clases Noticia y Categoría. La cardinalidad de esta relación establece que una noticia solo pueda estar asociada a una única categoría. Al ser una relación uno-a-muchos las categorías podrán tener asociadas varias noticias.
- **gestiona:** relaciona las clases Gestor y Categoria. De nuevo la cardinalidad es uno-a-muchos. Esta cardinalidad implica que una categoría solamente podrá ser gestionada por un único gestor, pudiendo este gestionar múltiples categorías.

## 5.3 .Análisis y Diseño del sistema

---

- **Generalización:** esta relación establece la relación entre la clase general Usuario y la clase específica Gestor. Esta relación implica que la subclase Gestor hereda todos los atributos y operaciones de la superclase.

### 5.3.1.2 Diseño: Arquitectura

Una vez realizado la etapa de análisis se procede al diseño del sistema. Esta actividad, como se mencionó en el capítulo 4 , Metodología de desarrollo y herramientas, trata de aplicar diferentes técnicas y principios con el propósito de definir el sistema con el suficiente detalle para permitir posteriormente la construcción física, implementación.

En este apartado llegaremos, de forma elaborada, a obtener las arquitecturas software y hardware del sistema. Las arquitecturas software y hardware obtenidas para la solución final son las mostradas en el capítulo 3 , Solución para el proyecto. Teniendo conocimiento del resultado final, será más fácil para el lector el proceso desarrollado en el presente apartado para llegar a dicha solución. Dicho esto procedamos a la obtención, en primer lugar, de la arquitectura software mencionada.

#### A) Arquitectura software

Partiendo de información previa disponible de las etapas de modelado de requisitos, estructuración del sistema en subsistemas, y del análisis, modelo estático, debemos llevar a cabo un estudio de las arquitecturas disponibles y de los objetivos de diseño para establecer la arquitectura software del sistema. Para ello se debe tomar la decisión de cómo se va a dividir el sistema en subsistemas, como interactuarán dichos subsistemas y cuál va a ser la interfaz para cada uno de estos subsistemas.

Como ya se indicó en el capítulo 5.2, Modelado del sistema el sistema consta de cuatro subsistemas lógicos. En este apartado trataremos con los subsistemas: Web Service junto con el cliente y la Web para publicación por pantallas.

Durante esta etapa es necesario el estudio de los estilos arquitectónicos software existentes. Los estilos arquitectónicos nos permitirán expresar la organización estructural del sistema software. Estos no proporcionan la arquitectura software del sistema, pero si nos facilitan una guía de cómo obtener la arquitectura software final al problema.

Los estilos arquitectónicos disponibles son:

- Arquitectura multicapa (Microkernel): modela la interacción entre los subsistemas organizando un sistema en una serie de capas.
  - Cada capa presta servicios a la capa inmediatamente superior y actúa como cliente sobre las capas más internas.

### 5.3 .Análisis y Diseño del sistema

- El diseño incluye los protocolos que establecen cómo interactuará cada par de capas.
- Las capas más bajas proporcionan servicios de bajo nivel como protocolos de comunicación en red, acceso a base de datos...
- Se trata de una arquitectura cambiante y portable; preservando la interfaz una capa se puede sustituir por otra.
- Desventajas:
  - Es difícil estructurar los sistemas pues es posible que el usuario requiera acceso a capas internas.
  - El rendimiento puede resultar afectado por los múltiples niveles de interpretación de órdenes que se requieren a veces.

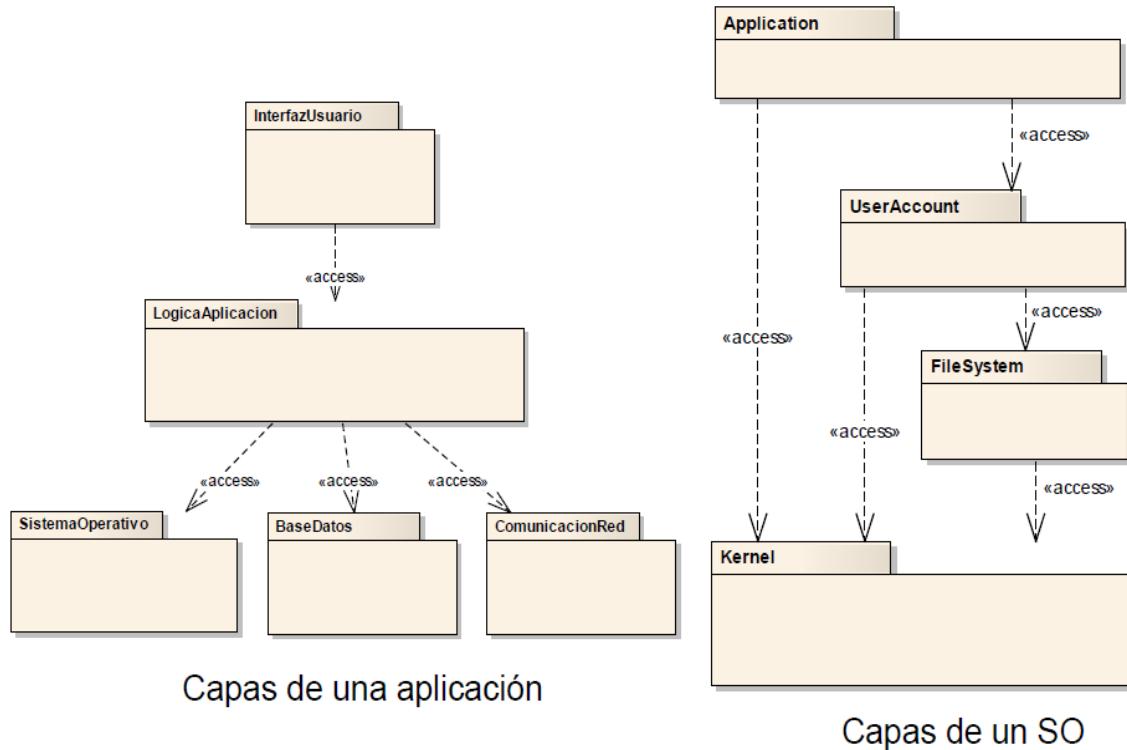


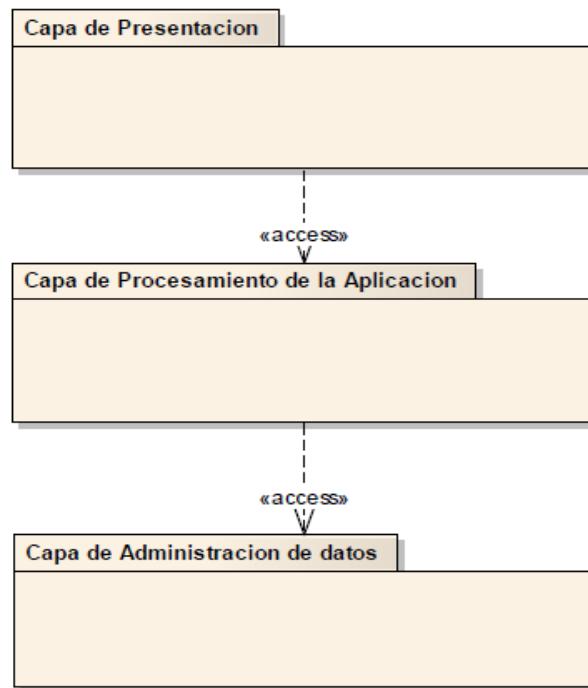
Ilustración 5.3.2: Arquitectura Multicapa

- Arquitectura cliente-servidor: sigue el modelo cliente/servidor:
  - Modelo de sistemas distribuido que muestra cómo datos y procesamiento se distribuyen a lo largo de varios procesadores.
  - Componentes:

### 5.3 .Análisis y Diseño del sistema

---

- Conjunto de servidores independientes que ofrecen servicios a otros subsistemas.
  - Servidores impresión
  - Servidor de administración de archivos.
  - Servidores de compilación.
  - ...
- Conjunto de clientes
  - Invocan los servicios ofrecidos por los servidores.
  - Existen varias instancias de un programa cliente que se ejecutan de forma concurrente.
  - Tienen que conocer los nombres de los servidores disponibles y los servicios que suministran, pero los servidores no conocen a los clientes.
- Una red que permite a los clientes acceder a los servicios



*Ilustración 5.3.3: Arquitectura Cliente/Servidor*

- Arquitectura par a par (Peer to Peer): Se trata de una generalización de la arquitectura cliente/servidor.

### 5.3 .Análisis y Diseño del sistema

---

- Los subsistemas clientes pueden ser servidores y los subsistemas servidores pueden ser clientes.
- Son más difíciles de implementar al poder aparecer problemas de sincronización.

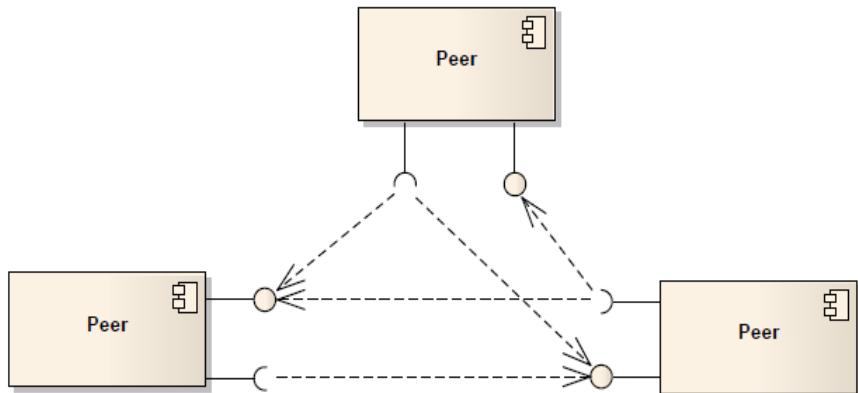


Ilustración 5.3.4: Arquitectura Par a par

- Arquitectura de repositorio (Repository):
  - Arquitectura en la que todos los datos se ubican en una base de datos central a la que acceden todos los subsistemas.
    - Los subsistemas están muy desacoplados al interaccionar sólo a través del subsistema central.
    - El repositorio es pasivo, y el control es responsabilidad de los subsistemas.
  - Útil en sistemas que utilizan grandes cantidades de datos, generados por un subsistema y utilizados por otro.
  - Una variante es la arquitectura de pizarra (blackboard) en el cual el repositorio puede activar algunos subsistemas.

### 5.3 .Análisis y Diseño del sistema

---

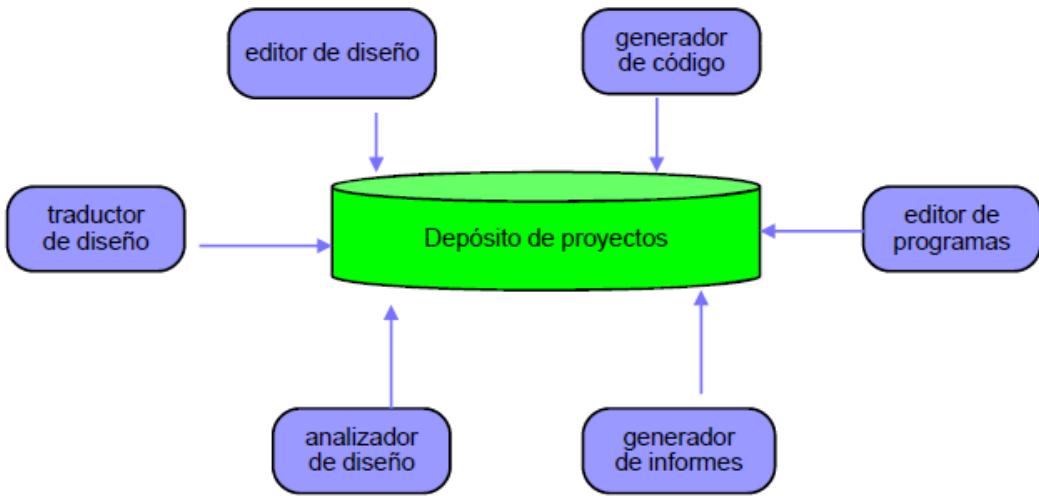


Ilustración 5.3.5: Arquitectura de repositorio

- Arquitectura transformacional de datos:
  - El sistema se estructura en subsistemas que llevan a cabo el procesamiento de flujos de datos.
    - Cada subsistema transforma los datos de entrada y produce datos de salida.
    - Los datos fluyen de un subsistema a otro.
  - Se distinguen dos tipos de arquitecturas transformacionales de datos:
    - Arquitectura secuencial por lotes cuando los subsistemas se ejecutan secuencialmente.
    - Arquitectura de tubería y filtro (pipe and filter) cuando los subsistemas se ejecutan concurrentemente.
  - Se utiliza fundamentalmente para el desarrollo de sistemas de procesamiento de datos que no requieren intervención del usuario.
  - Permite organizar de una forma más intuitiva el procesamiento de datos en términos de entradas y salidas.

### 5.3 .Análisis y Diseño del sistema

---

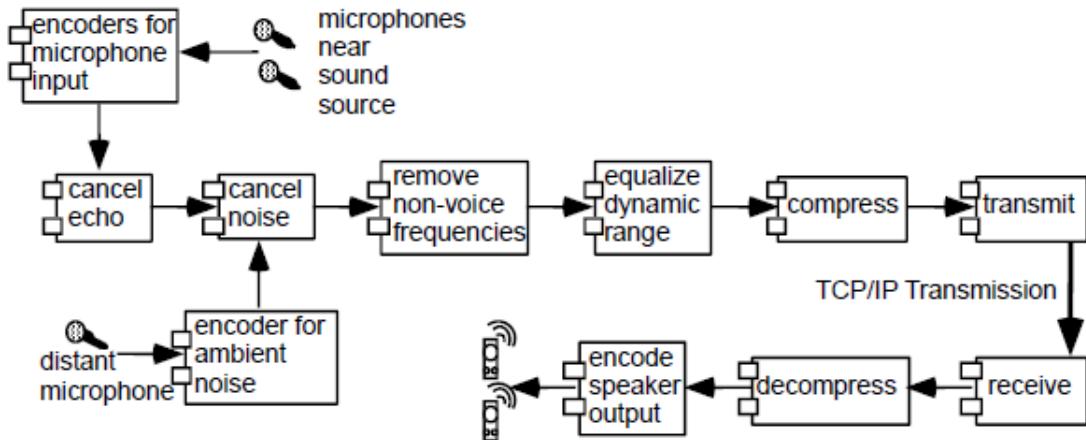


Ilustración 5.3.6: Arquitectura transformacional; Ejemplo

- Arquitectura Modelo-Vista-Controlador: Ayuda a separar la capa de interfaz de usuario de otras partes del sistema.
  - Los subsistemas se clasifican en tres tipos:
    - Subsistema Modelo: Responsable del conocimiento del dominio de aplicación.
    - Subsistema Vista: Responsable de mostrar los objetos del dominio de aplicación al usuario.
    - Subsistema Controlador: Responsable de la secuencia de interacciones entre el usuario.

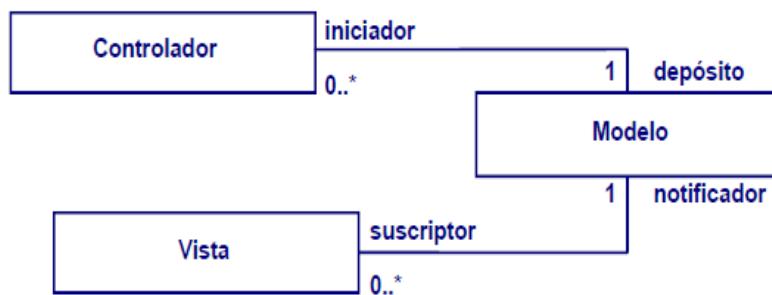


Ilustración 5.3.7: Arquitectura Modelo-Vista-Controlador

### A.1 Web Service

Con el estudio de toda la información mencionada se decide usar para el subsistema Web Service:

- Utilizar una arquitectura basada en capas cerradas que separe la aplicación en dos niveles: lógica de aplicación y servicios (Base de Datos).
  - Razón: Mejora la estructura del sistema y fomenta la flexibilidad en cuanto a sustitución de servicios.
- Utilizar una arquitectura MVC para desacoplar la lógica de aplicación (en este caso el Web Service se podría ver como el controlador).
- Utilizar una arquitectura cliente-servidor para establecer la comunicación entre los subsistemas de cada capa que pueden estar situados en distintos nodos del sistema.
  - Razón: Permite distribuir los distintos subsistemas del sistema reduciendo el acoplamiento sólo a través del protocolo de comunicaciones. Fomenta la interoperabilidad, y facilita la escalabilidad del sistema.

La arquitectura quedaría formada por tres capas cerradas bien diferenciadas:

- Capa Servicios: en esta capa se proporcionan los servicios de más bajo nivel. En esta capa se facilitan los servicios de acceso a base de datos.
- Capa Lógica: en esta capa se implementa toda la lógica de la aplicación. La funcionalidad ofrecida por el servicio web es implementada en la capa Lógica. Esta capa contiene todas las clases que representan el modelo de datos.
- Capa Interfaz: proporciona una interfaz con la que interactuar de forma sencilla con el sistema. Esta capa estará formada por un cliente del servicio web.

La arquitectura en capas con los paquetes propuesta anteriormente es la mostrada en la figura 5.3.8, Arquitectura subsistema Web Service en capas.

### 5.3 .Análisis y Diseño del sistema

---

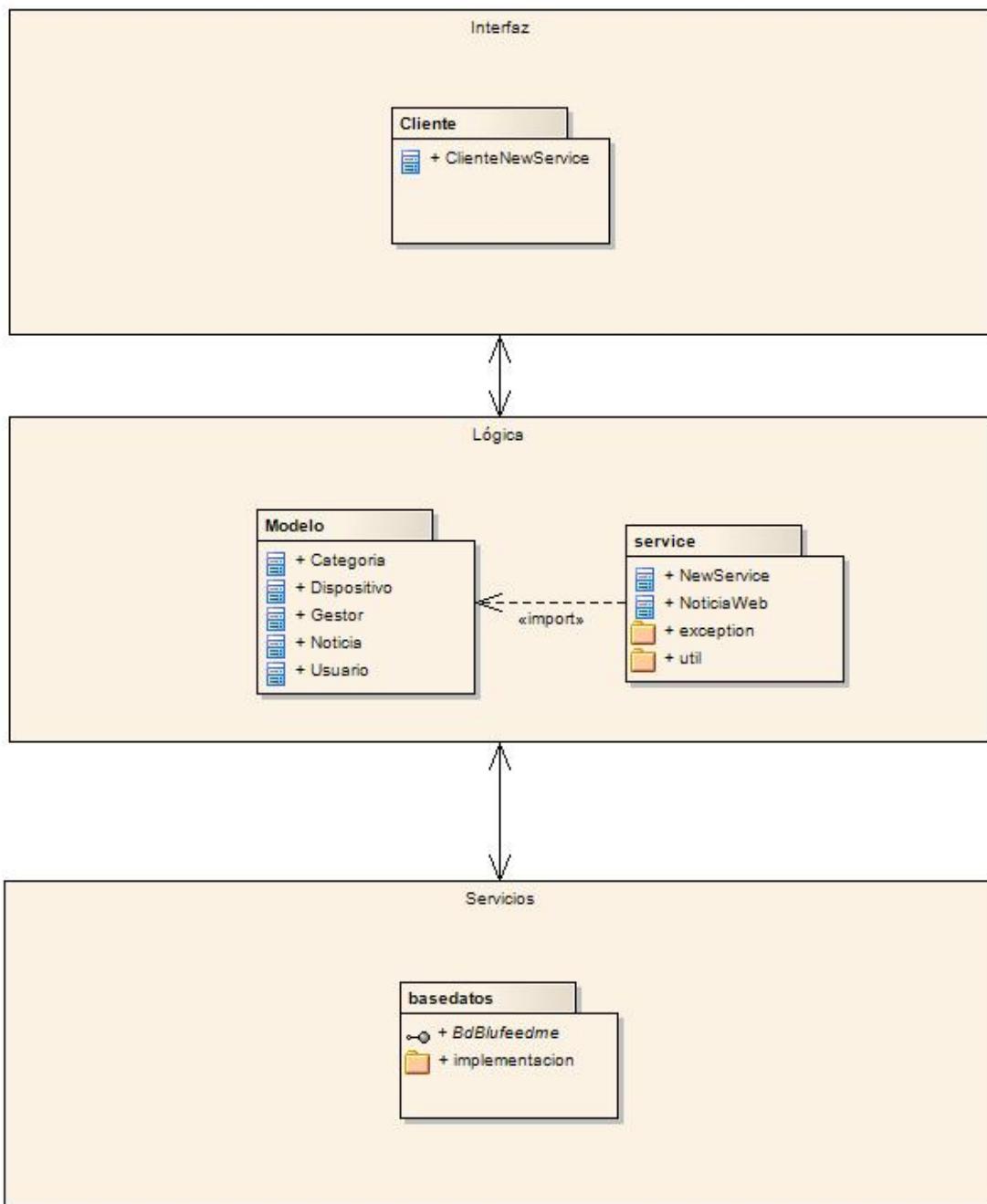


Ilustración 5.3.8: Arquitectura subsistema Web Service en capas

#### A.1.1 Capa de Servicios

En la capa de Servicios disponemos del paquete **basedatos**. Este paquete contiene las clases necesarias para proporcionar el servicio de acceso a la base de datos centralizada del sistema. Dicho servicio ofrece una interfaz con todas las operaciones disponibles, interfaz **BdBlufeedme**. Es el paquete

### 5.3 .Análisis y Diseño del sistema

---

implementación quien contiene la clase que implementa la interfaz mencionada, *BdBlufeedmeMySQL*.

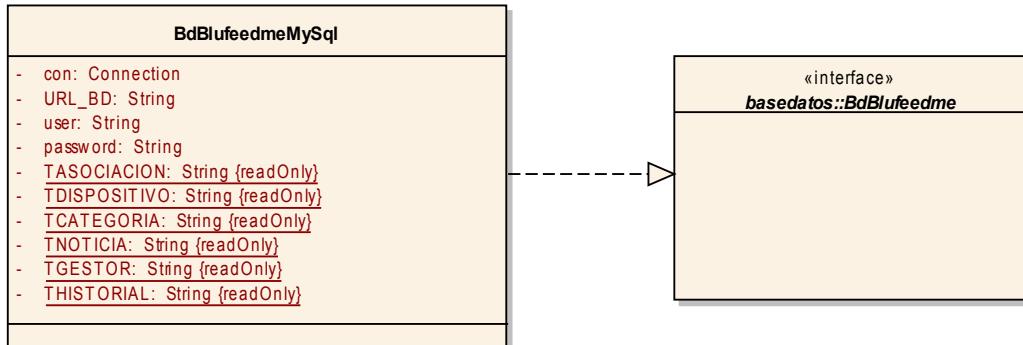


Ilustración 5.3.9: Diagrama clases paquete implementación

#### A.1.2 Capa Lógica

La capa Lógica accederá a la capa de servicios mediante el uso de la interfaz proporcionada.

En la capa Lógica se encuentra el paquete *Modelo*. Este paquete contiene las clases obtenidas en el análisis, que representan el modelo de datos. Clases usadas para representar los objetos conceptuales que existen en el dominio del problema de publicación de noticias.

Como se ha comentado la capa Lógica implementa toda la funcionalidad indicada en los casos de uso. Es aquí donde se implementa el servicio web, *NewsService*.

### 5.3 .Análisis y Diseño del sistema

---

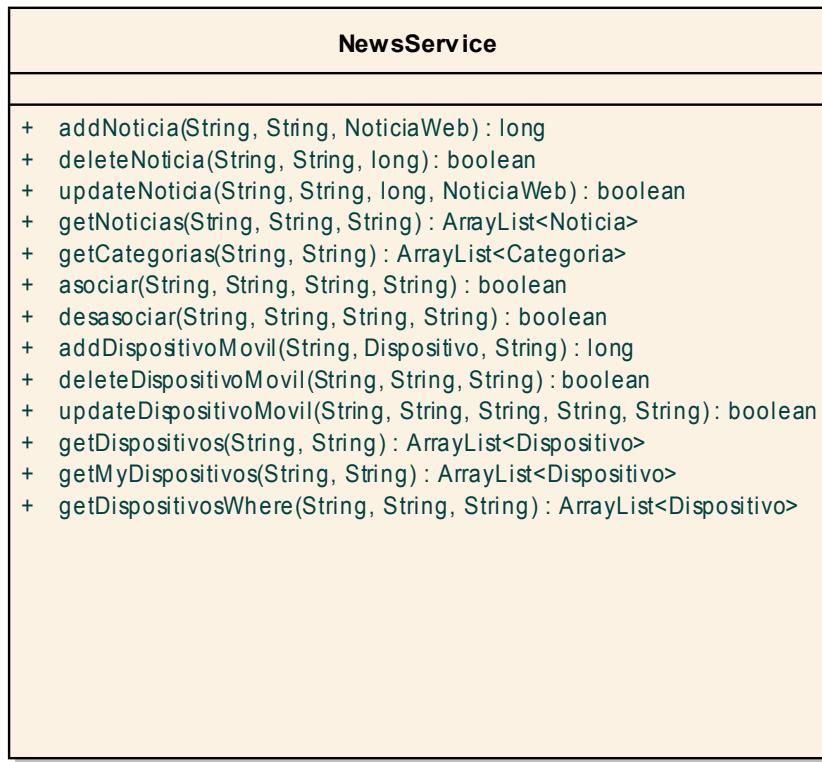


Ilustración 5.3.10: Clase NewsService

Esta clase se encuentra alojada dentro del paquete service, que, además, contiene dos paquetes adicionales con clases que son de utilidad para la implementación del Web Service. Estos paquetes contienen las clases mostradas en la figura 5.3.11. La implementación del método de encriptación SHA1 usado en todas y cada una de las operaciones del servicio web será implementadas en dichos paquetes (clase *MiSHA*).

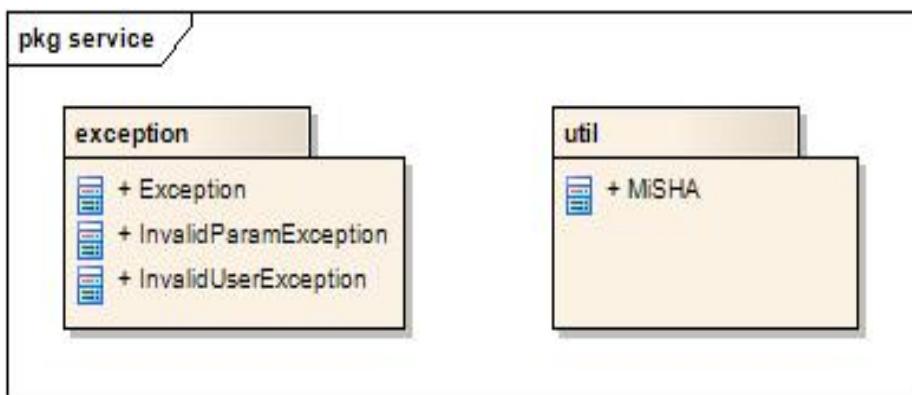
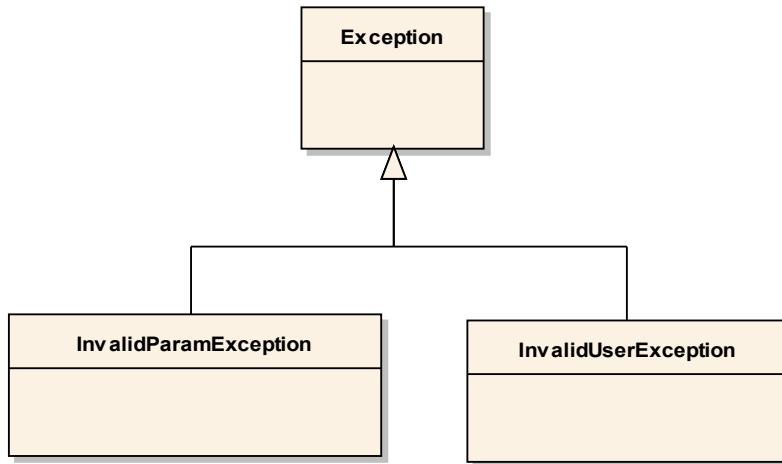


Ilustración 5.3.11: Diagrama paquetes de paquete service

### 5.3 .Análisis y Diseño del sistema

---

El diagrama de clases correspondiente al paquete exception es el que se muestra en la figura 5.3.12. Dicho paquete contiene las clases para el control de excepciones del servicio web.



*Ilustración 5.3.12: Diagrama clases paquete exception*

La descripción detallada de cada una de las clases de los diagramas mostrados anteriormente pueden consultarse en el Anexo II: Diccionario de datos análisis y diseño.

Dada la importancia de la clase que implementa el Web Service y por tanto, el conjunto de funciones de los que consta el servicio que se ofrecen a los usuarios, vamos a detallar las funciones de la clase NewsService y su empleo a través del servicio web por parte de los usuarios. La clase NewsService consta de los métodos mostrados en la figura 5.3.10.

En base a los métodos el conjunto de funciones disponibles en el servicio web son:

- ✓ **addNoticia:** añade una noticia al sistema. Los parámetros necesarios son:
  - *usuario*: usuario gestor que va a crear la noticia
  - *firma*: firma del mensaje para llevar a cabo la autentificación del usuario.
  - *noticia*: noticia a crear en el sistema.
- ✓ **DeleteNoticia:** elimina una noticia del sistema. Los parámetros necesarios son:
  - *usuario*: usuario que desea eliminar la noticia y por tanto gestor de la categoría a la que pertenece la noticia.
  - *firma*: firma del mensaje para llevar a cabo la autentificación del

usuario.

- *id\_noticia*: identificador de la noticia a eliminar.
- ✓ **UpdateNoticia**: Modifica la información relativa a una noticia del sistema. Los parámetros necesarios son:
  - *usuario*: usuario que desea modificar la noticia y por tanto gestor de la categoría de la noticia.
  - *firma*: firma del mensaje.
  - *id\_noticia*: identificador de la noticia a modificar.
  - *new\_noticia*: noticia con la nueva información actualizada.
- ✓ **getNoticias**: Obtiene las noticias correspondientes a la categoría/as que gestiona el usuario indicado. Parámetros:
  - *usuario*: usuario del que se obtendrán las noticias
  - *categoria*: categoría de la que se desea obtener las noticias.
  - *firma*: firma del mensaje.
- ✓ **getCategorias**: Obtiene las categorías gestionadas por el usuario indicado. Parámetros:
  - *usuario*: usuario del que se desean obtener las categorías.
  - *Firma*: firma del mensaje.
- ✓ **asociar**: Asocia un dispositivo a una categoría. Una asociación provoca que se envíen las noticias de la categoría al dispositivo. Parámetros:
  - *usuario*: usuario gestor de la categoría.
  - *MAC*: dirección MAC del dispositivo.
  - *categoria*: categoría a asociar.
  - *firma*: firma del mensaje.
- ✓ **desasociar**: Deshace la asociación entre un dispositivo y una categoría. Una vez desasociados el dispositivo no recibirá noticias de la categoría. Parámetros:
  - *usuario*: usuario gestor de la categoría.
  - *MAC*: dirección MAC del dispositivo.
  - *categoria*: categoría a desasociar.
  - *firma*: firma del mensaje.

- ✓ **addDispositivoMovil:** da de alta un dispositivo en el sistema.  
Parámetros:
  - *usuario*: usuario que da de alta el dispositivo. Debe ser el gestor de todas las categorías a las que está asociado el dispositivo.
  - *dispositivo*: dispositivo a dar de alta en el sistema.
  - *firma*: firma del mensaje.
  
- ✓ **deleteDispositivoMovil:** Eliminar un dispositivo del sistema. El dispositivo no puede tener ninguna categoría asociada.  
Parámetros:
  - *usuario*: usuario registrado en el sistema que elimina el dispositivo.
  - *MAC*: dirección MAC del dispositivo.
  - *firma*: firma del mensaje.
  
- ✓ **updateDispositivoMovil:** Actualiza la información relativa a un dispositivo. Parámetros:
  - *usuario*: usuario que modifica el dispositivo. Debe de ser el gestor de al menos una de las categorías a las que está asociado el móvil.
  - *MAC*: dirección MAC del dispositivo a modificar.
  - *MACN*: nueva dirección MAC del dispositivo.
  - *pinN*: nuevo número pin del dispositivo.
  - *firma*: firma del mensaje.
  
- ✓ **getDispositivos:** Obtiene los todos los dispositivos registrados en la BD. Parámetros:
  - *usuario*: usuario con el que se va a realizar la consulta.
  - *firma*: firma del mensaje enviado.
  
- ✓ **getMyDispositivos:** Obtiene los dispositivos registrados en la base de datos asociados a las categorías gestionadas por el usuario que realiza la consulta. Parámetros:
  - *usuario*: usuario con el que se realiza la consulta.
  - *firma*: firma del mensaje.

### 5.3 .Análisis y Diseño del sistema

---

- ✓ **GetDispositivosWhereCategoria:** Obtiene los dispositivos asociados a una determinada categoría. Parámetros:
  - *usuario*: usuario que va a realizar la consulta.
  - *categoria*: categoría de la que se desean obtener los dispositivos asociados.
  - *firma*: firma del mensaje.

Todos y cada uno de los métodos anteriormente descritos están disponibles para ser usados por los usuarios a través del servicio Web. Estos serán accedidos mediante un cliente del servicio web usando el archivo de descripción del servicio WSDL. La forma en la que interactúan un cliente y el Web Service se puede observar claramente en la siguiente figura:



*Ilustración 5.3.13: Comunicación Cliente Web Service-Servidor Web Service*

Donde UDDI es uno de los estándares básicos de los servicios Web cuyo objetivo es ser accedido por los mensajes SOAP y dar paso a documentos WSDL, en los que se describen los requisitos del protocolo y los formatos del mensaje solicitado para interactuar con los servicios Web del catálogo de registros. Para una descripción más detallada de los estándares y protocolos usados por esta tecnología consultar el Anexo IV: Web Service. En cuanto a la descripción WSDL y los schemas del Web Service se pueden consultar en el cdrom adjunto al presente documento.

#### A.1.3 Capa de interfaz de usuario

Para facilitar la tarea a los usuarios que quieran usar el Servicio Web en sus sistemas se facilita un cliente elaborado en PHP. El cliente se puede consultar en el código fuente contenido en el cdrom adjunto al presente documento. El cliente en PHP actuará como una clase intermedia que realizará las llamadas a las funciones del Web Service y nos trasladará los resultados

### 5.3 .Análisis y Diseño del sistema

---

devueltos a un formato más cómodo para manejar en una interfaz web. Dicha clase cliente la incluiremos en la capa de interfaz, como se puede apreciar en la figura 5.3.8, ya que no implementa nada Web Service de lógica de aplicación, sino que lleva a cabo una llamada a las operaciones del y un formateo de los datos para facilitar el trabajo a la hora de mostrarlos en una aplicación web.

En el presente proyecto se facilita una aplicación web, a modo de ejemplo práctico, que emplea el cliente PHP mencionado anteriormente. Dicha aplicación web será la interfaz gráfica que podrá usar un usuario para interactuar con el servicio sin necesidad de crear un sistema externo que interactúe con el sistema. La interfaz de esta aplicación será mostrada en el apartado Diseño: Interfaz de usuario.

#### A.2 Web Pantallas

Una vez obtenida la arquitectura software del subsistema de Web Service, procedemos a la obtención de la arquitectura para el subsistema de publicación de noticias a través de pantallas.

De nuevo, teniendo en cuenta los objetivos de diseño, extraídos de las etapas anteriores de modelado de requisitos y análisis, y las arquitecturas software disponibles, se obtiene la arquitectura software para el subsistema.

Durante el diseño del subsistema web para mostrar noticias por las pantallas de nuevo se decide usar las arquitecturas:

- Utilizar una arquitectura basada en capas cerradas que separe la aplicación en tres niveles: interfaz, lógica de aplicación y servicios (Base de Datos).
  - Razón: Mejora la estructura del sistema y fomenta la flexibilidad en cuanto a sustitución de servicios.
- Utilizar una arquitectura MVC para desacoplar la lógica de aplicación.
- Utilizar una arquitectura cliente-servidor para establecer la comunicación entre los subsistemas de la capa de lógica de aplicación y de servicios.
  - Razón: Permite distribuir los distintos subsistemas del sistema reduciendo el acoplamiento sólo a través del protocolo de comunicaciones. Fomenta la interoperabilidad, y facilita la escalabilidad del sistema.

Al igual que sucede con el subsistema Web Service, la arquitectura software está formada por tres capas diferenciadas:

- Capa Configuración: en esta capa se proporcionan los servicios de más bajo nivel. En esta capa se facilitan los servicios de acceso a base de datos.

### 5.3 .Análisis y Diseño del sistema

---

- Capa Lógica: en esta capa se implementa toda la lógica de la aplicación. La funcionalidad ofrecida por el servicio web es implementada en la capa Lógica.
- Capa Interfaz: proporciona una interfaz con la que interactuar de forma sencilla con el sistema. Esta capa estará formada por un cliente del servicio web.

Todo lo mencionado puede verse reflejado en la figura 5.3.14, Arquitectura subsistema web pantallas en capas.

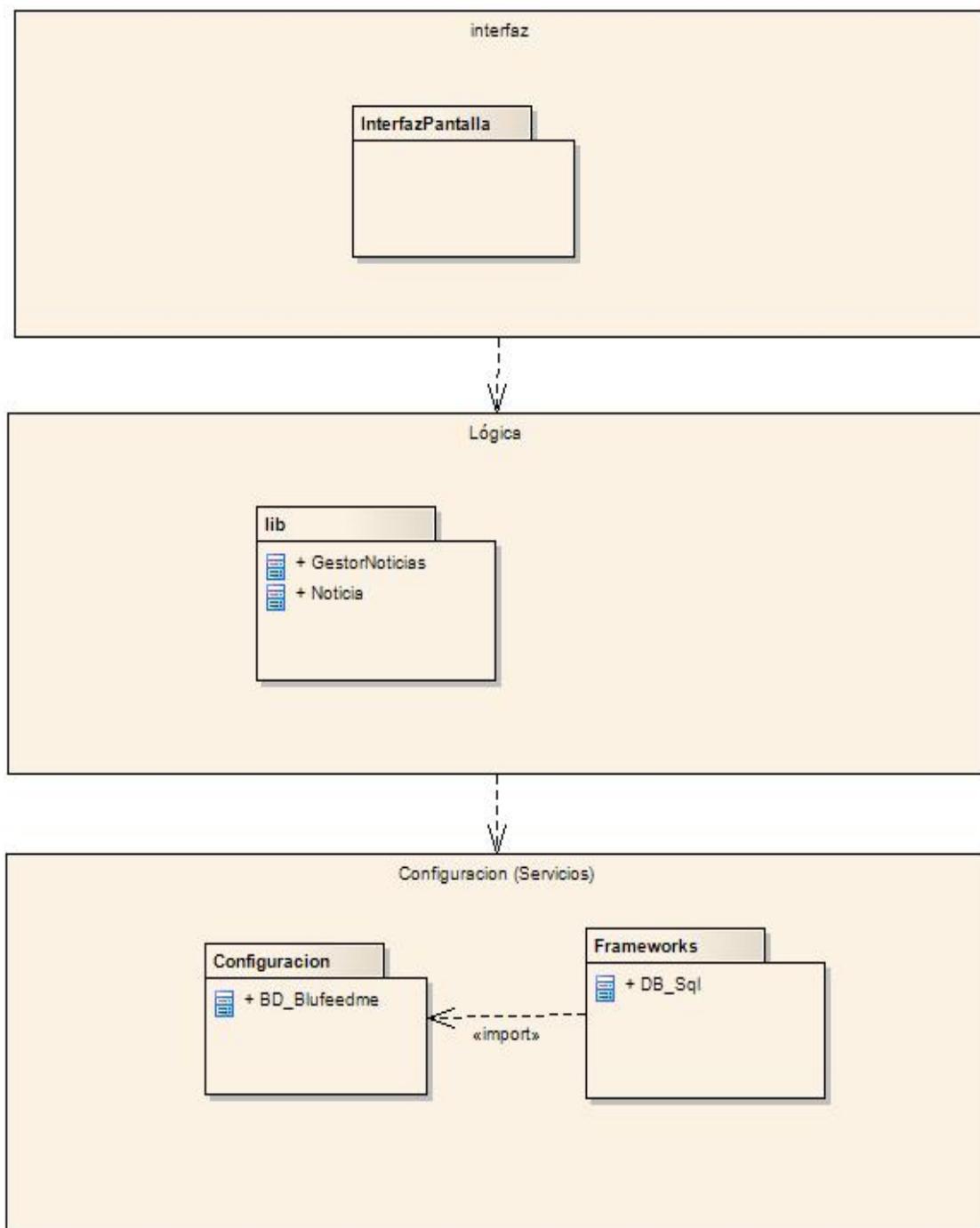


Ilustración 5.3.14: Arquitectura subsistema web pantallas en capas

## 5.3 .Análisis y Diseño del sistema

---

### A.2.1 Capa Servicios

La capa de Configuración contiene un paquete configuración que emplea un Framework de PHP para implementar el acceso a base de datos. Todos los accesos a la información disponible en la base de datos se efectúan a través de la clase *BD\_Blufeedme*, contenida en el paquete *configuracion*.

### A.2.2 Capa Lógica

En la capa de Lógica encontramos el paquete *lib*. Dentro de este paquete se encuentra la clase *GestorNoticias*. Esta clase implementa la lógica de la aplicación, empleando la clase *Noticia* para la representación de la información con la que trabaja.

### A.2.3 Capa interfaz de usuario

La capa interfaz consistiría en la implementación de la web que muestra la información obtenida de la capa Lógica. Se trata de una web dinámica por lo que actualiza su información de forma periódica. Para ello se emplean tecnologías que dotan la web de dinamismo, PHP y JavaScript.

## B) Arquitectura Física

El punto de unión entre ambos sistemas, Web Service y el sistema web de publicación por pantallas, lo proporciona la Base de Datos. La base de datos supondrá el medio de comunicación entre ambos sistemas. Al acceder ambos al mismo sistema de almacenamiento los cambios realizados por el subsistema de Web Service serán inmediatamente detectados en el subsistema de publicación de noticias. Se podría decir que la arquitectura utilizada entre el subsistema de Web Service y la Web de publicación de noticias es una arquitectura de repositorio.

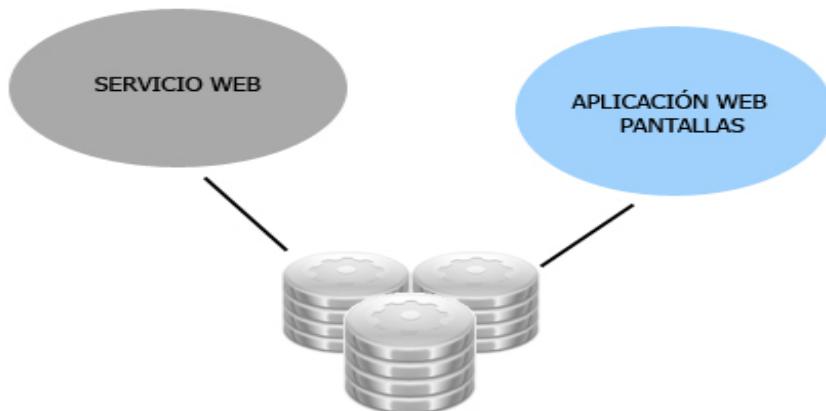


Ilustración 5.3.15: Esquema de diseño repositorio

### 5.3 .Análisis y Diseño del sistema

---

Una vez diseñadas las capas de la arquitectura de cada uno de los subsistemas que estamos estudiando en el presente apartado, vamos a determinar los posibles nodos de la parte del sistema que nos encontramos desarrollando. De esta forma obtendremos la arquitectura hardware para el sistema.

El sistema estará formado por un servidor de persistencia en el que se encontrará la base de datos en MySQL que almacenará de forma centralizada toda la información del sistema.

A dicho servidor de persistencia accederán a través de TCP/IP el servidor de aplicaciones, Glassfish, y el servidor Web, Apache.

Las pantallas colocadas por el edificio accederán mediante un navegador web al servidor Apache y mostrarán la información devuelta.

Para la arquitectura mostrada en la figura 5.3.16, se plasma un cliente del servicio web implementado con tecnologías web. Es el caso del cliente del servicio web que facilitamos en el presente proyecto. Si un sistema externo al nuestro usará tecnologías web para acceder al servicio web podría requerir el uso de un servidor Web independiente para poder ejecutar su sistema. En nuestro caso, aún apareciendo un servidor independiente para el cliente en la arquitectura hardware mostrada, se utilizará para la ejecución del cliente, elaborado en PHP, el mismo servidor en el que tenemos alojada la Web para la publicación de noticias a través de las pantallas. Claro está, los clientes del servicio web podrán estar implementados en cualquier lenguaje y sobre cualquier plataforma.

La solución propuesta, que cumple con los requisitos deseados, se representa en la figura 5.3.16.

Los elementos principales de la arquitectura propuesta son:

- **Servidor Base de Datos:** Este servidor se encontrará en una ubicación distinta al resto de servidores. De esta forma se garantiza una mayor escalabilidad del sistema y una mayor disponibilidad. Este servidor almacena de forma centralizada toda la información recopilada a través de la ejecución, por parte de los sistemas externos, de el servicio Web y la información introducida por el administrador del sistema (gestores, categorías, etc.).
- **Servidor de aplicaciones y Servidor Web:** Pueden estar situados en la misma o distinta ubicación, o pueden encontrarse ambos instalados y funcionando en el mismo servidor físico. Se ha considerado la colocación en diferentes servidores para dotar de mayor disponibilidad al sistema. Si se encontrasen en la misma ubicación un fallo en el servidor dejaría inaccesible tanto el servicio como la publicación de noticias, situación a todos los efectos indeseable. Colocándolos en ubicaciones diferentes dotaremos de mayor disponibilidad y fiabilidad al sistema, pudiendo funcionar

### 5.3 .Análisis y Diseño del sistema

ambos subsistemas de forma independiente. En el servidor de aplicaciones se encontrara desplegado el servicio web NewsService. En el servidor Web se encuentra funcionando la plataforma Web desarrollada para publicar las noticias a través de las pantallas colocadas por el edificio.

- Clientes Servicio Web (Sistemas externos): Los clientes podrán acceder al servicio empleando cualquier plataforma y cualquier lenguaje de programación. En la arquitectura mostrada se muestra un sistema externo cliente basado en tecnologías Web. Es el caso del cliente que se facilita en el presente proyecto adjunto a la documentación.
- Usuarios (con privilegios de gestión): En el caso de que los usuarios no deseen integrar el servicio web en sistemas propios existe la posibilidad de gestionar la información a través del cliente del servicio Web facilitado. Para ello podrán acceder mediante el uso de algún navegador web que soporte las tecnologías usadas. El cliente ha sido validado a través de W3C para comprobar que cumple con los estándares utilizados actualmente.
- Pantallas: consistirá en pantallas colocadas por la facultad, conectadas a un pc, que accederán al servidor web mediante http. Para ello simplemente emplearán un navegador web actual.

### 5.3 .Análisis y Diseño del sistema

---

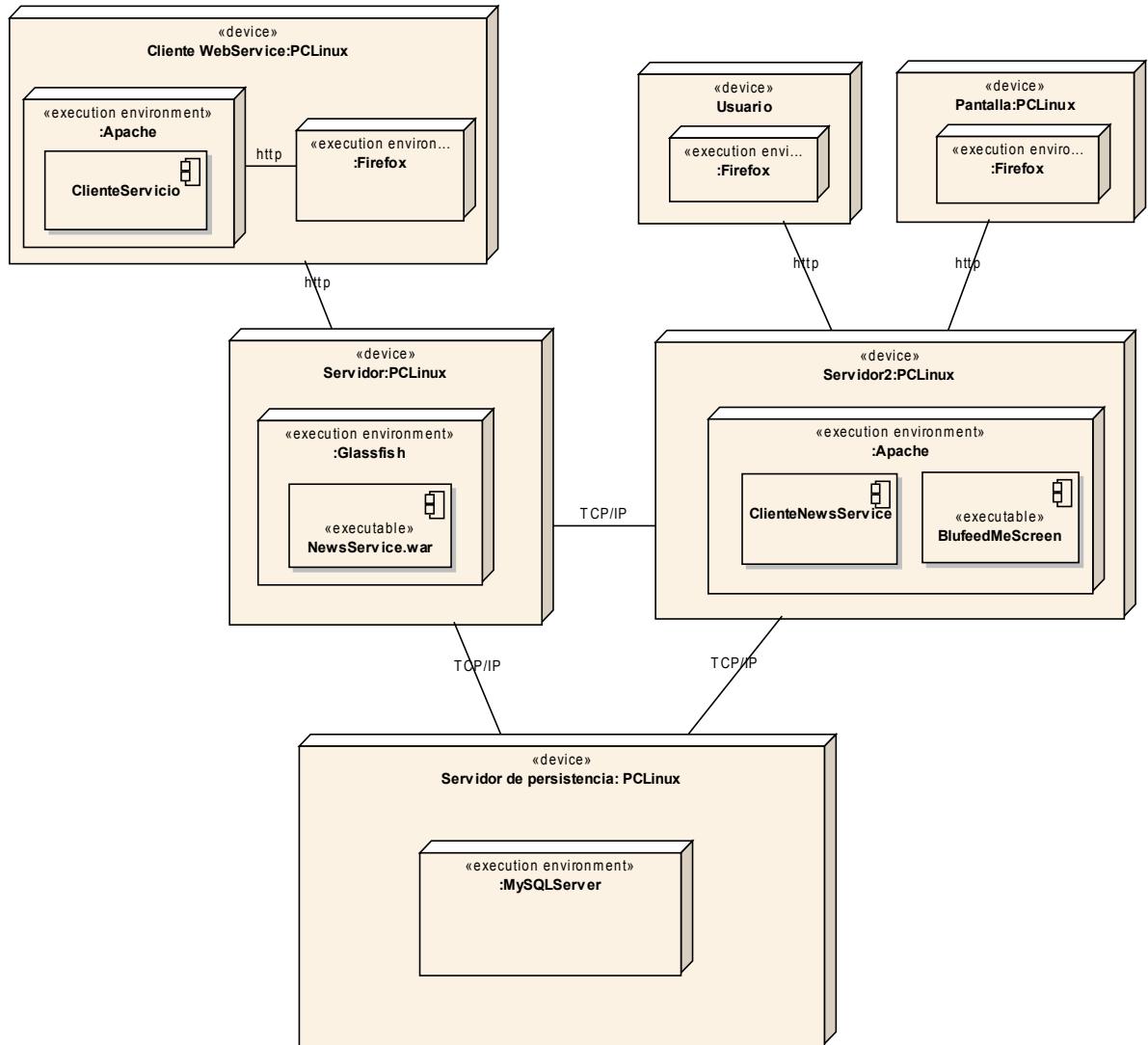


Ilustración 5.3.16: Diagrama de despliegue subsistemas Web Service y pantallas

## 5.3 .Análisis y Diseño del sistema

### 5.3.1.3 Diseño: Interfaz de usuario

La parte del sistema que estamos desarrollando consta de dos interfaces web, web para mostrar noticias por pantallas y cliente web del Web Service.

En el caso de la web para mostrar noticias por pantallas la interfaz debe de adaptarse a las pantallas disponibles. Se debe de tratar conseguir una interfaz legible, de fácil compresión e intuitiva y agradable.

Las noticias deben ir apareciendo sucesivamente y debe de existir el tiempo suficiente como para que con una velocidad de lectura razonable sea posible leer todo el contenido de la misma.

Teniendo en cuenta todo lo anterior se ha pensado en un carrusel de noticias. El carrusel irá avanzando de noticia desplegando el contenido de la misma durante un periodo de tiempo razonable para su lectura. El carrusel se adaptara a las pantallas, es decir, a la hora de desplegar noticias tendrá en cuenta la dimensión de las pantallas. Las noticias se irán desplazando y la que se encuentra en primer lugar desplegará su contenido.

Charla: "Empresas e Ingeniería del Software" 2010-11-08 23:42:15  
Subtítulo de la noticia Publicada por: jlobillo  
Charla: "Empresas e Ingeniería del Software" Ponente: Germán Sánchez Mariscal. Antiguo profesor de ETSIIT. Empresa UNIT4 Viernes día 30 de Abril a las 12 horas. Aula 0.2 de la ETSIIT Temas de los que hablará: - Uso de herramientas de Ingeniería del Software en la empresa TIC. - Estimación temporal y económica. - Gestión de Recursos Humanos y entrevistas de trabajo. - Formación de los Ingenieros Informáticos y el mercado laboral. En la charla se podrán hacer preguntas sobre el trabajo del informático en la empresa.

INTEL-EUROPA: Hands on Digital Prototyping - Summer School - 2010-11-09 00:00:00  
Solicitud de adaptación a los nuevos grados 2010-11-10 00:00:00  
Programación Web 2.0: Desarrollo Rápido de Aplicaciones con Python y Django (5a 2010-11-12 23:47:07

Ilustración 5.3.17: Interfaz Web Pantallas

Para cada noticia se muestra como información principal el título y la fecha/hora en la que la noticia fue publicada. La información que se muestra al desplegar cada noticia es el subtítulo, el autor de la noticia y todo el texto que conforma el cuerpo de la noticia.

El cliente del Web Service conforma la otra parte web del sistema. Este cliente hace uso de las funciones ofrecidas por el Web Service, proporcionando una interfaz web para interactuar con el sistema. Teniendo en cuenta todos los requisitos obtenidos en el capítulo 5.2.1 , Especificación de requisitos se diseña cada una de las interfaces necesarias para abarcar toda la funcionalidad necesaria. Se llevan a cabo validaciones de todas y cada una de las interfaces

### 5.3 .Análisis y Diseño del sistema

del subsistema cliente a través de la plataforma W3C. De esta forma se cumplirá la restricción de que el sistema sea compatible con cualquier navegador que soporte los estándares usados. Se trata de conseguir una interfaz legible, de fácil compresión e intuitiva y agradable.

Teniendo en cuenta todo lo anterior se ha diseñado una interfaz con los siguientes componentes.

Una pantalla principal con el aspecto mostrado en la figura 5.3.18, Interfaz cliente Web Service; Pantalla principal.

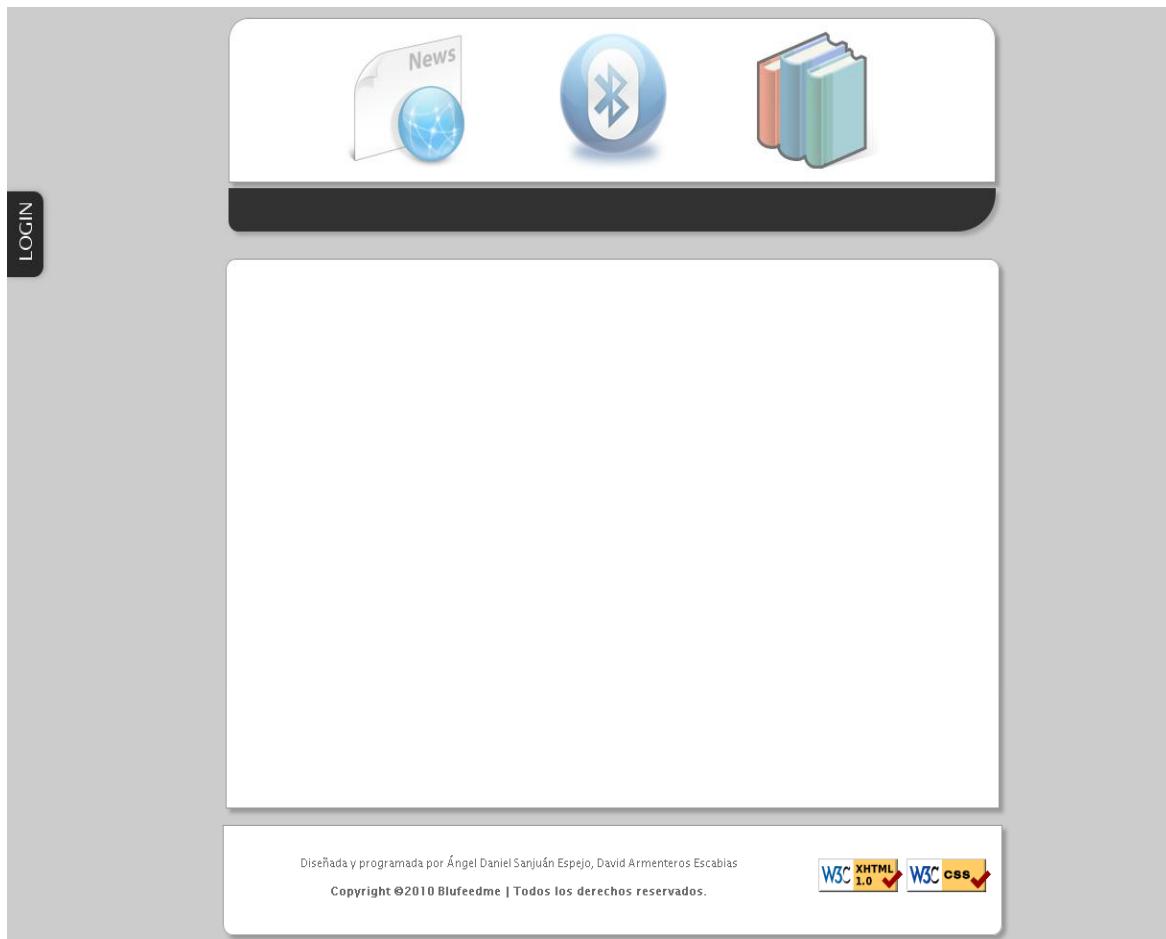


Ilustración 5.3.18: Interfaz cliente Web Service; Pantalla principal

Dispone de un panel lateral para logear. Para cada una de las operaciones que efectuamos en el Web Service es necesario autenticarse, por lo que sería muy incomodo estar constantemente introduciendo usuario y password. Una vez logeado en el sistema aparecerá debajo del menú el login del usuario y la posibilidad de cerrar sesión e iniciar sesión con un usuario diferente.

### 5.3 .Análisis y Diseño del sistema

---



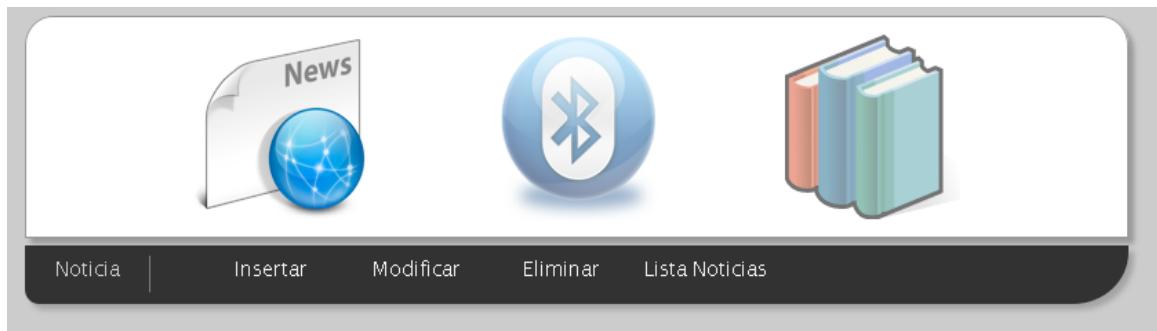
*Ilustración 5.3.19: Interfaz cliente Web Service; Login*

Dispone de un menú principal por el que navegar para llevar a cabo diferentes operaciones. El menú principal estará compuesto por iconos relacionados con las operaciones disponibles en el sistema:

- Un ícono para las operaciones sobre noticias.
- Un ícono para las operaciones sobre Bluetooth.
- Un ícono para las operaciones sobre categorías.

Para navegar por el menú simplemente bastará con pasar el cursor por encima del ícono que corresponda a las operaciones que se desean realizar y estas aparecerán en el submenú horizontal situado justo debajo de los íconos.

Dicho menú tiene la apariencia mostrada en la figura Ilustración 5.3.20: Interfaz cliente Web Service; Menú.



*Ilustración 5.3.20: Interfaz cliente Web Service; Menú*

El menú consta de un submenú en la barra inferior para concretar la operación a realizar. Cada operación tendrá un formulario en caso de ser una operación que requiera parámetros. Veamos la apariencia de la interfaz para cada una de las funcionalidades disponibles y especificadas en el apartado 5.2.1, Especificación de requisitos.

### 5.3 .Análisis y Diseño del sistema

- ◆ Noticia:

- Insertar Noticia:

The screenshot shows a web application interface for managing news. At the top, there's a navigation bar with icons for globe, user, and document, and buttons for 'Noticia', 'Insertar', 'Modificar', 'Eliminar', and 'Lista Noticias'. The user is logged in as 'gestor1' and has the option to 'cerrar sesión'. The main content area is titled 'Introducir Noticia' and contains the following fields:

- Titulo:**  (Introduce el título (必))
- Subtitulo:**  (Introduce el subtítulo)
- Autor:**  (Introduce el autor (必))
- Fecha publicación:**  (Introduce la Fecha)
- Fecha caducidad:**  (Introduce la Fecha (必))
- Categoría:**  (Introduce la Categoría (必))
- Texto:**  (Introduce el texto de la noticia (必))

A date picker calendar is displayed, showing the month of May 2011. The date '10' is highlighted, indicating it is the selected date for publication.

Ilustración 5.3.21: Interfaz cliente Web Service: Formulario Introducir Noticia

En este caso el formulario debe de disponer de los campos necesarios para poder insertar una noticia:

- Titulo: campo para poder introducir el título de la noticia.
- Subtitulo: campo para introducir el subtítulo de la noticia.
- Autor: campo para introducir el autor de la noticia.
- Fecha publicación: campo para indicar la fecha en la que se desea que la noticia sea publicada. Para facilitar la inserción de este campo se muestra un calendario, de forma que, al picar sobre el día deseado se rellena el campo de forma automática.
- Fecha caducidad: campo para indicar la fecha en la que la noticia dejará de ser publicada por los distintos medios de publicación

### 5.3 .Análisis y Diseño del sistema

disponibles en el sistema. Al igual que con el campo fecha de publicación, se facilita la tarea mostrando un calendario.

- Categoría: campo para indicar la categoría a la que será asociada la noticia. Este campo ya contiene la lista de categorías posibles a asignar en función del usuario que ha iniciado la sesión.
- Texto: es el campo para poder introducir todo el cuerpo de la noticia.
  - Modificar Noticia:

Noticia | Insertar | Modificar | Eliminar | Lista Noticias

Usuario: gestor1 | cerrar sesión

### Modificar Noticia

**ID Noticia:** 1

Introduce el identificador de la noticia a modificar (\*)

**Título:** Solicitud de adaptación a los nuevos

Introduce el nuevo título (\*)

**Subtítulo:** Subtítulo de la noticia

Introduce el nuevo subtítulo

**Autor:** Secretaría

Introduce el nuevo autor (\*)

**Fecha publicación:** 2010-11-10

Introduce la nueva fecha de publicación (\*)

**Fecha caducidad:** 2013-12-16

Introduce la nueva fecha de caducidad (\*)

**Categoría:** UGR

Introduce la nueva categoría (\*)

**Texto:**

Introduce el nuevo texto (\*)

Se ha abierto un plazo preferente para solicitar la adaptación a los nuevos grados hasta el 29 de Julio de 2010. Los alumnos de este centro interesados pueden iniciar el trámite a través de la siguiente aplicación:  
<https://etsit.ugr.es/app/static/SolicitadorDeCambioAGrado>  
E-mail de contacto: jbernier@ugr.es

...

Modificar

Ilustración 5.3.22: Interfaz cliente Web Service; Formulario Modificar Noticia

### 5.3 .Análisis y Diseño del sistema

- **Eliminar Noticia:**

Noticia | Insertar | Modificar | Eliminar | Lista Noticias

Usuario: gestor1 | cerrar sesión

### Eliminar Noticia

ID Noticia: Introduce el identificador de la noticia

1  
2  
3  
4  
5  
6

Eliminar

Ilustración 5.3.23: Interfaz cliente Web Service; Formulario Eliminar Noticia

A la hora de eliminar una noticia basta con indicar el identificador de la noticia. Una vez seleccionada la interfaz mostrará al usuario la información de la noticia que ha seleccionado para eliminar. De esta forma podrá verificar que se trata de la noticia deseada.

Noticia | Insertar | Modificar | Eliminar | Lista Noticias

### Eliminar Noticia

ID Noticia: Introduce el identificador de la noticia

1

Eliminar

**Solicitud de adaptación a los nuevos grados (ID: 1)**

**Subtitulo de la noticia ( UGR )**

Autor: Secretaría | Fecha publicación: 2010-11-10

Se ha abierto un plazo preferente para solicitar la adaptación a los nuevos grados hasta el 29 de Julio de 2010. Los alumnos de este centro interesados pueden iniciar el trámite a través de la siguiente aplicación: <https://etsit.ugr.es/app/static/SolicitadorDeCambioAGrado> E-mail de contacto: jbernier@ugr.es

Ilustración 5.3.24: Interfaz cliente Web Service; Formulario Eliminar Noticia(continuación)

## 5.3 .Análisis y Diseño del sistema

- **Lista Noticias:**



*Ilustración 5.3.25: Interfaz cliente Web Service; Lista Noticias*

Se trata de un carrusel que muestra todas las noticias. En primera instancia muestra el título y subtítulo de la noticia. Si se desea obtener más información de una de las noticias bastaría con hacer click sobre la noticia y el carrusel desplegará toda la información.

### 5.3 .Análisis y Diseño del sistema

- ◆ Bluetooth:
  - Introducir Dispositivo:

Introducir Dispositivo Bluetooth

MAC:  Introduce la MAC (")

PIN:  Introduce el PIN (")

Registrar

Dispositivos existentes en el sistema

00:23:AF:CD:2F:29

UGR (Universidad de Granada -- [www.ugr.es](http://www.ugr.es))

Secretaría E.T.S.I. (Secretaría de la Escuela Técnica Superior Ingenierías Informática y Telecomunicación -- [etsiit.ugr.es](http://etsiit.ugr.es))

TSI (Departamento de Lenguajes y Sistemas Informáticos -- [tsi.ugr.es](http://tsi.ugr.es))

Ilustración 5.3.26: Interfaz cliente Web Service; Formulario Introducir Dispositivo

El formulario muestra los campos necesarios para introducir un dispositivo móvil en el sistema:

- MAC: campo para introducir la dirección ethernet del dispositivo móvil que se va a registrar.
- PIN: campo para indicar el código que se usará para realizar el pairing en las conexiones que se establecerán entre el dispositivo y el sistema para el envío de noticias.

Además de los campos mencionados, la interfaz muestra los dispositivos existentes en el sistema (Todos independientemente del gestor). Para cada uno de los dispositivos muestra la lista de categorías a las que se encuentra asociado en ese momento.

### 5.3 .Análisis y Diseño del sistema

- Modificar:

The screenshot shows a web-based application interface for modifying a Bluetooth device. At the top, there is a navigation bar with buttons for 'Dispositivo', 'Introducir', 'Modificar', 'Eliminar', 'Asociar Categoría', and 'Desasociar Categoría'. The 'Modificar' button is highlighted. Below the navigation bar, the text 'Usuario: gestor1' and 'cerrar sesión' are displayed. The main content area is titled 'Modificar Dispositivo Bluetooth'. It contains three input fields: 'MAC:' with the value '00:23:AF:CD:2F:29', 'Nueva MAC:' (New MAC) with an empty input field, and 'Nuevo pin:' (New pin) with an empty input field. A 'Modificar' button is located at the bottom right of the form.

Ilustración 5.3.27: Interfaz cliente Web Service; Formulario Modificar Dispositivo

El formulario muestra una lista con los dispositivos registrados en el sistema sobre los que el gestor que ha iniciado sesión puede realizar operaciones. Una vez seleccionado el dispositivo bastaría con indicar los datos a modificar:

- Nueva MAC: campo para indicar la nueva dirección ethernet que se desea asignar al dispositivo móvil.
- Nuevo pin: campo para introducir el nuevo código que se usará para establecer la conexión entre el dispositivo móvil y el sistema.
- Eliminar:

The screenshot shows a web-based application interface for deleting a Bluetooth device. At the top, there is a navigation bar with buttons for 'Dispositivo', 'Introducir', 'Modificar', 'Eliminar', 'Asociar Categoría', and 'Desasociar Categoría'. The 'Eliminar' button is highlighted. Below the navigation bar, the text 'Usuario: gestor1' and 'cerrar sesión' are displayed. The main content area is titled 'Eliminar Dispositivo Bluetooth'. It contains a 'MAC:' input field with a dropdown menu showing the value '00:23:AF:CD:2F:29' and another entry '4C:54:99:06:4A:4D'. A 'Eliminar' button is located at the bottom right of the form.

Ilustración 5.3.28: Interfaz cliente Web Service; Formulario Eliminar Dispositivo

### 5.3 .Análisis y Diseño del sistema

El formulario para eliminar un dispositivo móvil del sistema muestra una lista con los dispositivos móviles existentes en el sistema que puedan ser dados de baja.

- Asociar:

The screenshot shows a web-based application interface. At the top, there is a navigation bar with links: Noticia, Insertar, Modificar, Eliminar, and Lista Noticias. To the right of these links, it shows the user is 'gestor1' and provides a 'cerrar sesión' (log out) link. Below the navigation bar, the main content area has a title 'Asociar DispositivoBluetooth-Categoría'. The form contains two dropdown menus: 'MAC' (with placeholder 'Introduce la MAC del dispositivo(")') and 'Categoría' (with placeholder 'Introduce la categoría(")'). The 'Categoría' dropdown has an item 'UGR' selected. Below the form is a light blue box titled 'Dispositivos existentes en el sistema' (Existing devices in the system) containing a list of devices. The first device in the list is '00:23:AF:CD:2F:29', followed by 'UGR (Universidad de Granada -- www.ugr.es)', and 'Secretaría E.T.S.I. (Secretaría de la Escuela Técnica Superior Ingenierías Informática y Telecomunicación -- etsiit.ugr.es)'. The 'UGR' entry is highlighted with a yellow background.

Ilustración 5.3.29: Interfaz cliente Web Service: Formulario Asociar Dispositivo

El formulario muestra la lista de dispositivos existentes en el sistema, campo MAC.

Para indicar la categoría a la que se desea asociar el dispositivo móvil la interfaz muestra una segunda lista, campo Categoría, en la que aparecen solamente las categorías que son gestionadas por el gestor con el que se ha iniciado la sesión.

Además, la interfaz muestra los dispositivos existentes asociados a categorías del gestor que va a realizar la operación. Para cada dispositivo muestra las categorías a las que está asociado actualmente.

## 5.3 .Análisis y Diseño del sistema

- Desasociar:

The screenshot shows a web-based application interface for managing device categories. At the top, there is a navigation bar with links: 'Dispositivo' (selected), 'Introducir', 'Modificar', 'Eliminar', 'Asociar Categoría', and 'Desasociar Categoría'. The user is logged in as 'gestor1'. Below the navigation bar, the main content area has a title 'Desasociar DispositivoBluetooth-Categoría'. The form contains two dropdown menus: 'MAC' (with placeholder 'Introduce la MAC del dispositivo (M)') and 'Categoría' (with placeholder 'Introduce la categoría (M)'). A 'Desasociar' (Unpair) button is located below the form. At the bottom, a section titled 'Mis dispositivos registrados' (My registered devices) lists two entries: '00:23:AF:CD:2F:29' (UGR (Universidad de Granada -- www.ugr.es)) and '4C:54:99:06:4A:4D'.

Ilustración 5.3.30: Interfaz cliente Web Service; Formulario Desasociar Dispositivo

Al igual que el caso anterior, el formulario muestra dos campos:

- MAC: lista con las direcciones ethernet de los dispositivos registrados en el sistema que están asociados a alguna de las categorías gestionadas por el gestor con el que se ha iniciado la sesión.
- Categoría: lista con las categorías a las que está asociado el dispositivo móvil seleccionado mediante el campo MAC del formulario.

Como en el resto de operaciones, se mostrarán los dispositivos registrados que están asociados a categorías del gestor y las categorías a las que se encuentra asociado el dispositivo pertenecientes, claro está, al gestor que realiza la operación.

### 5.3 .Análisis y Diseño del sistema

---

- ◆ Categoría:
  - Lista categorías:

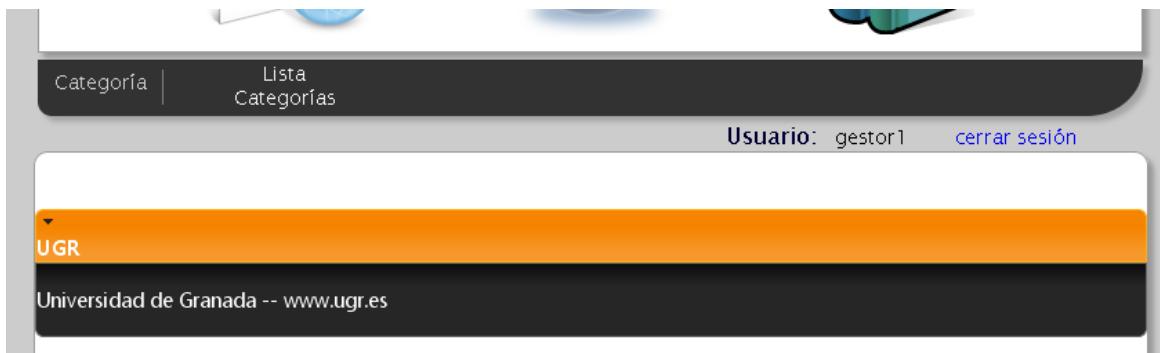


Ilustración 5.3.31: Interfaz cliente Web Service; Formulario Lista Categorías

Mostrará un carrusel con título y descripción de cada categoría gestionada por el gestor. En primera instancia se mostrarán solo los títulos de las categorías. Para poder obtener la información de la descripción de la categoría será necesario hacer click sobre la categoría deseada.

## 5.3.2 Bluetooth

Al igual que en el subsistema web, procedemos a realizar el análisis y el diseño del subsistema Bluetooth de forma separada, sin embargo, tenemos que tener en cuenta que ambos subsistemas poseen un modelo común que hay que respetar y tomar como base.

### 5.3.2.1 Análisis: Modelo estático

En el subsistema de Bluetooth nos basamos en el modelo definido anteriormente, por lo que mantenemos las clases definidas en Error: No se encuentra la fuente de referencia del subsistema Web:

- **Dispositivo:** representa a cada uno de los dispositivos registrados en el sistema, dispositivos móviles en este caso.
- **Categoría:** representa a todas aquellas categorías registradas en el sistema como tal que nos permiten agrupar las noticias de forma lógica.
- **Usuario:** representa el conjunto de usuarios que pueden hacer uso del sistema.
- **Gestor:** representa el conjunto de usuarios con la caracterización de gestor de una de las categorías.
- **Noticia:** representa el conjunto de noticias que pueden existir en el sistema.

### 5.3 .Análisis y Diseño del sistema

---

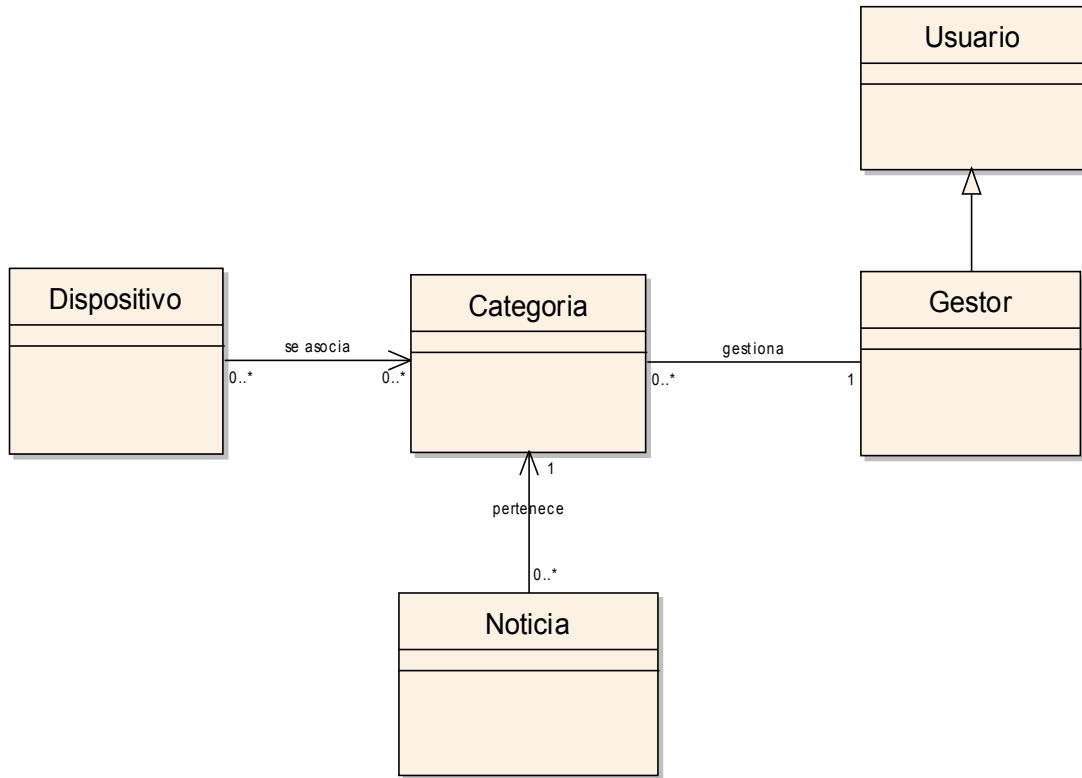
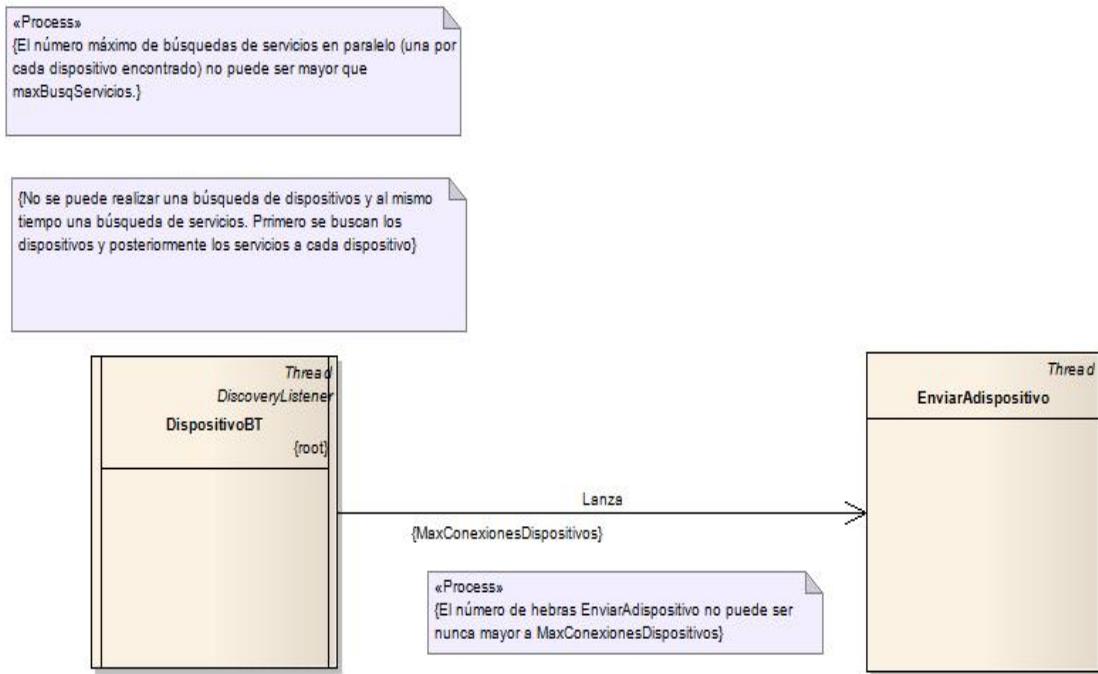


Ilustración 5.3.32: Diagrama de clases de análisis del modelo

Además se añadirán las siguientes dos clases:

- **DispositivoBT:** esta clase representa al dispositivo Bluetooth local (Hot-Spot) instalado en la máquina donde es ejecutado el programa bluetooth.
- **EnviarAdispositivo:** representa al envío de todas las noticias pendientes pertenecientes a un dispositivo móvil que ha sido detectado por el Hot-Spot.

### 5.3 .Análisis y Diseño del sistema



Ilustraci n 5.3.33: Diagrama de clases de an lisis subsistema bluetooth

Como se aprecia en la imagen superior no tenemos un gran n mero de clases, esto es debido a la simplicidad conceptual y de objetos que posee la propia aplicaci n de Bluetooth. Sin embargo, hay que tener en cuenta que estas dos clases ser n bastante complejas porque todo el peso de la aplicaci n recaer  sobre ellas, por un lado DispositivoBT que abstraer  todo el servidor Bluetooth y por otro la clase EnviarAdispositivo que ser  la respuesta del servidor a un dispositivo cliente.

La implicaci n o relaci n entre ambas clases vendr  dada por:

- **Lanza:** relaciona al dispositivo Bluetooth servidor de la aplicaci n con el env o de una respuesta (fichero) a un dispositivo cliente. La cardinalidad de la relaci n es de uno a muchos en el sentido que se indica en el diagrama.  unicamente existir  un objeto DispositivoBT y un n mero m ximo de objetos Env oAdispositivo, limitado por el n mero m ximo de conexiones soportadas por el dispositivo f sico Bluetooth Hot-Spot,

Las relaciones existentes entre el modelo de datos, la base de datos y las clases espec ficas para la aplicaci n Bluetooth est n reflejadas en el siguiente diagrama:

### 5.3 .Análisis y Diseño del sistema

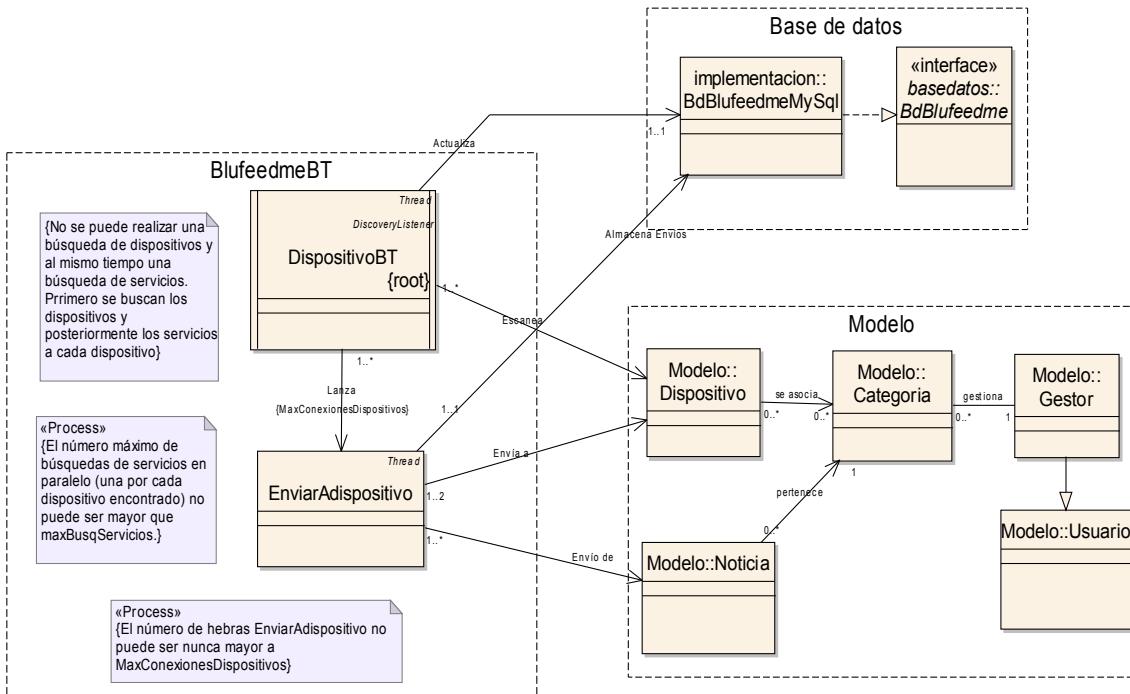


Ilustración 5.3.34: Diagrama de clases análisis Bluetooth completo

Las relaciones que aparecen son las siguientes:

- **Escanea:** esta relación refleja la detección o escaneo por parte del dispositivoBT a un dispositivo. La cardinalidad es de uno a muchos porque únicamente hay una instancia de DispositivoBT que es capaz de detectar muchos Dispositivos distintos.
- **Envía a:** esta relación muestra la abstracción de un envío de un fichero a un dispositivo móvil cliente. La multiplicidad de la relación es de uno a dos porque por cada envío están afectados dos dispositivos, uno el dispositivo de destino y el otro el dispositivo de emisión que será el Hot-Spot.
- **Envío de:** en esta relación se refleja la correspondencia entre un envío de un fichero y todas las noticias que están incluidas dentro del mismo. La multiplicidad es de uno a muchos ya que en cada envío a un dispositivo (fichero) se recogen varias noticias destinadas a él.
- **Almacena envíos:** esta relación refleja el historial del envío en la base de datos, es decir, representa el almacenamiento de las confirmaciones en la base de datos para cada envío a un dispositivo. La multiplicidad de esta relación es de uno a uno.
- **Actualiza:** esta relación refleja el acceso que realiza el DispositivoBT a la base de datos para actualizar cierta información de los dispositivos ya almacenado. La multiplicidad de la relación es de uno a uno, ya que

### 5.3 .Análisis y Diseño del sistema

únicamente disponemos de un DispositivoBT (servidor) y una base de datos.

Las clases pertenecientes a la interfaz de usuario no han sido incluidas en el diagrama de clases porque carecen de importancia desde el punto de vista analítico, sin embargo, estarán documentadas el siguiente apartado de Diseño: Arquitectura.

#### **5.3.2.2 Diseño: Arquitectura**

Al igual que en el subsistema Web (Error: No se encuentra la fuente de referencia) se decidirá por utilizar una arquitectura basada en capas cerradas con tres niveles: capa de interfaz, lógica de aplicación y capa de servicios.

- *Capa Servicios*: en esta capa se proporcionan los servicios de más bajo nivel. En esta capa se facilitan los servicios de acceso a base de datos.
- *Capa Lógica*: en esta capa se implementa toda la lógica de la aplicación, recogiendo por tanto todo el funcionamiento del servidor Bluetooth de la aplicación. Esta capa también están incluidas todas las clases pertenecientes al modelo de datos.
- *Capa Interfaz*: proporciona una interfaz gráfica al subsistema, tanto del dispositivo móvil que recibe las noticias como de la aplicación de auditoría del servidor Bluetooth.

### 5.3 .Análisis y Diseño del sistema

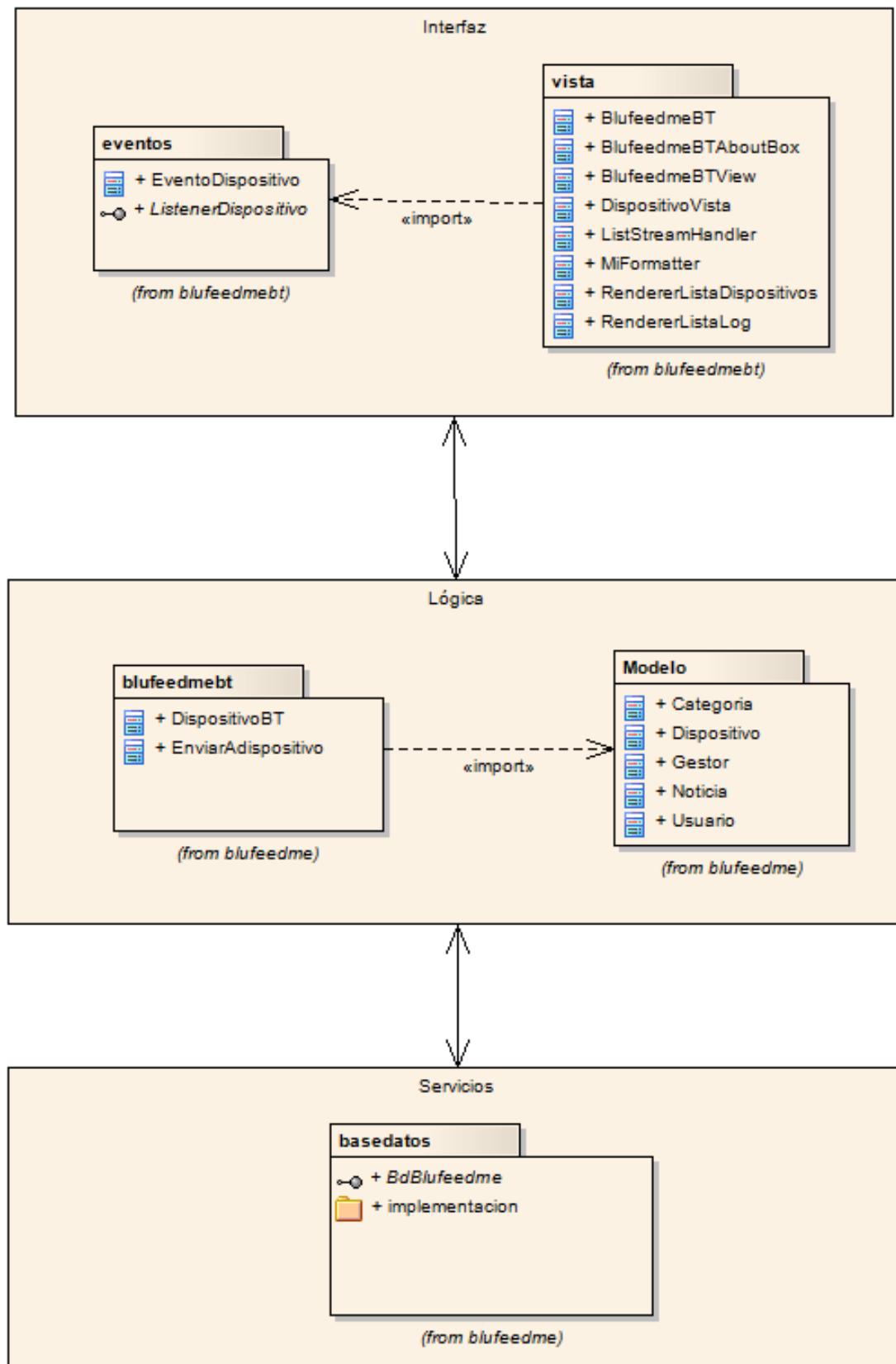


Ilustración 5.3.35: Diseño arquitectura Bluetooth

## 5.3 .Análisis y Diseño del sistema

### A) Capa de Servicios:

En la capa de Servicios disponemos del paquete *basedatos* que fue definido en el apartado Error: No se encuentra la fuente de referencia Web, en el se encuentran las clases que nos proporcionan el servicio de acceso a la base de datos de la aplicación.

### B) Capa Lógica:

En un nivel superior está la capa Lógica, está formada por las clases importadas del modelo de datos de la aplicación (definidas en el apartado anterior Error: No se encuentra la fuente de referencia Web) y por las clases propias de la aplicación Bluetooth.

Las clases DispositivoBT y EnviarAdispositivo forman parte de un paquete que se llama *blufeedmebt*, creado para separarlas del modelo de datos y seguir la arquitectura modelo-vista-controlador que fue adoptada para este subsistema.

A continuación se muestran estas dos clases con más detalle en el siguiente diagrama de clases:

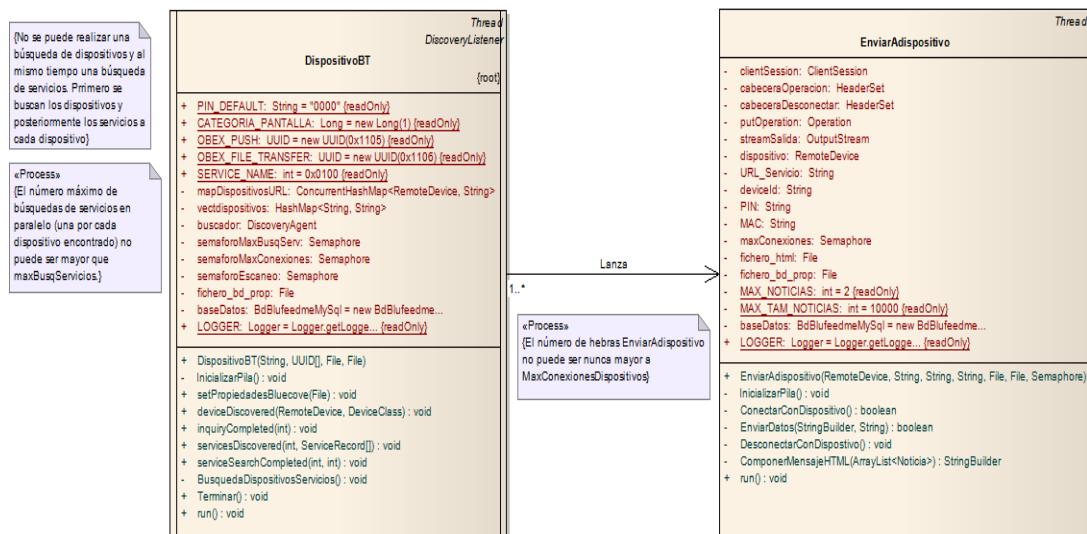


Ilustración 5.3.36: Diagrama de clases Bluetooth de diseño (parcial)

Como se observa en el diagrama, ambas clases heredan de la clase Thread, lo que nos permitirá crear distintas hebras de cada una de ellas durante la ejecución del programa. En el programa se creará únicamente una hebra del tipo DispositivoBT, correspondiente al Hot-Spot, y un número máximo simultáneo de hebras EnviarAdispositivo que vendrá dado por el número máximo de conexiones físicas que permite el Hot-Spot. En ambas clases se creará una conexión a la base de datos a través de la interfaz BdBlufeedmeMySQL definida en la capa de servicios.

### 5.3 .Análisis y Diseño del sistema

La clase DispositivoBT hereda de la interfaz DiscoveryListener que es una clase que está definida en la librería Bluecove y que todas implementa las funciones que serán ejecutadas cuando ocurran ciertos eventos en el dispositivo bluetooth, por ejemplo, finalización de la búsqueda de dispositivos, descubrimiento de dispositivos y servicios nuevos al alcance etc. La clase DispositivoBT posee tres miembros privados de tipo Semáforo (Semaphore) que están destinados para el control de concurrencia y número de hebras que serán creadas en el programa. También posee una tabla hash que está preparada para que sea accedida desde varias hebras simultáneamente, resolviendo los problemas de exclusión mutua, dicha tabla será del tipo ConcurrentHashMap y almacenará los dispositivos Bluetooth detectados y su URL de conexión.

La clase EnviarAdispositivo contiene variables pertenecientes al paquete javax.obex y que se utilizan para poder establecer una conexión OBEX entre dos dispositivos, en este caso entre el Hot-Spot y un dispositivo móvil cliente.

#### **C) Capa de interfaz de usuario:**

En la capa de interfaz se encuentran todas las clases que definen el aspecto visual de la aplicación, además también se incluyen aquellas clases que son necesarias para manejar los distintos eventos generados por el usuario que interactúe con la aplicación. A continuación se muestra un diagrama de clases con las relaciones que existen entre ellas (omitiendo la mayoría de miembros correspondientes a elementos gráficos):

### 5.3 .Análisis y Diseño del sistema

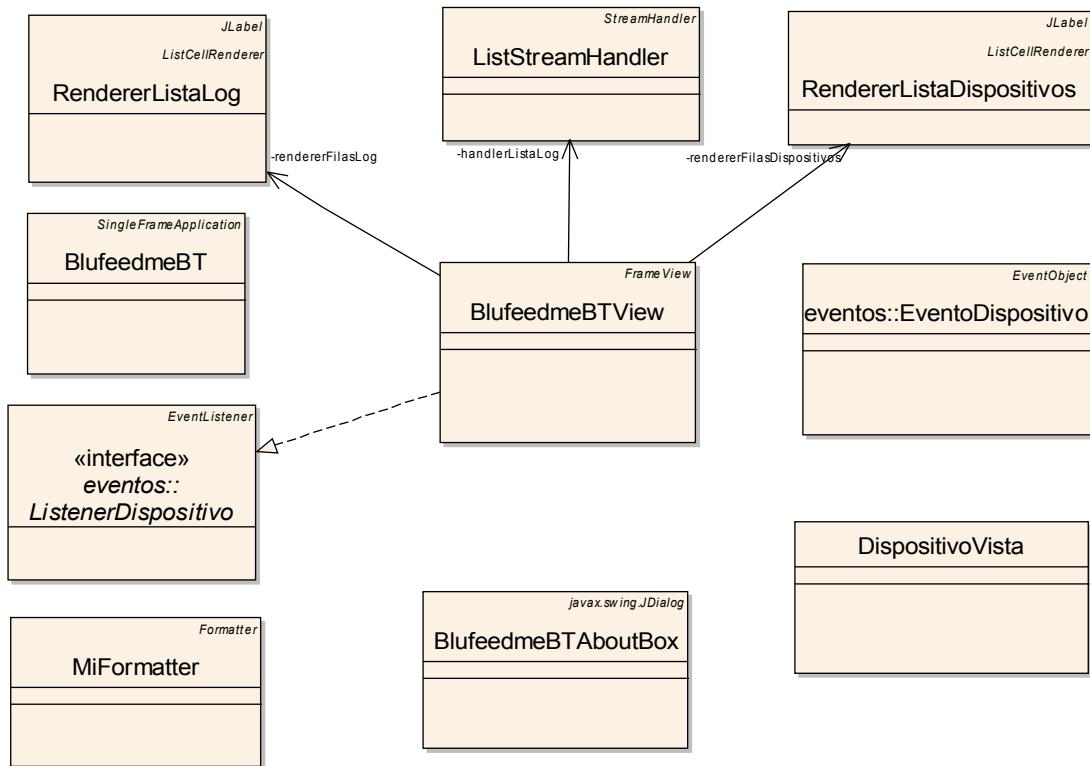


Ilustración 5.3.37: Diagrama del diseño de la interfaz gráfica bluetooth

La clase principal de la aplicación es BlufeedmeBT, que se encargará de lanzar en primer lugar la ventana gráfica BlufeedmeBTView y posteriormente iniciar la hebra DispositivoBT, perteneciente a la lógica de la aplicación.

El resto de clases se utilizan para el formateo de cadenas en la pantalla o para establecer y personalizar las distintas listas y paneles que están incluidos en la ventana principal de la aplicación. Además hay definida una interfaz que está implementada por la ventana principal para controlar los eventos de tipo EventoDispositivo. Esta clase implementa un tipo de evento personalizado que será lanzado desde la ventana principal cuando ocurran ciertas acciones y que provocará la actualización de datos en la vista.

También tenemos las clases DispositivoVista y EventoDispositivo que serán utilizadas para almacenar toda la información que será publicada en la ventana de la interfaz correspondiente a un dispositivo. Estas clases nos sirven de nexo entre las clases dispositivo, categoría y noticias del modelo y la propia interfaz gráfica, su único propósito es el de presentación de datos en pantalla.

#### D) Diseño global:

Una vez definidas una por una las distintas capas que componen el subsistema Bluetooth, mostramos las relaciones que existen entre ellas, de esta forma se puede observar de una forma global todo el diseño resultante. A continuación se muestra el diagrama de clases global del subsistema Bluetooth:

### 5.3 .Análisis y Diseño del sistema

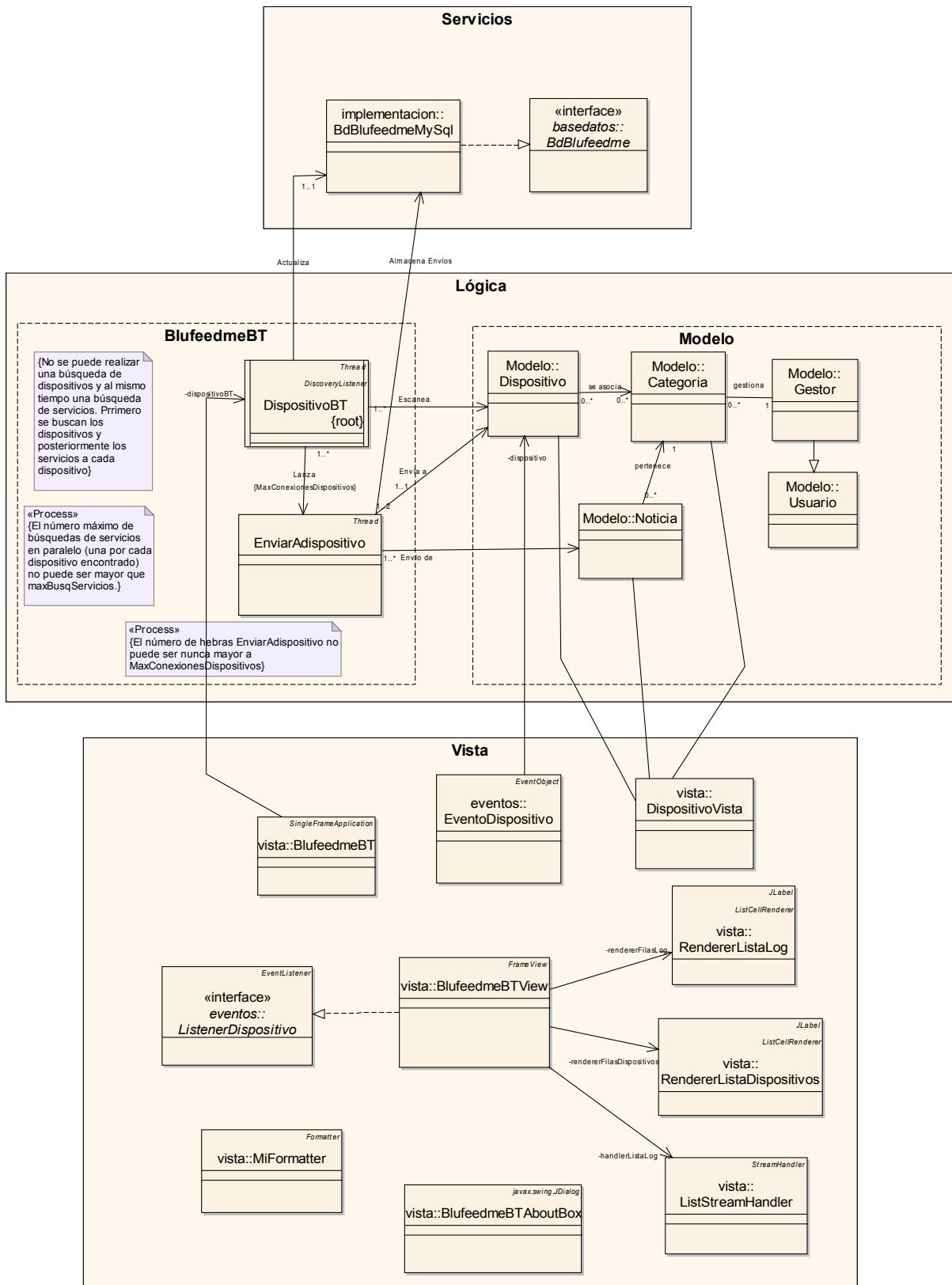


Ilustración 5.3.38: Diagrama de clases de diseño bluetooth completo

### 5.3 .Análisis y Diseño del sistema

En el diagrama podemos ver claramente las tres capas de las que está compuesta la aplicación. También observamos como la vista no se relaciona en ningún momento con la capa de servicios, sino que obtiene los datos directamente de la lógica de la aplicación y se encarga únicamente del aspecto visual de la aplicación. Del mismo modo, todas las operaciones que se realizan contra la base de datos por parte de la capa Lógica se realizan a través de la interfaz proporcionada por la capa de servicios, logrando así un desacoplamiento entre las distintas capas de la aplicación.

Todas las clases anteriores se organizarán de forma física a través de los paquetes mostrados en el siguiente diagrama de paquetes:

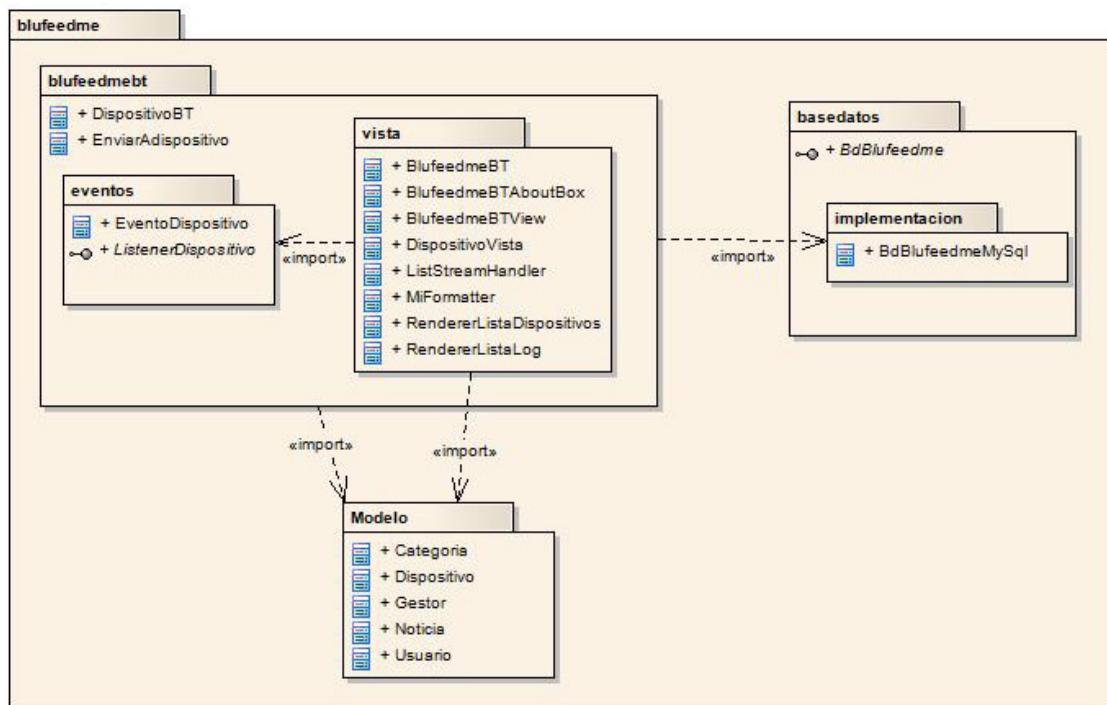


Ilustración 5.3.39: Diagrama de paquetes bluetooth

#### E) Arquitectura física:

En este apartado se nombrarán todos los elementos físicos que son necesarios para que el subsistema Bluetooth funcione. Los elementos principales de la arquitectura propuesta son:

- **Servidor Base de Datos:** Este servidor se encontrará en una ubicación distinta al resto de servidores, de esta forma se garantiza una mayor escalabilidad del sistema y una mayor disponibilidad. Este servidor almacena de forma centralizada toda la información recopilada a través de la ejecución de la aplicación Bluetooth y de la aplicación Web, por parte de los servidores externos. En él se encontrará instalado el sistema de base de datos MySql junto con

### 5.3 .Análisis y Diseño del sistema

---

la base de datos de la aplicación.

- Servidores Bluetooth: Estarán situados en la misma o distinta ubicación que el resto de servidores, aunque ya que se ha permitido su separación mediante el diseño, sería recomendable aprovechar dicha característica y separar en servidores diferentes y especializados. En cada servidor Bluetooth estará conectado un Hot-Spot o antena emisora que será por donde la aplicación se comunicará con los clientes móviles. Podrá haber más de un servidor Bluetooth funcionando simultáneamente en el sistema Blufeedme, de este modo se tendrían diferentes emisores de Bluetooth situados en distintas ubicaciones físicas. Esto es posible porque únicamente interactúan con la base de datos y son independientes entre sí. También se podría optar por añadir una serie de repetidores de señal Bluetooth para aumentar la cobertura de cada uno de los servidores Bluetooth o por una mezcla de ambos , varios servidores Bluetooth con repetidores de señal.

En el servidor Bluetooth se encontrará la aplicación blufeedmeBT junto con las librerías Bluecove para el manejo del dispositivo Bluetooth y la librería JDBC para Mysql que nos permitirá la conexión con la base de datos.

- Antena Bluetooth (Hot-Spot): Dispositivo físico que nos permitirá comunicarnos a través de señales Bluetooth con los distintos dispositivos móviles que estén al alcance. Estará conectada físicamente al servidor Bluetooth a través de la interfaz que posea dicha antena (usb, ethernet, pci-Express etc).
- Dispositivo móvil: Dispositivo que poseerán los clientes del servicio de noticias y que les permitirá recibir en dicho dispositivo las noticias.más importantes o de las categorías que le sean de interés. Este dispositivo puede ser un móvil, un tablet, una PDA, un portátil si tienen activo la opción de recibir ficheros a través de Bluetooth o cualquier otro dispositivo que soporte el protocolo OBEX PUSH a través de Bluetooth.

A continuación se muestra un diagrama de despliegue que representa la interconexión entre los distintos elementos hardware y software del subsistema Bluetooth:

### 5.3 .Análisis y Diseño del sistema

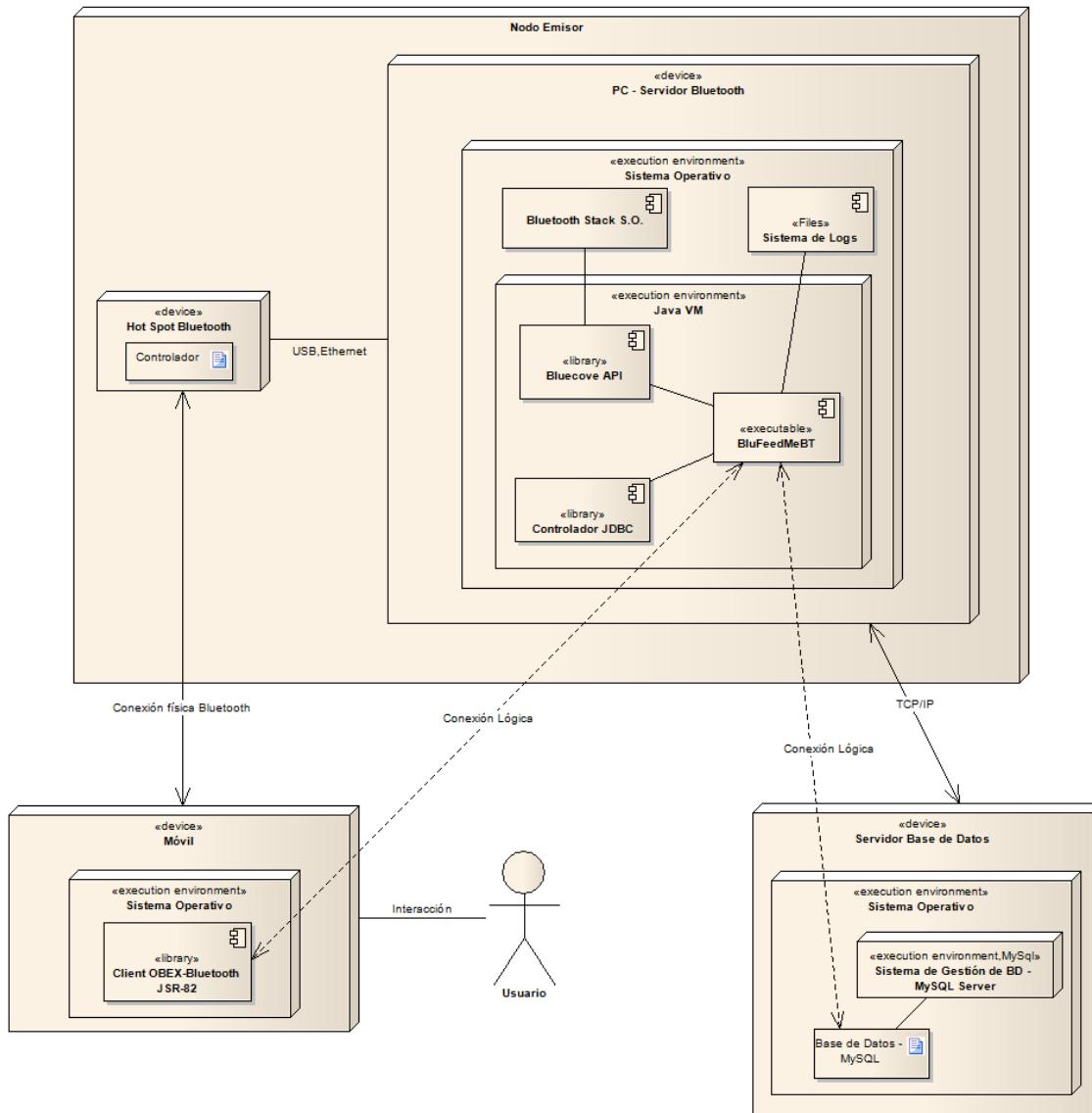


Ilustración 5.3.40: Diagrama de despliegue bluetooth

#### 5.3.2.3 Diseño: Interfaz de usuario

Teniendo en cuenta la información de los documentos previos de modelado de requisitos y análisis se obtendrá el diseño de la interfaz.

La interfaz del subsistema Bluetooth podemos dividirla en dos, por un lado la interfaz que visualizará el usuario en su teléfono móvil y por otro la que podrá visualizar el administrador del sistema que desee conocer las operaciones que está realizando el programa.

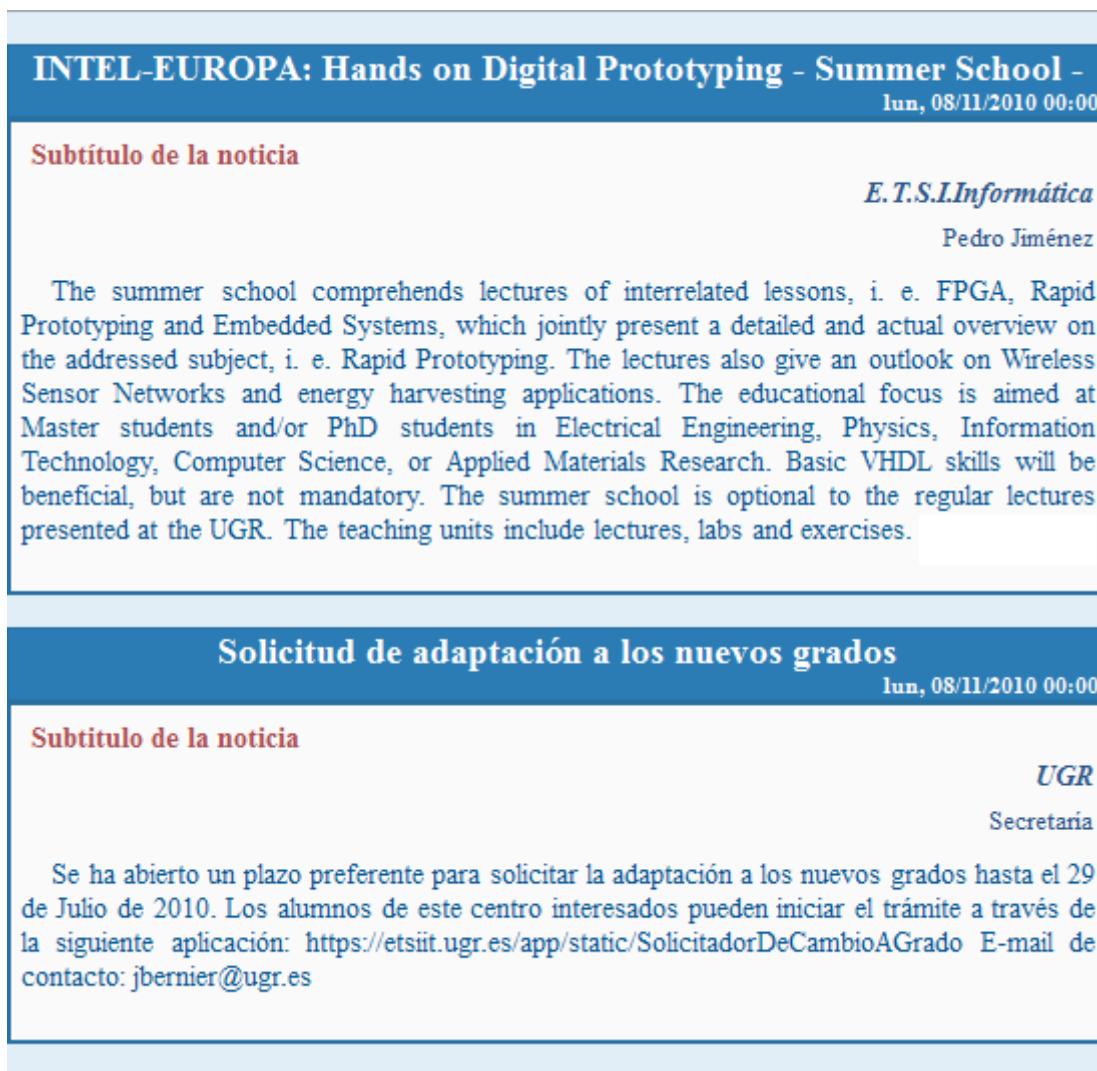
## 5.3 .Análisis y Diseño del sistema

### A) Interfaz para los dispositivos móviles

La interfaz del usuario que recibirá las noticias en su móvil estará desarrollada en xhtml. Se ha elegido dicho estándar para dar un aspecto visual más elaborado y facilitar la lectura al usuario que si de un simple texto plano se tratara. Además utilizando xhtml nos hemos asegurado una ganancia en la portabilidad de la aplicación, ya que cualquier dispositivo móvil del mercado relativamente antiguo, actual o futuro lo podrá visualizar adecuadamente.

Todo el código generado ha sido validado bajo el estándar XHTML 1.0 Strict, establecido por la W3C, para asegurarnos de que se genera un documento bien formado y por tanto sea compatible con un mayor número de dispositivos móviles.

A continuación se muestra el aspecto que tendría un listado de noticias visualizadas en cualquier dispositivo móvil:



The image displays two screenshots of mobile device screens, likely from an iPhone or similar smartphone. Both screens show news articles in a clean, modern layout.

**Top Screenshot (INTEL-EUROPA):**

- Header:** INTEL-EUROPA: Hands on Digital Prototyping - Summer School -
- Date:** lun, 08/11/2010 00:00
- Section:** Subtítulo de la noticia
- Author:** E.T.S.I Informática
- Author:** Pedro Jiménez
- Content:** The summer school comprehends lectures of interrelated lessons, i. e. FPGA, Rapid Prototyping and Embedded Systems, which jointly present a detailed and actual overview on the addressed subject, i. e. Rapid Prototyping. The lectures also give an outlook on Wireless Sensor Networks and energy harvesting applications. The educational focus is aimed at Master students and/or PhD students in Electrical Engineering, Physics, Information Technology, Computer Science, or Applied Materials Research. Basic VHDL skills will be beneficial, but are not mandatory. The summer school is optional to the regular lectures presented at the UGR. The teaching units include lectures, labs and exercises.

**Bottom Screenshot (Solicitud de adaptación a los nuevos grados):**

- Header:** Solicitud de adaptación a los nuevos grados
- Date:** lun, 08/11/2010 00:00
- Section:** Subtítulo de la noticia
- Author:** UGR
- Author:** Secretaría
- Content:** Se ha abierto un plazo preferente para solicitar la adaptación a los nuevos grados hasta el 29 de Julio de 2010. Los alumnos de este centro interesados pueden iniciar el trámite a través de la siguiente aplicación: <https://etsiit.ugr.es/app/static/SolicitadorDeCambioAGrado> E-mail de contacto: [jbernier@ugr.es](mailto:jbernier@ugr.es)

Ilustración 5.3.41: Aspecto de fichero recibido en dispositivo móvil

### 5.3 .Análisis y Diseño del sistema

---

Como se puede observar en la imagen superior, el diseño es sobrio pero de fácil lectura por el usuario. Se ha optado por un diseño sencillo que sea interpretado correctamente por la gran mayoría de navegadores, incluidos en los dispositivos móviles. Por este motivo se han utilizado únicamente las etiquetas básicas de XHTML en detrimento de un aspecto más vistoso porque de este modo lograremos una mayor compatibilidad, ya que conocemos algunas de las limitaciones de los navegadores.

Un aspecto muy importante que ha sido tenido en cuenta es la diversidad de resoluciones de pantalla de los distintos dispositivos móviles. Era necesario realizar un código que fuera capaz de visualizarse de igual modo en un móvil antiguo con una resolución pobre que en uno de última generación. Para ello hemos optado por un diseño elástico y fluido al mismo tiempo que hará que el documento ocupe la totalidad de la pantalla independientemente de la resolución de éste. Además hemos ajustado el tamaño de las distintas fuentes de forma proporcional, por lo que no habrá problemas de lectura independientemente del dispositivo.

#### **B) Interfaz de auditoría de la aplicación**

Como se comentó anteriormente, el subsistema de Bluetooth posee una interfaz destinada a las personas encargadas del mantenimiento de la aplicación. Desde esta aplicación se lanzará el proceso de escaneo y envío de noticias a los dispositivos y al mismo tiempo se mostrarán todas las operaciones que son realizadas durante el proceso. De este modo se puede realizar fácilmente una auditoría sobre el desarrollo de la aplicación y los posibles errores o advertencias encontradas.

La aplicación es iniciada a través de la siguiente pantalla:



*Ilustración 5.3.42: Pantalla ficheros de propiedades bluetooth*

### 5.3 .Análisis y Diseño del sistema

En esta ventana se solicitan tres ficheros que son de suma importancia para el funcionamiento de la aplicación, como son:

- fichero de configuración de la base de datos: en este fichero se define el nombre u ruta de la base de datos, el usuario con el que accederá la aplicación y su contraseña.
- fichero de configuración de la pila Bluetooth: en este fichero se indican las propiedades de la pila Bluetooth, como son los distintos timeouts de conexión, el número máximo de conexiones simultáneas etc.
- fichero de plantilla para el envío de noticias: en este fichero se almacena la plantilla html que servirá como base para la generación del fichero de noticias para los dispositivos.

La siguiente ventana será donde se reflejarán todos los dispositivos que están siendo escaneados en tiempo real por la antena Bluetooth y la información proporcionada por la base de datos.

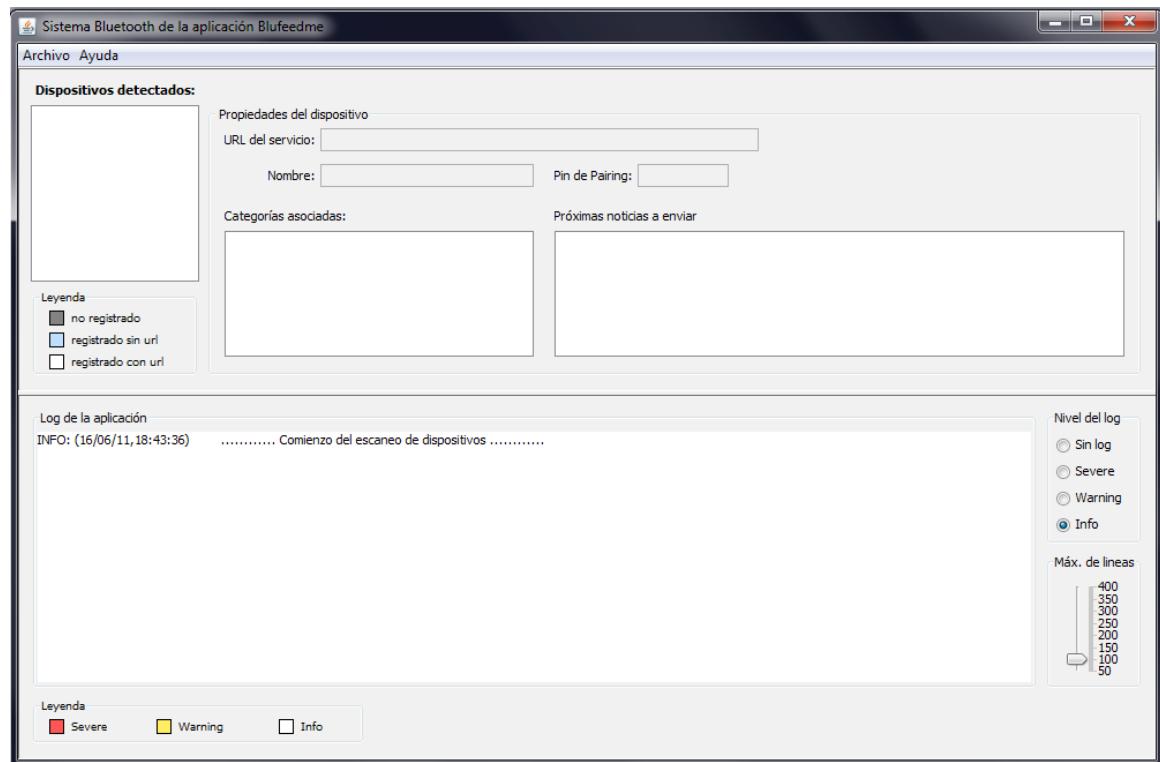


Ilustración 5.3.43: Pantalla de auditoría del proceso de envío bluetooth

La ventana la podemos dividir en dos zonas o secciones, una superior que se encarga de mostrar un listado en tiempo real de los dispositivos que están siendo detectados mediante Bluetooth y otra inferior donde se muestra un Log de la aplicación con todas las operaciones que se están ejecutando.

### 5.3 .Análisis y Diseño del sistema

---

- En la sección superior aparece un listado con las direcciones MAC de cada uno de los dispositivos detectados. Cada línea estará coloreada con alguno de los tres colores indicados en la leyenda (gris, azul, blanco) dependiendo del modo en el que esté el dispositivo almacenado en la base de datos:
  - Gris: Dispositivo no registrado en la base de datos. Se le enviarán únicamente las noticias más importantes ya que no posee categoría de noticias asignadas.
  - Azul: Dispositivo registrado pero sin URL de conexión conocida. Se trata de un dispositivo cuya MAC está registrada en el sistema, sin embargo, no se conoce la URL de conexión para el servicio OBEX PUSH por lo que tendrá que realizarse una búsqueda de sus servicios disponibles.
  - Blanco: Dispositivo registrado y con URL de conexión conocida. Se trata de un dispositivo registrado en el sistema y del que además poseemos la URL del servicio OBEX PUSH.

A la derecha del listado aparecen una serie de campos que nos mostrarán todos los datos relativos al dispositivo seleccionado en la lista de la izquierda. Se mostrará el nombre del dispositivo (en caso de que posea), el URL de conexión (si ya disponemos de él), el PIN de conexión (es 0000 para dispositivos no registrados) y un listado con las categorías asociadas junto con las noticias que están pendientes de envío al dispositivo.

- En la sección inferior aparece una lista con la traza de todas las operaciones del sistema. En dicha lista se irán añadiendo líneas conforme vayan sucediendo eventos en el sistema . Cada una de las líneas poseerá uno de los colores que aparecen en la leyenda inferior, indicando así la gravedad del evento. Los eventos en amarillo indican que ha habido algún tipo de inconveniente pero que no posee excesiva importancia, por ejemplo, se ha cumplido algún timeout para una transacción o habido un error al obtener el nombre del dispositivo. Los eventos en rojo mostrarán los errores importantes como puede ser un fallo durante el envío del fichero de noticias, sin embargo, el sistema tendrá en cuenta dicho fallo y retransmitirá aquellos ficheros que no se hayan completado satisfactoriamente.

Las opciones de la derecha sirven para filtrar o modificar la visualización de la lista de eventos, indicando el número máximo de líneas almacenadas o mostrando únicamente aquellos eventos que posean un nivel de prioridad por debajo del indicado.

A parte del listado de eventos parcial obtenido en pantalla, recordamos que todas las operaciones y eventos serán almacenadas

## 5.3 .Análisis y Diseño del sistema

simultáneamente de forma permanente en los ficheros de logs (en formato xml) establecidos por la aplicación.

En la ventana principal de la aplicación ha sido añadida una barra de tareas en la que podremos elegir la opción de salir de la aplicación, y por tanto detener el proceso de envío, o mostrar un diálogo de información sobre la aplicación, la licencia bajo la que se rige y las distintas fuentes de información en Internet sobre la aplicación. A continuación se muestra dicha ventana:



Ilustración 5.3.44: Diálogo información de la aplicación bluetooth

### 5.3.3 Base de Datos

Una de las partes fundamentales del sistema de publicación de noticias, si no la más importante, es la Base de Datos. Como se comentó en el apartado 3, Solución para el proyecto se decide dotar al sistema de una base de datos centralizada en la que se recoge la información de gestión del sistema: noticias, categorías, dispositivos móviles, gestores... Por ello, un buen diseño de la Base de Datos es fundamental para garantizar el buen funcionamiento del sistema y mantener la integridad de los datos que se almacenarán en ella, así como sus interrelaciones.

#### 5.3.3.1 Modelización semántica

La modelización semántica trata de dotar la información que tienen los sistemas de base de datos de algo más que la comprensión limitada que ofrece un simple almacenamiento de datos en sí. Una de las tareas básicas de la modelización semántica es la estructuración de la información que se va a almacenar y manejar.

Para describir con claridad esta estructura, es necesario definir previamente el concepto de modelo de datos.

Los modelos de datos pueden clasificarse de distintas formas atendiendo

### 5.3 .Análisis y Diseño del sistema

---

a diferentes criterios. Usando una clasificación basada en el nivel de abstracción al que se manejan los datos, los modelos se clasifican en *modelos de datos lógicos* y *modelos de datos implementables*.

- Modelos de datos lógicos. Se usan para describir datos a nivel conceptual y externo y se caracterizan porque tienen una amplia capacidad expresiva, son muy flexibles y permiten especificar ciertas restricciones de integridad.
  - El modelo entidad-relación.
  - El modelo orientado a objetos.
- Modelos de datos implementables. Se usan para describir modelos de datos a nivel conceptual y físico y permiten especificar la estructura lógica global de la base de datos, así como una descripción a un nivel más bajo, el de implementación, entrando en detalles de almacenamiento, operadores empleados, restricciones de integridad genéricas, etc., que solo pueden ser plasmados de forma parcial en los modelos lógicos.
  - El modelo jerárquico.
  - El modelo en red.
  - El modelo relacional.

Para el presente proyecto usamos el modelo de datos lógico entidad-relación y el modelo de datos implementables relacional dada su difusión y repercusión en las bases de datos actuales.

#### A) Diseño conceptual: Modelo Entidad-Relación

El modelo Entidad-Relación es el modelo más extendido para el diseño conceptual de una base de datos dada su capacidad expresiva. Mediante esta técnica se identificarán los objetos de interés del proyecto (entidades), las propiedades relevantes de dichos objetos (atributos) y cómo éstos se relacionan entre sí (relaciones).

El diagrama Entidad-Relación para el presente proyecto es el que se muestra en la figura 5.3.45.

## 5.3 .Análisis y Diseño del sistema

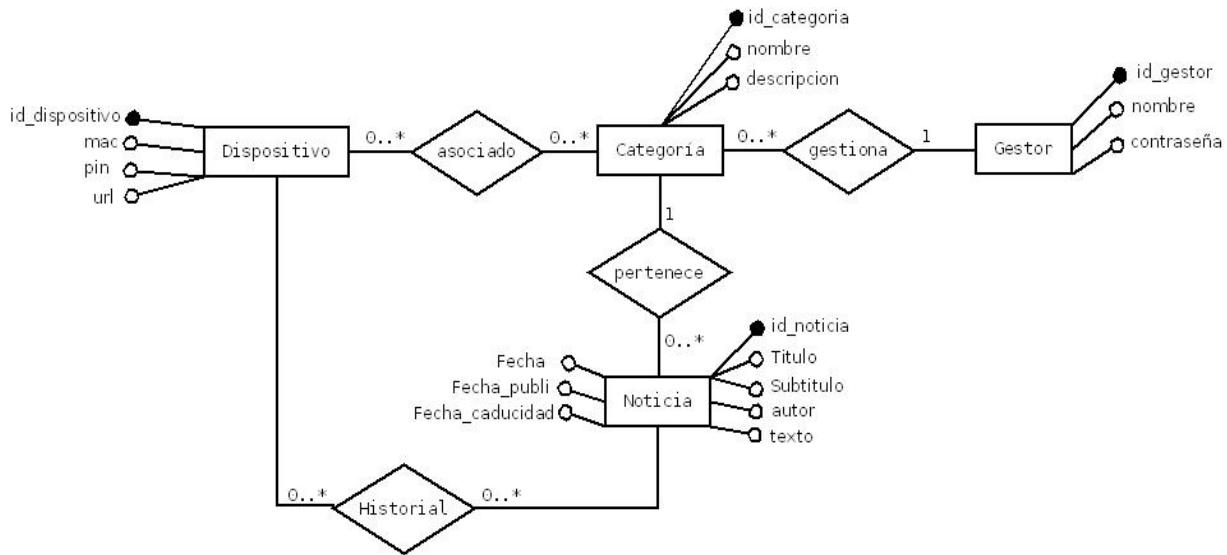


Ilustración 5.3.45: Base de Datos; Diagrama Entidad-Relación

La descripción detallada de cada una de las entidades, cada uno de los atributos de las mismas y las relaciones entre ellas, de la Base de Datos reflejada en el diagrama Entidad-Relación, pueden consultarse en Anexo III: Diccionario de datos Base de Datos..

### B) Modelo de datos implementable: Del modelo E-R al modelo Relacional

Una vez realizado el análisis de los datos que conformarán la base de datos siguiendo el modelo de datos lógico Entidad-Relación, el siguiente paso es la elección del modelo de datos implementable en el que traducir las estructuras empleadas en el primero, que en este caso es un diagrama E-R. Por las razones expuestas en el apartado anterior el modelo implementable elegido es el modelo relacional.

El paso del modelo E-R al Relacional de las entidades existentes es bastante directo, ya que éste se traduce directamente a una tabla en la que se conservan los atributos y la clave primaria sigue siendo la misma que la del conjunto original.

Del diagrama E-R tenemos las siguientes tablas resultantes de las entidades:

- ✓ DISPOSITIVO([Id\\_Dispositivo](#), MAC, Pin, URL).
- ✓ GESTOR([Id\\_Gestor](#), Nombre, Contraseña).
- ✓ CATEGORIA([Id\\_Categoría](#), Nombre, Descripcion).
- ✓ NOTICIA([Id\\_Noticia](#), Titulo, Subtitulo, Autor, Texto,

### 5.3 .Análisis y Diseño del sistema

---

`Fecha, Fecha_Publi, Fecha_Caducidad)`

Las relaciones entre las entidades también deben ser transformadas a tablas. En este caso, la tabla generada tendrá como atributos las claves primarias de las entidades que se conectan junto con los atributos propios de la conexión. La elección de la clave primaria para la tabla construida no es tan directa y depende de la cardinalidad de la conexión original.

El conjunto de tablas resultado de relaciones existentes entre las entidades anteriormente mencionadas es:

- ✗ **ASOCIADO**(Id\_Dispositivo,Id\_Categoría)
- ✗ **GESTIONA**(Id\_Categoría, Id\_Gestor)
- ✗ **PERTENECE**(Id\_Noticia, Id\_Categoría)
- ✗ **HISTORIAL**(Id\_Dispositivo, Id\_Noticia)

#### C) Mejoras en el esquema obtenido

Una vez obtenido el conjunto de tablas es necesario preguntarse, ¿es el mejor posible para capturar datos y las restricciones? La forma de mejorar un esquema es reduciendo el número de tablas fusionándolas de dos en dos teniendo en cuenta que es **condición necesaria** que ambas tengan la misma clave primaria. Teniendo en cuenta esta condición y analizando la conveniencia o no de la fusión , en función del espacio ocupado/desocupado, se efectúa la optimización del esquema obtenido en el apartado anterior:

- ✓ **DISPOSITIVO**(Id\_Dispositivo, MAC, Pin, URL) .
- ✓ **GESTOR**(Id\_Gestor, Nombre, Contraseña) .
- ✓ **CATEGORIA**(Id\_Categoría, Nombre, Descripcion) .
- ✓ **NOTICIA**(Id\_Noticia, Titulo, Subtitulo, Autor, Texto, Fecha, Fecha\_Publi, Fecha\_Caducidad)
- ✓ **ASOCIADO**(Id\_Dispositivo,Id\_Categoría)
- ✓ **GESTIONA**(Id\_Categoría, Id\_Gestor)
- ✓ **PERTENECE**(Id\_Noticia, Id\_Categoría)
- ✓ **HISTORIAL**(Id\_Dispositivo, Id\_Noticia)

Observando el esquema salta a la vista dos posibles fusiones: CATEGORIA Y GESTIONA, y NOTICIA y PERTENECE. Con la fusión ahorraríamos espacio ya que toda noticia debe de estar asociada a una categoría y una categoría a un gestor, por tanto el campo siempre tendría un

### 5.3 .Análisis y Diseño del sistema

---

valor distinto de NULL. Tras la mejora el esquema es:

- ✓ DISPOSITIVO(Id\_Dispositivo, MAC, Pin, URL).
- ✓ GESTOR(Id\_Gestor, Nombre, Contraseña).
- ✓ CATEGORIA(Id\_Categoría, Nombre, Descripcion, id\_gestor).
- ✓ NOTICIA(Id\_Noticia, Titulo, Subtitulo, Autor, Texto, Fecha, Fecha\_Publi, Fecha\_Caducidad, id\_categoria)
- ✓ ASOCIADO(Id\_Dispositivo,Id\_Categoría)
- ✓ HISTORIAL(Id\_Dispositivo, Id\_Noticia)

El anterior esquema conforma el conjunto de tablas existentes en la Base de Datos para el proyecto BLUFEEDME.

## 6 DETALLES DE LA IMPLEMENTACIÓN

---

En este capítulo se muestran los aspectos más destacables de la implementación de cada uno de los subsistemas: Web Service, cliente del Web Service, Web para publicación por pantallas y subsistema Bluetooth para el envío de noticias a dispositivos móviles.

### 6.1 *Subsistema Gestión Web Service y cliente Web Service*

#### 6.1.1 Web Service

En esta sección vamos a tratar de describir los aspectos más relevantes de la implementación del servicio Web y del cliente del mismo. El lenguaje usado para la implementación del Web Service es JAVA. Como plataforma de desarrollo se ha empleado el IDE Netbeans. Este entorno facilita la tarea de creación de servicios web y su despliegue. Integra en el entorno el servidor de aplicaciones empleado, Glassfish, y genera todos los archivos necesarios para la puesta en marcha del servicio Web, WSDL, schemas, etc. Con Netbeans se obtienen de forma sencilla el archivo .war, cuyas características pueden consultarse en el Anexo IV: Web Service., para posteriormente desplegarlo en el servidor de aplicaciones.

El servicio web lleva a cabo la gestión de la información generada en el sistema. Para almacenar toda esa información, el sistema cuenta con una base de datos centralizada sobre MySQL. Para establecer las conexiones se emplea la API JDBC (Java Database Connectivity). Mediante esta API se realizan las operaciones sobre la base de datos, independientemente del sistema operativo donde se ejecute el servicio Web ó de la base de datos a la cual se accede, utilizando el dialecto de datos que se utilice.

El API JDBC se presenta como una colección de interfaces Java y métodos de gestión de manejadores de conexión hacia cada modelo específico de base de datos. En este caso disponemos de una base de datos MySQL, por lo que se usa un manejador de conexiones para esa base de datos, concretamente *mysql-connector-java-5.1.13-bin.jar*.

Se dispone de una interfaz que, empleando la API anteriormente citada, proporciona métodos de acceso a la Base de Datos. Esta interfaz es *BdBlufeedme* y se pueden consultar los métodos en el código fuente o la documentación JAVADOC en el cdrom adjunto al presente documento.

La clase que implementa el servicio web, *NewsService.java*, cuenta con un atributo de tipo *BdBlufeedmeMySql*. Esta clase implementa la interfaz *BdBlufeedme*.

## 6.1 .Subsistema Gestión Web Service y cliente Web Service

---

```
BdBlufeedmeMySql bd = new BdBlufeedmeMySql();
```

La clase *BdBlufeedmeMySQL* cuenta con un atributo de tipo *Connection* y con los atributos necesarios para almacenar la información requerida a la hora de establecer una comunicación con la Base de Datos: dirección url, usuario y contraseña.

```
public class BdBlufeedmeMySql implements
blufeedme.basedatos.BdBlufeedme{
    private Connection con;
    private String URL_BD;
    private String user;
    private String password;
    ...
}
```

Para indicar la información de conexión se emplea un archivo de configuración. De esta forma podremos modificar la información sin necesidad de acceder al código fuente. El archivo de configuración debe de tener el siguiente formato:

```
URL:jdbc:mysql://localhost/blufeedme_db
user:root_blufeedme
password:1234
```

Donde *URL* indicará la dirección donde se encuentra el servidor de base de datos, *user* y *password* indican el usuario/password a usar para conectar con la BD.

Para manejar la información extraída de la base de datos, y necesaria durante el transcurso de las operaciones ofrecidas por el servicio web, se utilizan las clases detalladas en el diseño estático, capítulo 5.3 , Análisis y Diseño del sistema.

Todas y cada una de las operaciones disponibles en el servicio web llevan a cabo la autenticación del usuario mediante el mecanismo de firma digital de mensaje. Para realizar la firma digital se emplea un algoritmo basado en el empleo de funciones resumen o compendio, ya que entre las características de

## 6.1 .Subsistema Gestión Web Service y cliente Web Service

---

estas funciones cabe destacar:

- Son de cálculo sencillo.
- Proporcionan mensajes de salida de longitud fija, independientemente de la longitud del mensaje de entrada.
- Dados dos mensajes de entrada distintos, los resúmenes correspondientes son también distintos.
- Desde el punto de vista de la seguridad, resulta imposible para un ataque obtener un mensaje a partir de su compendio.

Como función resumen o compendio se utiliza SHA-1 (Secure Hash Algorithmin-1), propuesta por el NIST (National Institute of Standars and Technology) en 1993.

Los procesos puestos en marcha por el módulo SHA-1 en sí, son los siguientes (véase figura XXX):

- a) Se lleva a cabo un procesamiento en cuatro rondas sucesivas, de 20 pasos cada una, en las que se utilizan, respectivamente, las funciones  $f_1$ ,  $f_2$ ,  $f_3$  y  $f_4$  con argumentos:

$$X = ABCDE$$

$$Y = Y_q$$

$$K_w = 5A827999, \quad 0 \leq W \leq 19$$

$$6ED9EBA1, \quad 20 \leq W \leq 39$$

$$8F1BBCDC, \quad 40 \leq W \leq 59$$

$$CA62C1D6, \quad 60 \leq W \leq 79$$

- b) Completadas las cuatro rondas, se realiza una operación suma módulo  $2^{32}$
- c) La salida obtenida será la entrada para el procesamiento del siguiente bloque. Si  $q = L$ , lo que corresponde al último bloque, el resultado será el compendio del mensaje  $SHA(P) = SHA_{L+1}$ .

Para más detalles se puede consultar el sitio web <http://csrc.nist.gov>.

## 6.1 .Subsistema Gestión Web Service y cliente Web Service

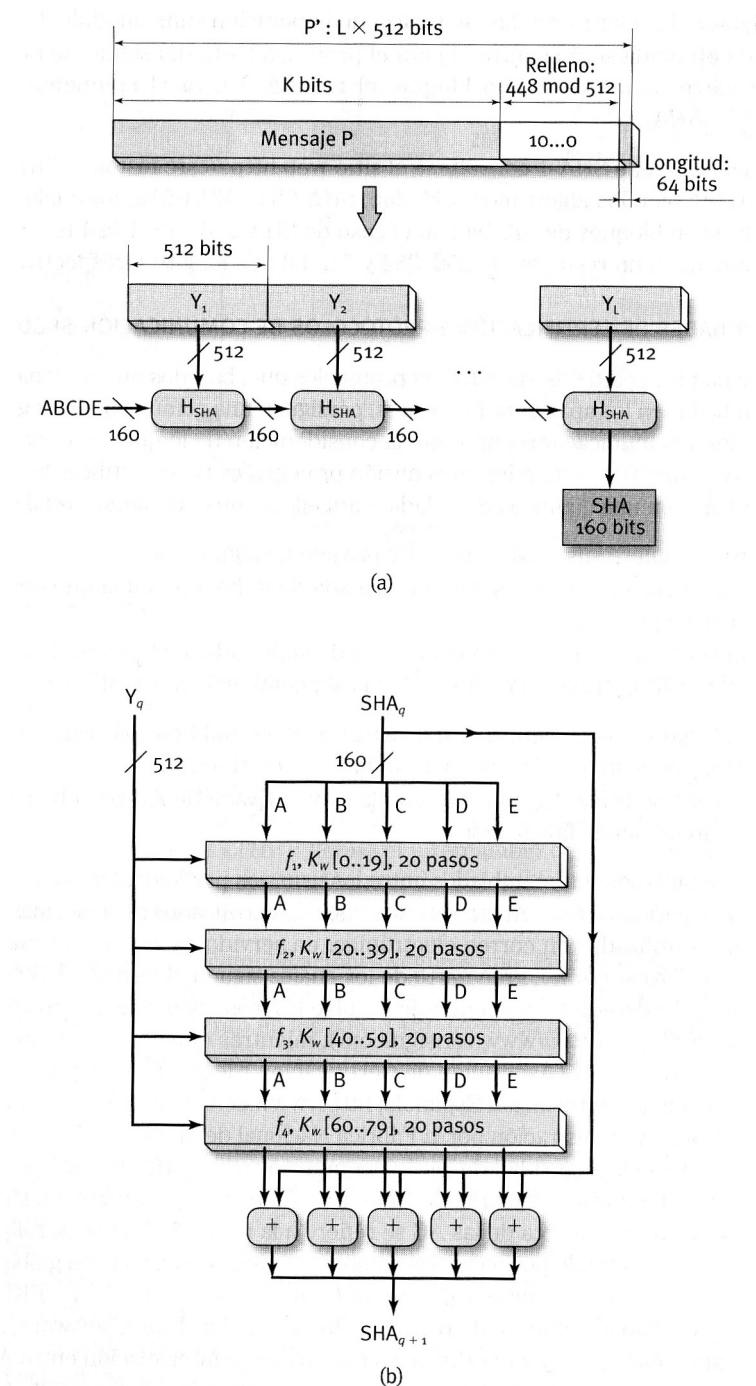


Ilustración 6.1.1: Algoritmo SHA-1

## 6.1 .Subsistema Gestión Web Service y cliente Web Service

---

Para implementar el método de cifrado SHA-1 se hace uso de una clase auxiliar *MiSHA.java*. El método que se utiliza para obtener el resumen SHA-1 es:

```
public static String SHA1(String text) throws  
NoSuchAlgorithmException, UnsupportedEncodingException {  
    MessageDigest md;  
    md = MessageDigest.getInstance("SHA-1");  
    byte[] shalhash = new byte[40];  
    md.update(text.getBytes("iso-8859-1"), 0,  
text.length());  
    shalhash = md.digest();  
    return convertToHex(shalhash);  
}
```

Para el presente proyecto, la firma del mensaje consiste en la obtención del resumen SHA-1 resultante de cifrar la cadena formada por los parámetros que requiera la operación más el usuario y password. Con este método de autentificación el password del usuario nunca viaja a través de la red.

Los parámetros de todas las operaciones requerirán que entre ellos este la firma del mensaje y el usuario que firma el mensaje. Una vez recibidos los parámetros de la operación se comprueba que sean correctos y a continuación se comprueba que el usuario es quien dice ser. La forma de comprobar si el usuario indicado es el que ha firmado el mensaje es:

- Se obtiene el password almacenado en la base de datos para el usuario indicado.
- Se obtiene la firma de los parámetros usando el password del usuario indicado como parámetro por el cliente del servicio.
- Se comprueba que la firma obtenida es idéntica a la firma enviada por el cliente del servicio web.

A modo de ejemplo, veamos cómo se obtiene la firma para la operación *deleteNoticia()*:

```
String cadena = g.getNombre() + g.getContrasenia() +  
id_noticia;  
String f = null;  
try {  
    f = MiSHA.SHA1(cadena);
```

## 6.1 .Subsistema Gestión Web Service y cliente Web Service

```
    } catch (NoSuchAlgorithmException ex) {
        Logger.getLogger(NewsService.class.getName()).log( Level.SEVERE, null, ex);
    } catch (UnsupportedEncodingException ex) {
        Logger.getLogger(NewsService.class.getName()).log( Level.SEVERE, null, ex);
    }
}
```

En este caso los parámetros de la operación son nombre/login del gestor, identificador de la noticia a eliminar y la firma.

El orden en el que se concatenan cada uno de los parámetros usados para obtener el resumen mediante SHA-1 debe ser el indicado en la documentación JAVADOC de cada una de las operaciones disponibles. Si se altera el orden, la firma obtenida en el servicio web no coincidirá con la indicada por el usuario como parámetro y la autenticación será incorrecta.

Otro aspecto importante en la implementación del servicio web es como informar al cliente del servicio de que se ha producido un error. En un servicio web también están presentes las excepciones y fallos. Con el uso de excepciones podremos dar “seguridad” al servicio de usuarios.

Para la gestión de los posibles fallos se crean dos tipos de excepciones: *InvalidParamException* y *InvalidUserException*.

Mediante la excepción *InvalidParamException* se informa al usuario de que ha ocurrido un error con los parámetros que se han indicado. Mediante un código de error y un mensaje que acompañan a la excepción producida el cliente podrá detectar cuál a sido el problema. Los códigos de error posibles son:

- ✗ **201**: indica un error relacionado con las fechas indicadas como parámetros.
- ✗ **202**: indica un error relacionado con la categoría indicada como parámetro.
- ✗ **203**: indica que se ha producido un error con la noticia indicada como parámetro.
- ✗ **204**: indica que se ha producido un error con el dispositivo/os indicado/os como parámetro/os.
- ✗ **205**: indica que se ha producido un error con el usuario indicado como parámetro.

```
public static final int DATEERROR = 201;
public static final int CATEGORYERROR = 202;
public static final int NEWERROR = 203;
```

## 6.1 .Subsistema Gestión Web Service y cliente Web Service

---

```
public static final int DISPERROr = 204;  
public static final int LOGINERROR = 205;
```

Un aspecto a tener en cuenta para la seguridad e integridad del sistema es hacer frente a posibles ataques. Existe la posibilidad de que un intruso capte un paquete de, pongamos un ejemplo, actualizar la información de una noticia. Este intruso podría suplantar al gestor y usar dicho paquete en cualquier momento, deshaciendo posibles cambios posteriores llevados a cabo por el gestor. Para evitar este tipo de ataques se añade a la cadena a firmar por SHA-1 un sello de tiempo. El sello consistirá en fecha y hora exacta en la que se envía el paquete. Incluyendo dicho elemento en la cadena a usar para obtener el resumen SHA-1 se aceptará en el destino dicho paquete durante un periodo de una hora. Transcurrida dicha hora, si el servicio recibe dicho paquete, este será descartado al no coincidir la firma enviada con la obtenida por el servicio web.

Este aspecto se tiene en cuenta en todas las operaciones de actualización y borrado. En el caso de las inserciones se almacena la firma, y se impide la existencia de dos noticias iguales con la misma firma.

### 6.1.2 Cliente Web Service

Explicados los aspectos más destacados de la implementación del servicio web pasamos a detallar los correspondientes al cliente de dicho servicio Web.

Para la implementación del cliente del servicio web se utilizan tecnologías web que dotan de dinamismo: PHP y JavaScript.

El cliente web implementado lleva a cabo un control de sesiones. Dado que todas las operaciones del servicio web requieren indicar como parámetro usuario gestor y firma, sería muy incomodo estar continuamente introduciendo esos datos. Para facilitar el uso de la web, se lleva a cabo un control de sesiones usando la librería PHPLib implementada para PHP. PHPLIB es una biblioteca de funciones que, entre otras, incorpora una capa de abstracción de Bases de Datos, gestión de sesiones, autentificación, etc.

El control de sesiones requiere la existencia de una base de datos. En este caso se alojará en el mismo servidor de base de datos de MySQL usado para la base de datos central del sistema.

El acceso a esta base de datos local para el cliente del servicio web se realiza mediante el uso de PHPLib. Concretamente se utiliza la clase *DB\_Sql*.

La clase *DB\_Sql* implementa la funcionalidad necesaria para acceder a cualquier base de datos que soporte la PHPLIB. ¿Por qué usar esta clase cuando podríamos usar directamente las funciones de acceso que nos proporciona PHP?. El motivo que nos lleva a usar PHPLIB es que con su interfaz de acceso a Bases de Datos, nos podemos olvidar tranquilamente de los

## 6.1 .Subsistema Gestión Web Service y cliente Web Service

---

aspectos particulares del Sistema de Gestión de bases de datos en el cual vayamos a albergar los datos. Con PHPLIB tan sólo tenemos que escribir nuestras consultas en SQL estándar y las funciones de acceso se encargan del resto de tareas, como son establecer la conexión a la base de datos, asegurarse de que la conexión sigue activa antes de lanzar cualquier consulta, recoger los resultados y presentar un interfaz uniforme para acceder a ellos, manejar los errores, etc.

Para comenzar a utilizar esta clase, se implementa una clase propia con los parámetros particulares de acceso a la base de datos, tales como el nombre del host, nombre de la base de datos, usuario, contraseña, etc.

```
//datos de conexión a la BD
class BD_Blufeedme extends DB_Sql {
    var $Host="localhost"; //host donde está ubicado el SGBD
    var $Database="clientblufeedme_bd"; //BD a la que vamos a
acceder
    var $User="cliente"; //usuario con privilegios para esa BD
    var $Password="1234"; //contraseña de acceso
    var $Halt_On_Error="report";
}
```

Esta clase se encuentra dentro del archivo *configuracion.php* para poder ser incluido en cualquier otro archivo de los existentes en el cliente.

Uno de los parámetros que se ha indicado en la definición de la clase para el acceso a la base de datos es *var \$Halt\_On\_Error*. Este parámetro indica si ante un error en el acceso a la base de datos se detendrá la ejecución del programa o no. Los valores que acepta este parámetro son:

- "yes": Indica que ante un error en el acceso a la base de datos, se abortará la ejecución del programa. Normalmente este es el comportamiento que utilizaremos en los programas y si no se especifica nada, también es el comportamiento por defecto.
- "report": El interfaz de acceso a la base de datos informará de los errores, aunque no se parará la ejecución del programa. El código de error y mensaje de error lo podremos encontrar en las variables de instancia Errno y Error, respectivamente. También el comando que haya producido el error devuelve FALSE a la aplicación. Más adelante veremos un ejemplo de uso.
- "no": No se aborta la ejecución del programa ni se muestra mensaje de error, aunque se sigue devolviendo FALSE a la aplicación.

## 6.1 .Subsistema Gestión Web Service y cliente Web Service

---

En este caso se ha indicado la opción *report*.

Una vez creada la clase para la gestión de la conexión con la base de datos se implementa el control de sesiones. Una de las funcionalidades de PHPLib empleadas es la gestión de páginas. Mediante gestión de páginas se incorpora la funcionalidad de control de sesiones y el almacenamiento de variables de usuario.

```
page_open(array("sess" => "Sesion_Blufeedme"));  
...  
page_close();
```

Una vez iniciada la gestión de la página es posible emplear los mecanismos de control de sesiones. Las sesiones permiten usar variables persistentes, es decir, variables que se puedan almacenar y leer entre distintas páginas. Cuando ejecutamos una página PHP que tiene variables, estas "desaparecen" al finalizar la ejecución de la misma. Con las sesiones podemos hacer que las variables que deseemos perduren a través del conjunto de páginas que el usuario cargue. Esto se consigue haciendo uso de una base de datos para almacenar las variables de sesión, concretamente la base de datos mencionada al comienzo del apartado.

Antes de poder hacer uso de las sesiones, se configura PHPLIB para que pueda almacenar en la base de datos la información de la sesión. Para ello PHPLIB hace uso de dos clases especiales, la clase *Session* y la clase *CT\_Sql*.

La clase *CT\_SQL* se usa como "contenedora" de los datos de la sesión. En ella se especifica el método para que PHPLIB pueda almacenar las variables de sesión. Para la solución propuesta usamos una base de datos SQL.

La configuración de esta clase se hace mediante la clase *Blufeedme\_CT\_Sql*, que hereda de *CT\_SQL* de la siguiente manera:

```
class Blufeedme_CT_Sql extends CT_Sql {  
    var $database_class = "BD_Blufeedme";           // nombre de la  
    BD  
    var $database_table = "active_sessions"; // tabla con datos  
    de sesiones  
}
```

Con ésto estamos definiendo que la base de datos *BD\_Blufeedme* va a contener una tabla llamada *active\_sessions* con la información de las sesiones. La clase para el acceso a la base de datos *BD\_Blufeedme* ha sido definida

## 6.1 .Subsistema Gestión Web Service y cliente Web Service

---

previamente.

En la base de datos existe una tabla *active\_sessions* con la siguiente estructura:

Field	Type	Null	Key	Default	Extra
sid	varchar(32)		PRI		
name	varchar(32)		PRI		
val	text	YES		NULL	
changed	varchar(14)		MUL		

La clase *Session* es la que implementa los métodos para grabar y recuperar las variables dentro de un contenedor de datos *Blufeedme\_CT\_Sql*. Para configurarla, creamos y configuramos nuestra propia clase de la siguiente manera:

```
class Sesion_Blufeedme extends Session {  
    var $classname= "Sesion_Blufeedme"; //el propio nombre  
    var $cookiename= ""; //nombre de la "cookie". Por defecto  
    es $classname  
    var $magic = "Patadecabra"; //valor aleatorio para  
    introducir "ruido"  
    var $mode = "cookie"; //modo de propagar las sesiones  
    var $fallback_mode = "get"; //modo auxiliar por si no  
    tenemos "cookies"  
    var $lifetime= 0; //tiempo de vida 0=hasta cerrar el  
    navegador  
    var $that_class = "Blufeedme_CT_Sql"; //clase contenedora  
    var $gc_probability = 5; //probabilidad de borrar sesiones  
    obsoletas  
    var $allowcache = "no"; //¿permitir caché? ("public",  
    "private", ó "no")  
}
```

Los métodos que se utilizan para manipular variables de sesión son los siguientes:

- **register('nombre\_variable')**: Registra (almacena) una variable global en la sesión. La variable puede ser un escalar, un array o un objeto.
- **unregister('nombre\_variable')**: Elimina una variable global de la sesión. La variable no se borra, aunque al finalizar la ejecución de la página no se

## 6.1 .Subsistema Gestión Web Service y cliente Web Service

---

volverá a grabar en la base de datos.

- ***is\_registered('nombre\_variable')***: Devuelve TRUE si la variable está registrada en la sesión. En caso contrario devuelve FALSE.
- ***Delete()***: Borra todos los datos de la sesión de la base de datos.

Como se indicó en el apartado 5.3 , Análisis y Diseño del sistema se emplea una clase para establecer la comunicación con el servicio web, la clase *ClienteNewsService*. Esta clase esta implementada en PHP. La versión 5 empleada soporta WS de forma nativa, permitiendo la conexión con WSDL o sin él. Para la implementación del cliente se emplea el frameworks nativo de PHP5. Dada la importancia de la clase, y el hecho de que pueda ser usada por alguno de los gestores para integrar el servicio web en su sistema, vamos a ver el código completo. El código corresponde al archivo *clienteNewsService.php* contenido en el cdrom adjunto a la documentación.

```
class ClienteNewsService{  
    private $client;  
    private $user;  
    private $password;  
  
    function __construct($user, $password) {  
        $this->user = $user;  
        $this->password = $password;  
        $this->client = new SoapClient(  
            "http://127.0.0.1:8080/ServicioBlufeed  
me/NewsServiceService?wsdl");  
    }  
    public function setClient($client) {  
        $this->client = $client;  
    }  
    public function setUser($user) {  
        $this->user = $user;  
    }  
    public function setPass($password) {  
        $this->password = $password;  
    }  
}
```

## 6.1 Subsistema Gestión Web Service y cliente Web Service

```
public function getClient(){
    return $this->client;
}

public function getUser(){
    return $this->user;
}

public function getPassword(){
    return $this->password;
}

public function addNews($news){
    $str = $this->user . $this->password . $news->autor .
    $news->categoria . $news->titulo . $news->texto . $news-
    >subtitulo;
    $str = $str . $news->fechaPubli . $news-
    >fechaCaducidad;
    $signature = sha1(mb_convert_encoding($str, "ISO-8859-
    1", "UTF-8"));
    $param = array('usuario'=>
        $this->user, 'firma'=>$signature, 'noticia'=>
        $news);

    $result = $this->client->addNoticia($param);
    if($result->return!=NULL)
        return $result->return;
    return -1;
}

public function deleteNews($id_news){
    $str = $this->user . $this->password . $id_news;
    //Añadimos la fecha actual con hora para firmar el mensaje
    $str = $str . $now->format('Y-m-d\TH');
    $signature = sha1(mb_convert_encoding($str, "ISO-8859-
    1", "UTF-8"));
    $param = array('usuario'=>$this->user,
        'firma'=>$signature, 'id_noticia'=>$id_news);
    $result = $this->client->deleteNoticia($param);
```

## 6.1 Subsistema Gestión Web Service y cliente Web Service

---

```
    if($result->return!=NULL)
        return TRUE;
    return FALSE;
}

public function updateNews($id_news,$news) {
    $str = $this->user . $this->password . $id_news .
    $news->autor . $news->categoria . $news->titulo . $news->texto .
    $news->subtitulo;

    try{
        $now = new DateTime("now", new
                            DateTimeZone("Europe/Madrid"));
    }catch(Exception $ex){
        echo $e->getMessage();
        exit(1);
    }
    $str = $str . $news->fechaPublicacion . $news-
>fechaCaducidad;
    $str = $str . $now->format('Y-m-d\TH');

    $signature = sha1(mb_convert_encoding($str, "ISO-8859-
1", "UTF-8"));
    $param = array('usuario'=>$this->user,
                  'firma'=>$signature,
                  'id_noticia'=>$id_news, 'new_noticia'=>$news);
    $result = $this->client->updateNoticia($param);

    if($result->return!=NULL)
        return TRUE;

    return FALSE;
}
public function getNews($category) {
    $str = $this->user . $this->password . $category;
    $signature = sha1(mb_convert_encoding($str, "ISO-8859-
```

## 6.1 .Subsistema Gestión Web Service y cliente Web Service

```
1", "UTF-8"));

$params = array('usuario'=>$this->user,
                'categoria'=>$category,
                'firma'=>$signature);

$result = $this->client->getNoticias($params);

if($result->return != NULL) {
    if(is_array($result->return)) {
        return $result->return;
    }
    else if(is_object($result->return)) {
        return array($result->return);
    }
}
else return;
}

public function getCategory() {
    $str = $this->user . $this->password;
    $signature = sha1(mb_convert_encoding($str, "ISO-8859-1", "UTF-8"));

    $params = array('usuario'=>$this->user, 'firma'=>$signature);

    $result = $this->client->getCategorias($params);

    if($result->return != NULL) {
        if(is_array($result->return)) {
            return $result->return;
        }
        else if(is_object($result->return)) {
            return array($result->return);
        }
    }
}
else return;
}

public function addB($device) {
    $str = $this->user . $this->password . $device->mac .
```

## 6.1 .Subsistema Gestión Web Service y cliente Web Service

---

```
$device ->pin;

        $signature = sha1(mb_convert_encoding($str, "ISO-8859-1", "UTF-8"));

        $params = array('usuario'=>$this->user,
'dispositivo'=>$device, 'firma'=> $signature);

        $result = $this->client->
                addDispositivoMovil($params);

        if($result->return != NULL)
                return $result->return;
        else
                return -1;
}

public function deleteB($mac) {
        $str = $this->user . $this->password . $mac;
//Añadimos la fecha actual con hora para firmar el mensaje
        $str = $str . $now->format('Y-m-d\TH');

        $signature = sha1(mb_convert_encoding($str, "ISO-8859-1", "UTF-8"));

        $params = array('usuario'=>$this->user, 'MAC'=>$mac,
'firma'=> $signature);

        $result = $this->client->
                deleteDispositivoMovil($params);

        if($result->return!=NULL)
                return TRUE;
        return FALSE;
}

public function updateB($mac, $macN, $pinN) {
        $str = $this->user . $this->password . $mac . $macN .
$pinN;

        try{
                $now = new DateTime("now", new
                        DateTimeZone("Europe/Madrid"));
        } catch(Exception $ex) {
```

## 6.1 .Subsistema Gestión Web Service y cliente Web Service

```
        echo $ex->getMessage();
        exit(1);
    }

    $str = $str . $now->format('Y-m-d\TH');

    $signature = sha1(mb_convert_encoding($str, "ISO-8859-1", "UTF-8"));

    $params = array('usuario'=>$this->user, 'MAC'=>$mac, 'MACN'=>$macN, 'pinN'=>$pinN, 'firma'=>$signature);

    $result = $this->client->
                updateDispositivoMovil($params);

    if($result->return!=NULL)
        return TRUE;

    return FALSE;
}

public function associate($mac, $category) {
    $str = $this->user . $this->password . $mac . $category;

    try{
        $now = new DateTime("now", new DateTimeZone("Europe/Madrid"));
    }catch(Exception $ex){
        echo $ex->getMessage();
        exit(1);
    }

    $str = $str . $now->format('Y-m-d\TH');

    $signature = sha1(mb_convert_encoding($str, "ISO-8859-1", "UTF-8"));

    $params = array('usuario'=>$this->user, 'MAC'=>$mac, 'categoria'=> $category, 'firma'=>$signature);

    $result = $this->client->asociar($params);

    if($result->return!=NULL)
        return TRUE;
```

## 6.1 Subsistema Gestión Web Service y cliente Web Service

---

```
        return FALSE;
    }

    public function disassociate($mac, $category) {
        $str = $this->user . $this->password . $mac .
$category;
        try{
            $now = new DateTime("now", new
                DateTimeZone("Europe/Madrid"));
        }catch(Exception $ex){
            echo $ex->getMessage();
            exit(1);
        }
        $str = $str . $now->format('Y-m-d\TH');
        $signature = sha1(mb_convert_encoding($str, "ISO-8859-
1", "UTF-8"));
        $params = array('usuario'=>$this->user, 'MAC'=>$mac,
'Categoria'=> $category, 'firma'=>$signature);
        $result = $this->client->desasociar($params);
        if($result->return!=NULL)
            return TRUE;
        return FALSE;
    }

    public function getDispositivos(){
        $str = $this->user . $this->password;
        $signature = sha1(mb_convert_encoding($str, "ISO-8859-
1", "UTF-8"));
        $params = array('usuario'=>$this->user,
                       'firma'=>$signature);
        $result = $this->client->getDispositivos($params);
        if($result->return != NULL){
            if(is_array($result->return)){
                return $result->return;
            }
        }
    }
}
```

## 6.1 .Subsistema Gestión Web Service y cliente Web Service

```
        else if(is_object($result->return)) {
            return array($result->return);
        }
    }
    else return;
}

public function getMyDispositivos(){
    $str = $this->user . $this->password;
    $signature = sha1(mb_convert_encoding($str, "ISO-8859-1", "UTF-8"));
    $params = array('usuario'=>$this->user,
                    'firma'=>$signature);
    $result = $this->client->getMyDispositivos($params);
    if($result->return != NULL){
        if(is_array($result->return)){
            return $result->return;
        }
        else if(is_object($result->return)){
            return array($result->return);
        }
    }
    else return;
}

public function getDispositivosWhereCategoria($categoria){
    $str = $this->user . $this->password . $categoria
    $signature = sha1(mb_convert_encoding($str, "ISO-8859-1", "UTF-8"));
    $params = array('usuario'=>$this->user,
                    'categoria'=>$categoria, 'firma'=>$signature);
    $result = $this->client->
                getDispositivosWhereCategoria($params);
    if($result->return != NULL){
        if(is_array($result->return)) {
```

## 6.1 .Subsistema Gestión Web Service y cliente Web Service

---

```
        return $result->return;
    }
    else if(is_object($result->return)) {
        return array($result->return);
    }
    else return;
}
}
```

Para implementar el cliente se hace uso del frameworks de PHP SOAP. Como se puede observar el cliente básicamente realiza llamadas a las operaciones del servicio web. Adapta los parámetros indicados por el usuario a las especificaciones del servicio web y convierte los resultados devueltos por las operaciones del servicio a un formato más cómodo para su utilización en la capa de interfaz.

En lo que respecta a la interfaz se ha empleado JavaScript, concretamente la librería JQuery. Se emplean técnicas de actualización de contenido como AJAX para solamente cargar partes concretas de la web sin necesidad de recargar la página completa, y diferentes funciones disponibles en la librería para dotar de efectos visuales la plataforma web.

Los datos introducidos por el usuario en cada uno de los formularios se validan en ambos extremos, es decir, en el lado del cliente mediante JavaScript y en el lado del servidor usando PHP. Se emplea el método POST para recibir los datos de los distintos formularios. Cuando se usa el método POST, los datos del formulario se reciben a través de la entrada estándar, es decir, el programa que lo recibe lee la cadena de parámetros como si se hubieran escrito desde el teclado, sin usar la variable QUERY\_STRING.

Esto tiene como ventajas e inconvenientes:

- Hay que pasar los datos forzosamente a través de un formulario.
- No existe límite en cuanto a la cantidad y tamaño de los datos que se transfieren.
- Es más seguro, puesto que ni se lee lo que se ha escrito en la barra del navegador, aunque también se pueden realizar llamadas directas al programa con parámetros arbitrarios.

Toda la plataforma web cliente del servicio web ha sido validada desde el validador disponible en **W3C** (CSS v3 y XHTML 1.0).

## 6.2 Subsistemas de Publicación de noticias

En este apartado veremos los detalles de implementación de cada uno de los sistemas de publicación de noticias disponibles en el sistema: publicación mediante web mostrada en las pantallas colocadas por el edificio y publicación mediante el envío de noticias por Bluetooth a los usuarios.

### 6.2.1 Subsistema Web pantallas

El subsistema para publicación de noticias a través de las pantallas está implementado a través de una web. Dicha web se implementa usando de nuevo lenguajes que aportan dinamismo, PHP y JavaScript.

La web consiste en un acordeón de noticias, combinado con las características de un carrousel, implementado usando la librería JQuery. Para los efectos acordeón a la hora de mostrar y desplegar el contenido de la noticia se emplea *UI/API/1.7.2/Accordion*.

```
$( "#accordion" ).accordion({  
    header: "h3",  
    collapsible: true,  
    active: indice,  
    autoHeight: false,  
    clearStyle: true});  
...
```

- *header*: selector para los elementos cabecera.
- *Collapsible*: indica si todas las secciones se pueden cerrar a la vez.
- *Active*: indica el elemento activo.
- *AutoHeight*: si se activa, la parte más alta de contenido se utiliza como referencia de altura para el resto de las partes.
- *ClearStyle*: Si se activa, borra la altura y el desbordamiento de los estilos después de acabar las animaciones. Esto permite trabajar con contenido dinámico.

De nuevo el subsistema accede a la base de datos central disponible en el sistema. Para acceder se utiliza la librería PHPLib disponible en PHP. Concretamente se utiliza la clase *DB\_Sql*.

La clase *DB\_Sql* implementa la funcionalidad necesaria para acceder a cualquier base de datos que soporte la PHPLIB. El motivo que nos lleva a usar

## 6.2 .Subsistemas de Publicación de noticias

---

PHPLIB es que con su interfaz de acceso a Bases de Datos, nos podemos olvidar tranquilamente de los aspectos particulares del Sistema de Gestión de bases de datos en el cual vayamos a albergar los datos. Con PHPLIB tan sólo tenemos que escribir nuestras consultas en SQL estándar y las funciones de acceso se encargan del resto de tareas, como son establecer la conexión a la base de datos, asegurarse de que la conexión sigue activa antes de lanzar cualquier consulta, recoger los resultados y presentar un interfaz uniforme para acceder a ellos, manejar los errores, etc.

Para comenzar a utilizar esta clase, se implementa una clase propia con los parámetros particulares de acceso a la base de datos, tales como el nombre del host, nombre de la base de datos, usuario, contraseña, etc.

```
class BD_Blufeedme extends DB_Sql {  
    var $Host="localhost"; // host donde está ubicado el SGBD  
    var $Database="blufeedme_db"; //BD a la que vamos a acceder  
    var $User = "root_blufeedme"; //usuario con privilegios para  
esa BD  
    var $Password = "1234"; // contraseña de acceso  
    var $Halt_On_Error = "yes"; // ¿abortar si hay algún error?  
    ...  
}
```

Esta clase se encuentra dentro del archivo *configuracion.php* para poder ser incluido en cualquier otro archivo de los existentes en el cliente.

Uno de los parámetros que se ha indicado en la definición de la clase para el acceso a la base de datos es *var \$Halt\_On\_Error*. Este parámetro indica si ante un error en el acceso a la base de datos se detendrá la ejecución del programa o no. Los valores que acepta este parámetro son:

- *"yes"*: Indica que ante un error en el acceso a la base de datos, se abortará la ejecución del programa. Normalmente este es el comportamiento que utilizaremos en los programas y si no se especifica nada, también es el comportamiento por defecto.
- *"report"*: El interfaz de acceso a la base de datos informará de los errores, aunque no se parará la ejecución del programa. El código de error y mensaje de error lo podremos encontrar en las variables de instancia Errno y Error, respectivamente. También el comando que haya producido el error devuelve FALSE a la aplicación. Más adelante veremos un ejemplo de uso.
- *"no"*: No se aborta la ejecución del programa ni se muestra mensaje de

## 6.2 .Subsistemas de Publicación de noticias

---

error, aunque se sigue devolviendo FALSE a la aplicación.

En este caso se ha indicado la opción *report*.

A través de la clase *BD\_Blufeedme* el gestor de la capa de aplicación accede a los datos existentes en la base de datos central. Una vez obtenidos los datos el gestor formatea los datos obtenidos a XML para que la capa de interfaz pueda acceder a la información de forma sencilla. Con esta estrategia, en un futuro, si se desea, la interfaz podría leer noticias de archivos con formato XML.

Cada cierto tiempo el carrusel obtiene las noticias existentes en la base de datos en formato XML. Una vez obtenidas comprueba si se ha producido alguna modificación en las noticias existentes. En caso afirmativo, se actualiza el contenido de la página web mediante AJAX.

```
setInterval('updateXML()',velocidad_updateXML);
```

Siendo el código de la función *updateXML*:

```
function updateXML() {
    $.ajax({
        url: "generaXML.php",
        success: function(xml) {
            if(xml != xml_actual){
                ObtenerItemsHTML(xml);
                xml_actual = xml;
            }
        }
    });
}
```

Los principales parámetros para proporcionar los efectos de un carrusel al acordeón son:

```
//Creación del carousel
MiCarousel = $("#mycarousel").msCarousel({
    boxClass:'div.box',
    vertical:true,
    width:ancho_carousel,
    height:alto_carousel,
```

## 6.2 .Subsistemas de Publicación de noticias

---

```
loop:true,  
autoSlide:false,  
scrollSpeed:velocidad_scroll,  
showMessage:true,  
messageClass:'.message',  
messageOpacity:0.4  
}).data("msCarousel");
```

Con el parámetro *scrollSpeed* controlamos la velocidad de la animación.

Para mostrar las noticias de forma correcta se comprueban las dimensiones de la pantalla y la dimensión del conjunto de noticias que conforman el carrusel. El carrusel puede ser controlado por el usuario, desplegando, seleccionando noticias mediante ratón, etc. pero sin la intervención del usuario este ejecuta una animación continuada.

La animación del carrusel, que añade un efecto acordeón, tiene en cuenta dimensiones tanto de pantalla como de noticias y en función de ellas seguirá una serie de pasos u otros. Para obtener el tamaño total de las noticias se suma el tamaño de las divisiones ocupadas por la información básica para todas las noticias, más el cuerpo de la noticia de mayor extensión.

```
...  
$cuerpos_noticias = $("#mycarousel div.box p.parrafo");  
$cuerpos_noticias.each( function(index) {  
    //Tomamos la altura con paddings y margins incluidos (true)  
    aux = $(this).outerHeight(true);  
    if(aux > alt_max) alt_max = aux;  
});  
    //Ahora realizamos el calculo añadiendo el tamaño de todas  
    las cabeceras  
alt_max = alt_max + ($cuerpos_noticias.prev().outerHeight() *  
n_noticias);  
...
```

Un primer caso es que el conjunto de noticias que se tienen que mostrar en el carrusel, tienen un tamaño menor al de la pantalla por lo que se está mostrando. En este, dado que todas entran en la pantalla, van desplegando su contenido de forma continuada. Una vez desplegadas todas se vuelve a

## 6.2 .Subsistemas de Publicación de noticias

---

comenzar desde la primera noticia (noticia situada en la parte superior de la pantalla).

```
...
if(tam_menor_pantalla) {
    //Abrimos la siguiente (la anterior se cierra sola)
    indice = (indice+1) % n_noticias;
    $("#accordion").accordion("activate", indice);
}
...
```

En el caso de que las noticias no puedan visualizar la información principal de forma simultánea se añade un nuevo efecto a la animación. Las noticias se van desplegando y desplazando hacia arriba en el carrusel. La noticia situada en la parte superior es la noticia que se despliega. Una vez contraída las noticias suben una posición. Cuando se han mostrado todas las noticias se comienza de nuevo con la noticia que inicialmente se encontraba en la parte superior.

```
//Recogemos la noticia actual
$("#accordion").accordion("activate", false);
//Avanzamos el carousel pero esperando el tiempo necesario para
que se cierre la noticia
setTimeout('MiCarousel.next()', 1000);
//Esperamos a que se haya avanzado por completo una posición y
desplegamos la noticia actual
setTimeout('$("#accordion").accordion("activate",
MiCarousel.getCurrentID());', 1000+velocidad_scroll*1.1);
```

En lo que respecta a la posibilidad de que el usuario intervenga de forma independiente a la animación para visualizar el contenido se añaden:

- Modificación de la velocidad de animación mediante la rueda de scroll.
- Detección de la animación cuando se coloca el puntero sobre algún elemento del carrusel.
- Reactivación de la animación cuando se sale del área del elemento sobre el que estaba situado el puntero.

## 6.2 .Subsistemas de Publicación de noticias

---

```
//Añadimos el efecto si se pulsa la rueda de scroll
$('#mycarousel').bind('mousewheel', function(event, delta) {
    var dir = delta > 0 ? 'Up' : 'Down', vel = Math.abs(delta);
    //Indicamos una mayor velocidad de movimiento
    MiCarousel.setVelocidad(velocidad_scroll*0.15);
    (dir == 'Up') ? MiCarousel.previous() : MiCarousel.next();
    //Reestablecemos la velocidad original
    MiCarousel.setVelocidad(velocidad_scroll);
});

//Añadimos el control del evento mouseover (deteniendo la
animación)
$("#mycarousel").bind("mouseover", function() {
    detener_mouseover = true; });

//Añadimos el control del evento mouseout (reanudamos la
animación)
$("#mycarousel").bind("mouseout", function() {
    detener_mouseover = false; });
```

Toda la plataforma web cliente del servicio web ha sido validada desde el validador disponible en **W3C** (CSS v3 y XHTML 1.0 strict).

### 6.2.2 Subsistema Bluetooth

El subsistema Bluetooth está implementado mediante el lenguaje Java. Como se explicó en el apartado de Diseño: Arquitectura correspondiente a este subsistema, todo el peso de la aplicación recae sobre dos clases: DispositivoBT y EnviarAdispositivo. A continuación explicaremos los detalles más significativos de la implementación de ambas clases.

#### 6.2.2.1 Clase DispositivoBT

En primer lugar explicaremos las particularidades de la clase DispositivoBT implementada en el fichero DispositivoBT.java. Esta clase representa al dispositivo que actuará como servidor Bluetooth y que realizará el envío de ficheros de noticias al resto de dispositivos móviles con los que se conecte. En la clase se hace uso de la librería Bluecove que se puede ver con más detalle en el Anexo V: Bluetooth:JSR 82. Esta librería nos proporciona una serie de interfaces y clases que nos permiten gestionar gran variedad de pilas Bluetooth.

## 6.2 .Subsistemas de Publicación de noticias

---

La clase DispositivoBT hereda de la clase `java.lang.Thread` por lo que nuestra clase será una hebra con todas sus características. Como todas las hebras, tiene un método `run()` que es ejecutado cuando se crea una hebra y se llama al método `start()` de la hebra correspondiente. También implementa la interfaz `javax.bluetooth.DiscoveryListener` perteneciente al estándar JSR-82 y que está implementada en Bluecove, esta interfaz es la que nos proporciona una serie de métodos, que tendremos que implementar, para comunicarse con dicha librería.

Los métodos implementados por parte de la interfaz son:

- `void deviceDiscovered(RemoteDevice btDevice, DeviceClass cod)`

Este método será llamado por la propia librería Bluecove cuando un nuevo dispositivo Bluetooth es descubierto en el área de cobertura de la antena durante un escaneo de dispositivos (Inquiry). Los datos del dispositivo descubierto nos serán transferidos a través de las variables del método.

- `void servicesDiscovered(int transID, ServiceRecord[] servRecord)`

Este método será llamado por la propia interfaz Bluecove cuando un nuevo servicio perteneciente a un dispositivo es descubierto por la antena durante un escaneo de servicios. Los datos del servicio nos serán transferidos a través de las variables del método.

- `void serviceSearchCompleted(int transID, int respCode)`

Este método será llamado por la propia interfaz Bluecove cuando la búsqueda de servicios haya finalizado. El motivo por el cuál a finalizado la búsqueda de servicios se obtiene por medio de la variable `respCode` que puede tener los siguientes valores:

- `DiscoveryListener.SERVICE_SEARCH_COMPLETED`  
Búsqueda de servicios completada con éxito.
- `DiscoveryListener.SERVICE_SEARCH_TERMINATED`  
Búsqueda de servicios terminada por la aplicación y no ha sido completada.
- `DiscoveryListener.SERVICE_SEARCH_DEVICE_NOT_REACHABLE`  
Dispositivo no alcanzable para el servicio indicado.
- `DiscoveryListener.SERVICE_SEARCH_NO_RECORDS`  
No se encontraron registros del servicio indicado.
- `DiscoveryListener.SERVICE_SEARCH_ERROR`  
Error en la búsqueda de servicios.

- `void inquiryCompleted(int discType)`

## 6.2 .Subsistemas de Publicación de noticias

---

Este método será llamado por la propia interfaz Bluecove cuando la búsqueda de dispositivos haya finalizado. El motivo por el cuál a finalizado la búsqueda de dispositivos se obtiene por medio de la variable discType que puede tener los siguientes valores:

- DiscoveryListener.INQUIRY\_COMPLETED  
Búsqueda de dispositivos completada con éxito.
- DiscoveryListener.INQUIRY\_TERMINATED  
Búsqueda de dispositivos terminada por la aplicación y no ha sido completada.
- DiscoveryListener.INQUIRY\_ERROR  
Error en la búsqueda de dispositivos.

Además hay otras tres funciones que no pertenecen a la interfaz sino al estándar JSR-82, son utilizadas en la clase DispositivoBT para enviar peticiones a la librería Bluecove que contiene la implementación de dicho estándar:

- *buscador = LocalDevice.getLocalDevice().getDiscoveryAgent()*

La función LocalDevice.getLocalDevice() nos devuelve una instancia del dispositivo local con el que se ha arrancado la librería Bluecove. Con dicho dispositivo obtenemos un buscador o DiscoveryAgent que nos permitirá realizar las búsquedas de servicios y dispositivos (inquiry).

- *buscador.searchServices(int[] attrSet, UUID[] uuidSet, RemoteDevice btDev, DiscoveryListener discListener)*

A través del buscador obtenido del dispositivo local, se ejecuta la orden de que comience una búsqueda de servicios para el dispositivo btDev en una nueva hebra. Todas las respuestas serán enviadas al discListener a sus correspondientes funciones que han sido explicadas más arriba y que realizarán las operaciones que se hayan establecido.

El UUID es el código del servicio que será buscado en el dispositivo, en nuestro caso el código para OBEX\_PUSH es 0x1105 (otros serían RFCOMM=0x0003, L2CAP=0x0100, etc).

El attSet indica los atributos que serán obtenidos del servicio, en este caso únicamente nos interesa la URL pero también se puede obtener el nombre, el id etc.

**Nota:** En la aplicación únicamente se buscan los servicios en los dispositivos no registrados previamente en la base de datos o en aquellos dispositivos de los que no dispongamos la URL de servicio, de este modo ahorraremos tiempo y recursos del dispositivo Bluetooth.

## 6.2 .Subsistemas de Publicación de noticias

---

- *buscador.startInquiry(int accessCode, DiscoveryListener discListener)*

A través del buscador obtenido del dispositivo local, se ejecuta la orden de que comience una búsqueda de dispositivos que estén al alcance a través de una nueva hebra. Todas las respuestas serán enviadas al discListener a sus correspondientes funciones que han sido explicadas más arriba y que realizarán las operaciones que se hayan establecido.

El accessCode es el tipo de búsqueda que se realizará, hay cuatro tipos de búsqueda diferentes:

- *DiscoveryAgent.CACHED*: obtienen los dispositivos que han sido encontrados en la búsqueda anterior.
- *DiscoveryAgent.GIAC*: obtiene todos los dispositivos físicos que estén al alcance en el momento de la búsqueda y que estén en modo visible.
- *DiscoveryAgent.LIAC*: obtiene los dispositivos físicos de forma limitada, que estén al alcance en el momento de la búsqueda y que estén en modo visible.
- *DiscoveryAgent.NOT\_DISCOVERABLE*: obtiene los dispositivos que estén en modo invisible o no detectable.

En este caso hemos utilizado GIAC como modo de búsqueda porque no queremos detectar los dispositivos que estén en modo invisible. De esta forma la aplicación no será intrusiva para los usuarios y podrán elegir cuándo recibir las noticias.

### 6.2.2.2 Clase EnviarAdispositivo

A continuación pasamos a explicar la clase EnviarAdispositivo que se encarga del envío de ficheros con noticias a los distintos dispositivos móviles detectados previamente. En esta clase se establece una conexión temporal entre el dispositivo móvil detectado y el Hot-Spot, de tal forma que se podrán enviar datos a través del protocolo OBEX PUSH. Los datos se enviarán mediante un fichero con formato xhtml que será compuesto en tiempo de ejecución por la propia clase, realizando peticiones a la capa de servicios (base de datos).

Para realizar la conexión con el dispositivo móvil, es necesario previamente realizar un Pairing entre el dispositivo móvil y el Hot-Spot. Para ello se utilizan las siguientes dos funciones implementadas en Bluecove:

- *RemoteDeviceHelper.implIsAuthenticated(RemoteDevice rmtdev)*

Esta función nos devuelve un booleano indicando si existe un pairing entre el dispositivo local y el dispositivo remoto rmtdev.

- *RemoteDeviceHelper.authenticate(tRemoteDevice rmtdev, String pin)*

Esta función nos permite establecer un pairing entre el dispositivo local Hot-Spot y el dispositivo móvil remoto rmtdev. Para establecer el pairing entre ambos dispositivos, se indica el PIN que deberá introducir

## 6.2 .Subsistemas de Publicación de noticias

---

obligatoriamente el usuario poseedor del dispositivo móvil y que corresponderá con el PIN almacenado en la base de datos para dicho dispositivo.

Una vez establecido el pairing entre ambos dispositivos se abre una conexión mediante la función javax.microedition.io.Connector.open(String URL, int mode, boolean timeouts). Mediante la variable mode se indica si se realizará una conexión de únicamente lectura o la que hemos utilizado nosotros, lectura/escritura (Connector.READ\_WRITE). La URL será la correspondiente al servicio OBEX PUSH para el dispositivo móvil con el que se quiere conectar y tendrá el siguiente formato:

***btgoep://B8F9346A27D3:12;authenticate=false;encrypt=false;master=false***

La función open nos devuelve una sesión o (ClientSession) que será el objeto que nos permita crear una conexión entre dos dispositivos. En primer lugar se crea la cabecera de conexión mediante la función clientSession.createHeaderSet(), dicha cabecera será enviada en el momento de conexión, realizado por la función clientSession.connect(cabeceraOperacion). La función connect nos devuelve una respuesta que deberá coincidir con la siguiente constante ResponseCodes.OBEX\_HTTP\_OK para comprobar que la conexión se realizó con éxito.

Si la conexión tuvo éxito se crea el fichero xhtml que será enviado al dispositivo. Para ello se realizan una serie de consultas a la base de datos y se forma un documento xhtml tomando como plantilla el fichero indicado al inicio de la aplicación bluetooth (ver Ilustración 5.3.42: Pantalla ficheros de propiedades bluetooth). Dicho fichero xhtml cumple el estándar XHTML 1.0 Strict junto con CSS3, de esta forma su visualización será compatible con la mayoría de dispositivos actuales y futuros. Además se ha realizado un diseño proporcional, tanto en dimensiones como en tamaños de tipo de letra, para que pueda ser visualizado sin problemas independientemente de la resolución del dispositivo receptor.

**Nota:** En la clase EnviarAdispositivo se encuentran definidas dos constantes que regulan el máximo número de noticias a enviar en un mismo fichero. Estas constantes son MAX\_NOTICIAS=5 y MAX\_TAM\_NOTICIAS=10000, que indican el número máximo de noticias en un mismo fichero y la suma máxima de caracteres obtenida de los cuerpos de las noticias, respectivamente. De este modo se evita enviar ficheros con un gran tamaño que provocarían un gasto importante en los recursos, ya que ocuparía durante un tiempo excesivo un slot de emisión del Hot-Spot y también evitamos que el usuario tenga en un mismo fichero una cantidad muy alta de noticias.

Para el envío del fichero de datos, una vez establecida la conexión, se utiliza la función `clientSession.put(HeaderSet)` que establece la cabecera de envío de datos que será utilizada por medio del objeto devuelto `putOperation`. Con este objeto `putOperation` se obtiene un stream de datos, a través de la función `putOperation.openOutputStream()`, que será el objeto donde se escriben directamente todos los datos que se deseen enviar al dispositivo, utilizando funciones `write` del propio stream.

**Nota:** el nombre del fichero recibido en el dispositivo móvil viene dado por:

***UGR\_ + última fecha de publicación entre todas las noticias entregadas+.html***

Siendo el formato de la fecha: `dd_MM_yy.HH_mm`

De este modo el usuario podrá visualizar todos los ficheros recibidos ordenados cronológicamente, nada más que ordenándolos por el nombre de fichero.

Una vez finalizado el envío se obtiene la respuesta de la operación `put`, ésta deberá tener el valor `ResponseCodes.OBEX_HTTP_OK` para que la transferencia se haya realizado con éxito. Si la transferencia se realiza con éxito anotamos en la base de datos de la aplicación que las noticias que estaban contenidas en el fichero enviado han sido transmitidas correctamente al dispositivo en cuestión, de esta forma no se repite el envío de una misma noticia a un dispositivo.

**Nota:** Todas las operaciones de búsqueda de dispositivos (Inquiry), servicios, establecimiento de pairing, envío de ficheros, espera de la confirmación de envío etc. poseen un timeout asociado para evitar una espera indefinida en caso de error. Dichos timeouts son indicados mediante el fichero de propiedades de la pila bluetooth, seleccionado al inicio de la aplicación (ver Ilustración 5.3.42: Pantalla ficheros de propiedades bluetooth). Algunos ejemplos de timeouts incluidos en dicho fichero son:

`PROPERTY_OBEX_TIMEOUT:180000`  
`PROPERTY_INQUIRY_DURATION_DEFAULT:20`  
`PROPERTY_CONNECT_UNREACHABLE_RETRY:3`  
`PROPERTY_CONNECT_TIMEOUT:180000`

### 6.2.2.3 Concurrencia

La aplicación bluetooth de blufeedme posee la cualidad de realizar distintas operaciones simultáneamente, con el objetivo de aumentar la eficiencia y así aprovechar al máximo las características físicas que nos proporciona el dispositivo Bluetooth (Hot-Spot) servidor.

## 6.2 .Subsistemas de Publicación de noticias

---

En primer lugar explicaremos las distintas hebras que son creadas durante la ejecución del programa, así como su ciclo de vida y las dependencias entre ellas. Para ello comenzaremos nombrando las hebras más significativas que son creadas en la aplicación:

- **Escaneo\_bluetooth:** esta hebra es la principal de toda la lógica de la aplicación, pertenece a la clase DispositivoBT y se llama Escaneo\_bluetooth por el nombre que se le establece a la hebra en su creación mediante el método `setName()`, heredado de la clase Thread. Esta hebra se ejecuta desde que comienza el programa hasta que termina y únicamente existe una instancia de ella.
- **Envio\_MAC:** esta hebra pertenece a la clase EnviarAdispositivo y se encarga de enviar el fichero de noticias al dispositivo que se indique. Su nombre cambia dependiendo de la MAC del dispositivo al que se realice el envío. Se pueden crear varias hebras, hasta un máximo que se corresponde con el límite de conexiones simultáneas que soporta el dispositivo físico Bluetooth (Hot-Spot). Este valor máximo se obtiene mediante la función `LocalDevice.getProperty("bluetooth.connected.devices.max")`. Su ejecución es incompatible con la hebra DeviceInquiry y SearchService, por tanto, se debe esperar a que finalicen para su comenzar entonces su ejecución.
- **DeviceInquiryThread-N:** esta hebra pertenece a la librería Bluecove y es creada cuando comienza el escaneo de dispositivos mediante el método `discoveryAgent.startInquiry()`, explicado en el apartado anterior. El nombre de la hebra viene dado por el número de búsquedas anteriores que se han realizado, por ejemplo, la primera búsqueda se llama DeviceInquiryThread-0, la siguiente DeviceInquiryThread-1 y así sucesivamente. Esta hebra es incompatible con la búsqueda de servicios y con la hebra de envío de ficheros, ya que el dispositivo no permite realizar envíos ni búsquedas de servicios paralelamente entre sí mientras se descubren dispositivos.
- **SearchServicesThread-N:** esta hebra pertenece a la librería Bluecove y es creada cuando comienza un escaneo de servicios para un dispositivo mediante el método `discoveryAgent.searchServices()`, explicado en el apartado anterior. Al igual que DeviceInquiryThread, el número final del nombre viene dado por un identificador que se va aumentando gradualmente con el tiempo. Esta hebra es incompatible con la búsqueda de dispositivos y con la hebra de envío de ficheros, ya que el dispositivo no permite realizar envíos ni búsquedas de dispositivos paralelamente entre sí mientras se descubren los servicios de un dispositivo. Sin embargo, se pueden crear varias instancias de esta hebra, tantas como permita el dispositivo físico, para buscar paralelamente los servicios en distintos dispositivos al mismo tiempo. Para obtener dicho valor máximo se utiliza la función `LocalDevice.getProperty("bluetooth.sd.trans.max")`.

## 6.2 Subsistemas de Publicación de noticias

- **ConnectThread-n:** esta hebra es creada por javax.microedition.io.Connector cuando se realiza la conexión con un dispositivo móvil mediante la función `Connector.open()`. Se realiza una conexión por cada hebra de Envio\_MAC, por lo que se crearán simultáneamente tantas hebras ConnectThreadCon como envíos de ficheros a dispositivos.

A continuación se muestran unos diagramas que muestran el orden y concurrencia de las distintas hebras:

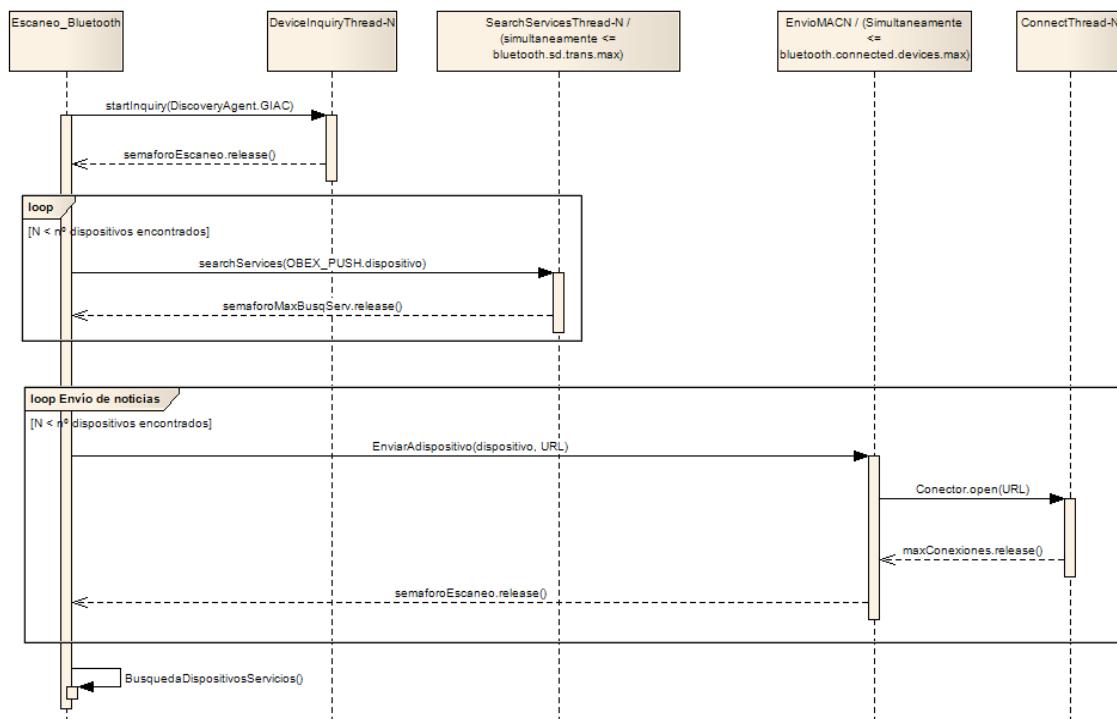


Ilustración 6.2.1: Diagrama secuencia (hebras)

## 6.2 .Subsistemas de Publicación de noticias

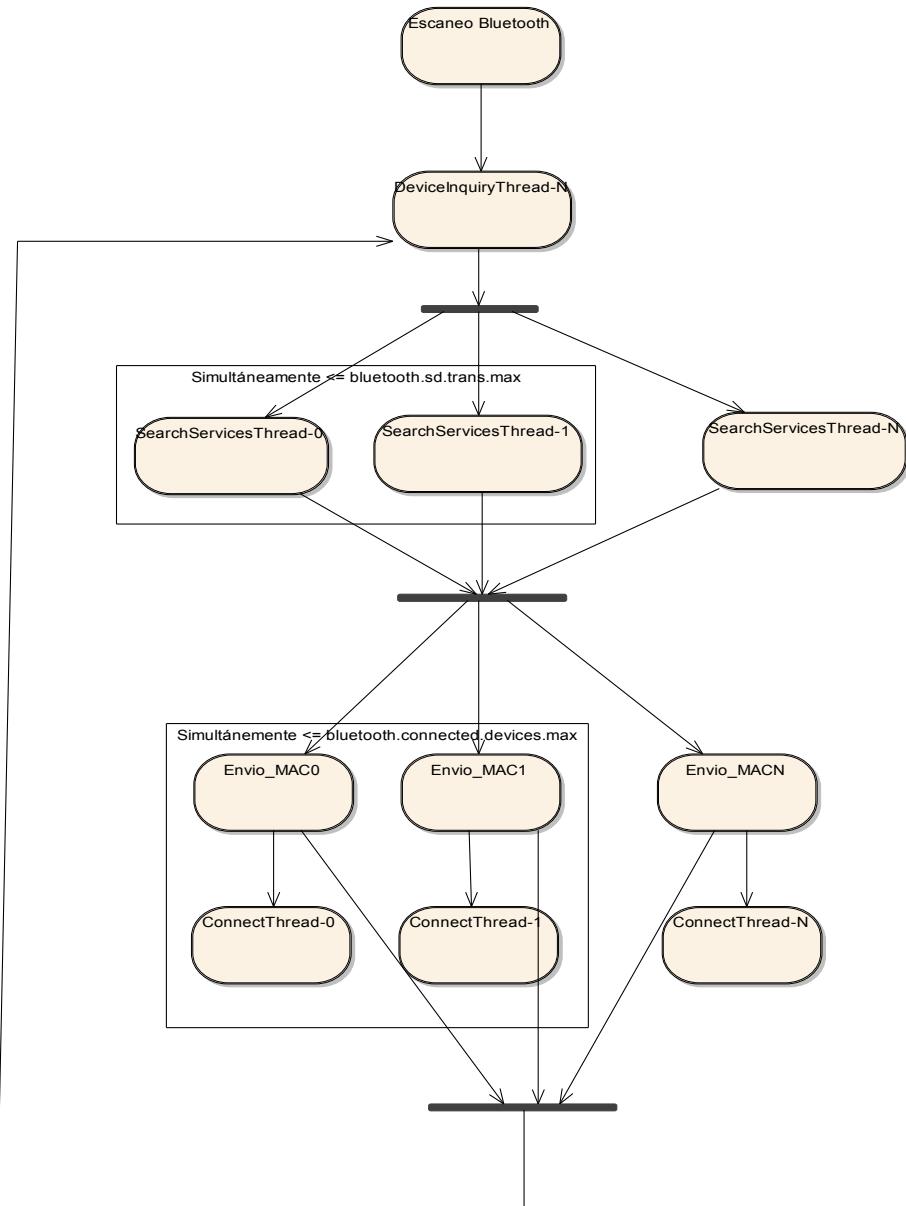


Ilustración 6.2.2: Diagrama dependencias de ejecución

Para la sincronización entre las distintas hebras, se ha utilizado una serie de variables de tipo `java.util.concurrent.Semaphore` que representan a los semáforos utilizados en el control de hebras. Dichas variables nos permiten bloquear la ejecución de determinadas hebras, quedando éstas a la espera de la finalización del resto de hebras de las que depende. También nos permite controlar el número máximo de hebras a lanzar simultáneamente, para que en ningún momento haya en ejecución un número mayor de hebras al establecido. Esto se consigue bloqueando la hebra padre, que se encarga de crear las

## 6.2 .Subsistemas de Publicación de noticias

---

distintas hebras hijas, cuándo se ha alcanzado el límite de hebras que se le permite crear, pasando a ejecutarse de nuevo cuando al menos una de las hebras hijas finaliza (continuar su ejecución o lanzar más hebras del mismo tipo).

Los semáforos utilizados en el programa son:

- **semaforoMaxBusqServ:** Controla el número máximo de hebras de Búsqueda de servicios que pueden ser ejecutadas paralelamente entre sí. Toma como valor, la propiedad del dispositivo *bluetooth.sd.trans.max*.
- **SemaforoMaxConexiones:** Controla el número máximo de hebras de envío que pueden ser ejecutadas paralelamente entre sí. Toma como valor, la propiedad del dispositivo *bluetooth.connected.devices.max*.
- **SemaforoEscaneo:** Controla que no se ejecute más de una hebra de Escaneo de dispositivos simultáneamente, bloqueando la ejecución del programa mientras que la hebra de Inquiry está escaneando los dispositivos. Toma como valor 1 para permitir la ejecución de una única hebra de escaneo de dispositivos.

Para controlar el acceso a variables compartidas entre las distintas hebras se han utilizado diferentes tipos de variables para tal efecto. Las variables que son accedidas simultáneamente por varias hebras y que por tanto es necesario controlar el acceso son:

- **mapDispositivosURL:** esta variable es fundamental para el correcto funcionamiento de la aplicación ya que es la encargada de almacenar todos los dispositivos que se detectan en tiempo real como las URLs de los servicios descubiertos. Se trata de una tabla hash de parejas clave-valor, donde la clave es el propio dispositivo y el valor es la URL de conexión para el servicio OBEX. Para garantizar la exclusión mutua entre hebras se ha utilizado un *java.util.concurrent.ConcurrentHashMap* que nos garantiza el orden de escritura y lectura de las hebras sobre dicha tabla hash.
- **dispositivosRestantesBusq:** esta variable es un simple contador que nos permite conocer el número de dispositivos a los que les falta por escanearle el servicio OBEX. Para esta variable hemos utilizado un *java.util.concurrent.atomic.AtomicInteger* que nos garantiza que el incremento, decremento y comparación de su valor se realiza de forma atómica para todas las hebras que tienen acceso a la variable.

### 6.2.2.4 Interfaz

La aplicación Bluetooth de Blufeedme posee una interfaz de usuario realizada en java mediante elementos swing, destinada a las personas encargadas de realizar la auditoría de todo el proceso. Dicha interfaz muestra toda la información que la aplicación recopila en tiempo real y que el propio usuario puede visualizar, así como los errores o advertencias que están

## 6.2 .Subsistemas de Publicación de noticias

---

ocurriendo durante su ejecución.

La aplicación comienza mediante la clase BlufeedmeBT.java que contiene un método estático main, que se encarga únicamente de lanzar la interfaz contenida en la clase BlufeedmeBTView.java. Dicha clase inicializa todos los componentes gráficos de la interfaz, lanza el diálogo para obtener los distintos ficheros de configuración y si son correctos arranca la hebra principal DispositivoBT perteneciente a la lógica de la aplicación.

La comunicación entre la hebra DispositivoBT perteneciente a la lógica de la aplicación y la clase BlufeedmeBTView perteneciente a la vista, es realizada mediante un sistema de eventos personalizados que transmiten toda la información necesaria. La definición de los eventos se encuentra en el paquete eventos que es independiente al paquete vista, ya que no están relacionado obligatoriamente con él aunque la vista haga uso de ellos.

Para la creación y manipulación de eventos personalizados han sido creadas dos clases:

- **EventoDispositivo:** Esta clase implementa el tipo de evento personalizado que será utilizado para notificar un acontecimiento y almacenar toda su información asociada. Para ello, esta clase hereda del objeto de eventos genérico java.util.EventObject y añade algunos campos personalizados. Algunos de los campos añadidos son: Dispositivo del que trata, nombre del dispositivo, lista de noticias enviadas etc. Además poseerá un campo que indicará el tipo de acontecimiento que notifica dicho evento, los distintos tipos de acontecimientos son:
  - NUEVA\_BUSQUEDA. Notifica de que una nueva búsqueda de dispositivos ha comenzado.
  - DISPOSITIVO\_REGISTRADO\_ENCONTRADO. Notifica de que un dispositivo, previamente almacenado en la base de datos de la aplicación, ha sido detectado por la antena.
  - DISPOSITIVO\_ENCONTRADO. Notifica de que un nuevo dispositivo, no almacenado en la base de datos, ha sido detectado por la antena.
  - SERVICIO\_ENCONTRADO. Notifica de que un nuevo servicio ha sido encontrado.
  - NOTICIAS\_A\_ENVIAR. Avisa de las siguientes noticias que serán enviadas a un dispositivo.
  - NOTICIAS\_ENVIADAS. Notifica de las noticias que acaban de ser enviadas a un dispositivo.
  - NO\_VALIDO. Evento con un tipo no válido.
- **ListenerDispositivo:** Esta interfaz hereda de EventListener y define las operaciones que deben ser implementadas por un listener que reciba los eventos personalizados de la aplicación.

## 6.2 .Subsistemas de Publicación de noticias

---

Con estas dos clases permitimos que la vista se comunique con la capa lógica de forma independiente a través de la gestión de eventos, sin la necesidad de insertar código de la interfaz en la capa lógica. Las clases de la capa lógica tendrán una lista de tipo EventListenerList con todos los ListenerDispositivo suscritos. Éstos recibirán mediante eventos de tipo EventoDispositivo cada uno de los acontecimientos que sucedan durante la ejecución de la aplicación, por ejemplo, el escaneo de un nuevo dispositivo o un servicio. De este modo las clases lógicas notificarán mediante un evento a los listeners que han sido suscritos por las clases de la capa de interfaz y que podrán realizar los cambios pertinentes en la vista con la información obtenida del objeto evento.

Para lograr una mejora en el aspecto visual de la aplicación, se han utilizado una serie de clases que personalizarán el aspecto de los datos presentados en pantalla. Estas clases son:

- **RendererListaDispositivos:** Esta clase hereda de javax.swing.JLabel y implementa la interfaz javax.swing.ListCellRenderer para la personalización de las líneas que aparecen en la lista de dispositivos. Cada línea de esta lista podrá tener un color u otro dependiendo del tipo de dispositivo que represente.
- **RendererListaLog:** Esta clase hereda de javax.swing.JLabel y implementa la interfaz javax.swing.ListCellRenderer para la personalización de las líneas que aparecen en la lista mensajes de log. Cada línea de esta lista podrá tener un color u otro dependiendo de la gravedad de evento que haya ocurrido.
- **MiFormatter:** Esta clase hereda de java.util.logging.Formatter y se utiliza para personalizar el formato (fecha, ubicación de cada tipo de información etc.) de cada uno de los mensajes que son mostrados en la lista de logs.
- **ListStreamHandler:** Esta clase hereda de java.util.logging.StreamHandler. Se encarga de recibir las notificaciones de los objetos LOGGER y escribir dicha información en la lista de logs de la vista

## 7 MANUAL DE USUARIO

---

En este apartado se recoge el manual para la puesta en marcha del sistema: configuración, instalación, ejecución y mantenimiento; y una guía de usuario para cada uno de los componentes del sistema: Web Service y cliente, web de publicación de noticias y aplicación de envío de noticias mediante Bluetooth.

### ***7.1 Puesta en marcha y mantenimiento***

#### **7.1.1 Instalación/Configuración**

Para el despliegue del sistema se debe de disponer de los componentes hardware y software de la arquitectura descrita en el apartado 3.5 , Arquitectura Física o Hardware. Una vez se disponga de dicha arquitectura la instalación y configuración del sistema es sencilla.

##### **7.1.1.1 Base de datos**

Según dicha arquitectura el sistema contará con un servidor de persistencia, concretamente una base de datos MySQL. El primer paso para la implantación del sistema es crear la base de datos. Para crear toda la estructuras de la base de datos se dispone del archivo *blufeedme\_db.sql*. Este archivo incluye todas las sentencias SQL necesarias para la correcta creación de la base de datos centralizada que debe existir en el sistema.

##### **7.1.1.2 Servicio Web**

Una vez creada la base de datos, un segundo paso puede ser el despliegue del servicio web. En la arquitectura contamos con un servidor de aplicaciones Glassfish. Para poner en funcionamiento el servicio web basta con desplegar el archivo .war en dicho servidor de aplicaciones. El archivo a desplegar es *ServicioBlufeedme.war*.

Para configurar la conexión con el servidor de base de datos existe en el archivo empaquetado un archivo de propiedades de conexión para la Base de datos. Para establecer los parámetros bastaría con modificar dicho archivo. La estructura del archivo es:

```
URL:jdbc:mysql://localhost/blufeedme_db
user:root_blufeedme
```

## 7.1 .Puesta en marcha y mantenimiento

---

```
password:1234
```

Donde *URL* indica la cadena de conexión donde se encuentra el servidor de base de datos MySQL. *User* indica el usuario a emplear para establecer la conexión y *password* indica la contraseña de dicho usuario.

Glassfish permite subdividir las aplicaciones que están residentes en dominios. Con esta subdivisión lo que se pretende es agrupar aplicaciones que pueden compartir librerías, logs o configuraciones. Los dominios existentes se encuentran almacenados en *<glassfish\_home>/domains*. Estos tienen una “masterpassword” que se utiliza para administrar el dominio, los cuales son administrados, configurados y activados de manera independiente. Todos los dominios tienen una estructura bien definida en carpetas:

- config: Fichero de configuración.
- Libs: Librerías compartidas entre las aplicaciones del dominio.
- Applications: Donde están almacenadas la información de las aplicaciones.

Para levantar el servidor Glassfish cuenta con una consola de administración con una gran cantidad de funciones. La consola de administración se encuentra en *<glassfish\_home>/bin/asadmin*. Para levantar el servidor Glassfish por consola debemos indicarle que dominio queremos levantar (en caso de omisión, si hay solo uno, levanta éste). Una vez dentro de la consola escribimos:

```
start-domain [<nombre_del_dominio>]
```

Dicho comando acepta también un gran número de parámetros para definir puertos, claves, permisos, etc...

Una vez ejecutado el comando Glassfish nos informa del estado del proceso. Glassfish por defecto levanta un servidor RMI. La información más relevante del comando es la lista de puertos:

```
Domain listens on at least following ports for connections:  
[8080 8181 4848 3700 3820 3920 8686 ].
```

Ilustración 7.1.1: Lista de puertos Glassfish

Donde los más relevantes son el primero para HTTP (8080), el segundo HTTPS(8181) y el tercero que se corresponde con la administración (4848).

Si se desea crear un dominio para el despliegue del servicio web basta con escribir en la consola de administración:

```
create-domain - -user [nombre] - -adminport [puerto] [nombre_dominio]
```

Para detener el dominio ejecutamos la sentencia *stop-domain*

## 7.1 .Puesta en marcha y mantenimiento

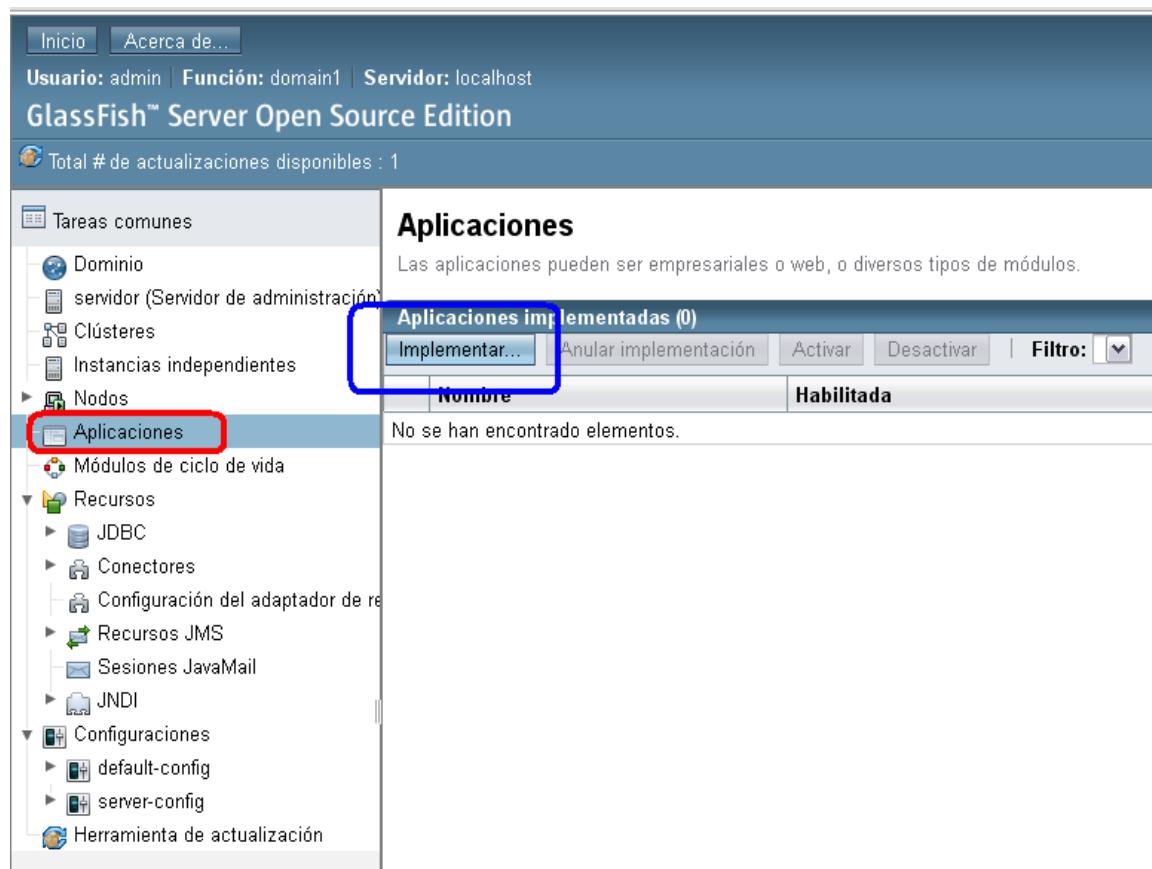
[nombre\_dominio].

Una vez levantado el dominio en Glassfish las nuevas aplicaciones se pueden desplegar en caliente mediante la web de administración. Para acceder a la web usamos la siguiente url:

`http://<ip del servidor>:<puerto de admin>/login.jsp`

El usuario/password por defecto si no a sido modificada es admin/adminadmin.

Para desplegar la aplicación en el menú lateral seleccionamos la opción *aplicaciones* y pulsamos el botón *implementar*:



Nombre	Habilitada

Ilustración 7.1.2: Web Administración Glassfish

Una vez pulsada la opción implementar aparece un dialogo en el que podemos elegir entre cargar el archivo en el servidor o usar la ruta local donde se encuentre el archivo empaquetado. En este caso, vamos a disponer del archivo empaquetado en el servidor por lo que seleccionamos la segunda opción:

## 7.1 .Puesta en marcha y mantenimiento

### Implementar aplicaciones o módulos

Especifique la ubicación de la aplicación o módulo que desea implementar. Una aplicación puede estar en un archivo comprimido o especificado como directorio.

Ubicación:  Archivo empaquetado que se cargará en el servidor

Archivo empaquetado local o directorio accesible desde GlassFish Server

Tipo: \*

Ilustración 7.1.3: Web Administración Glassfish (opción implementar)

Examinamos los directorios y buscamos el archivo empaquetado *ServicioBlufeedme.war*. La ruta del archivo empaquetado en el cdrom adjunto es *blufeedme\src\ServicioBlufeedme\dist\ServicioBlufeedme.war*.

**Servidor del navegador**

Nombre del servidor: d-ae792407942c4

Buscar en: c:\Documents and Settings\Dani\Escritorio\blufeedme\src\ServicioBlufeed

Filtro de archivos:

Para aplicar un valor nuevo en el campo, pulse Intro en ese campo.

Nombre	Tamaño	Fecha y hora
.svn\	0	24/06/2011 10:08
javadoc\	0	16/06/2011 17:53
<b>ServicioBlufeedme.war</b>	1620549	16/06/2011 17:53

Archivo seleccionado: c:\Documents and Settings\Dani\Escritorio\blufeedme\src\ServicioBlufeed

Ilustración 7.1.4: Web Administración Glassfish (opción implementar) cont.

## 7.1 .Puesta en marcha y mantenimiento

Una vez seleccionado el archivo aparecerán una serie de opciones de configuración para el despliegue de la aplicación, en este caso una aplicación web:

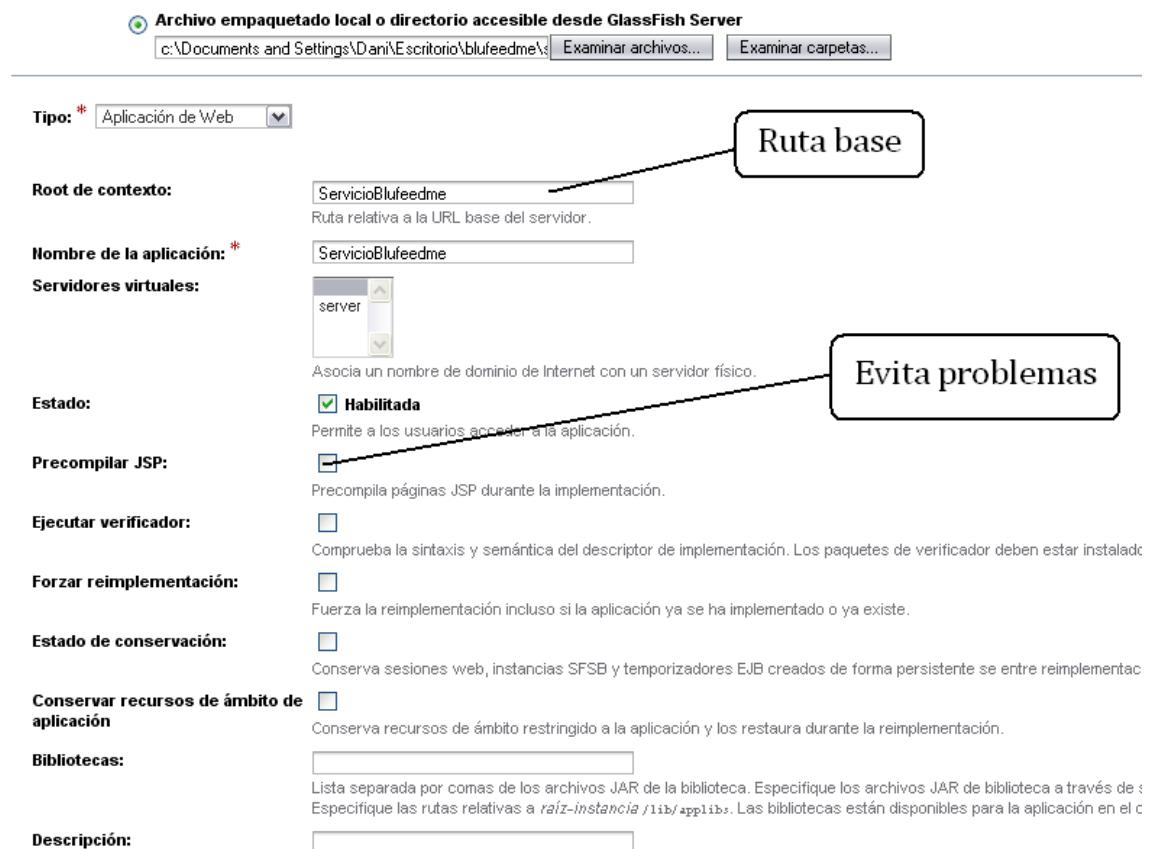


Ilustración 7.1.5: Web Administración Glassfish (opción implementar) cont. II

Seleccionadas las opciones y pulsando la opción aceptar nuestro servicio web estará activo y desplegado:



Ilustración 7.1.6: Web Administración Glassfish (opción implementar) cont. III

## 7.1 .Puesta en marcha y mantenimiento

---

Desplegado el servicio web ya podremos hacer uso del cliente. De nuevo, haciendo referencia a la arquitectura mostrada en 3.5 , Arquitectura Física o Hardware, disponemos de un servidor web Apache. En este servidor se alojarán todas las web que hay en el sistema: web para pantallas y cliente del servicio web.

Como se indicó en el diseño de la arquitectura, el cliente necesita una base de datos para llevar a cabo el control de sesiones. El servidor de base de datos que se usa para crearla es el mismo que almacena la base de datos central del sistema. Para crear la BD se facilita el archivo *clientblufeedme\_bd.sql*.

Creada la BD para el control de sesiones bastará con alojar el cliente en el lugar apropiado en el servidor web Apache. En caso de no disponer de la librería PHPLib es posible obtenerla en el cdrom adjunto al presente documento.

### 7.1.1.3 Web pantallas

El sistema web de pantallas se encontrará alojado en el servidor web Apache del que dispone el sistema. Para poner en funcionamiento la publicación por pantallas bastará con alojar en el lugar apropiado la plataforma web. De nuevo, es necesaria la instalación de la librería PHPLib, incluida en el cdrom adjunto al documento.

Para ejecutar la web pantallas en las pantallas tan solo bastaría con ejecutar en las mismas un navegador web.

En el caso de las pantallas, la animación se ejecutaría de forma automática, aunque puede intervenir el usuario.

### 7.1.1.4 Aplicación Bluetooth

La aplicación Bluetooth de Blufeedme se encuentra ubicada en el servidor Bluetooth como se indicó el apartado de Arquitectura física:. En este servidor estará instalada una máquina virtual Java y conectada la antena Bluetooth al equipo junto con los drivers provistos por el fabricante o por el propio sistema operativo. Los pasos a seguir para la instalación de la máquina virtual Java (JRE) pueden ser consultados en la siguiente dirección web [http://www.java.com/es/download/help/download\\_options.xml](http://www.java.com/es/download/help/download_options.xml).

La aplicación bluetooth no posee instalación, ya que únicamente consta de un fichero ejecutable llamado BlufeedmeBT.jar que podrá ser ejecutado mediante el comando “*java -jar BlufeedmeBT.jar*”. Además del ejecutable, serán necesarios los tres ficheros de configuración para la conexión con la base de datos, propiedades de la pila Bluetooth y plantilla html respectivamente. Tanto los ficheros de configuración como el propio ejecutable podrán situarse en la ruta del servidor que se desee. Únicamente se deberá indicar la situación de los distintos ficheros de configuración en el arranque de la aplicación java mediante los respectivos cuadros de diálogo que posee.

### **7.1.2 Mantenimiento**

Las principales tareas de mantenimiento están plenamente relacionadas con la gestión de la información de la Base de Datos centralizada.

El administrador del sistema deberá de registrar en la base de datos los gestores que harán uso del sistema, ya que no se permite a ningún usuario registrarse por sí mismo. Deberá asignarles usuario y password y, en el caso que lo precise, asociarles las categorías que gestionarán. Una vez registrados se les comunicará sus credenciales para acceder al sistema y estos podrán hacer uso del mismo. Durante la actividad del sistema toda esta información puede cambiar por lo que el gestor deberá de modificar los datos pertinentes en la base de datos: cambio de login, cambio de password, asociar/desasociar categorías a gestores, etc.

De igual forma deberá registrar las categorías que sean demandadas y eliminar las innecesarias.

Durante la actividad en el sistema la información de la base de datos cambia continuamente. Los gestores crearán noticias, registrarán dispositivos móviles, asociarán/desasociarán dispositivos a categorías, etc. Además de la información referente a los gestores y categorías, y las relaciones entre ambos, existe información adicional que el administrador debe gestionar. Se trata del historial de noticias enviadas a los dispositivos móviles. Cada vez que se envía una noticia a un dispositivo móvil se registra en la BD el envío, de forma que no se le volverá a enviar. Los datos de la tabla de la BD que almacena dicha información crecen de forma muy rápida. Por ese motivo, puede ser interesante que cada cierto tiempo se elimine del historial los envíos de aquellas noticias que ya estén caducadas (una vez caducadas no son enviadas), y que aún pue se desee que persistan en el sistema (cuando se elimina una noticia se elimina en cascada todas sus referencias).

La aplicación Bluetooth está configurada para que genere un máximo de 200 ficheros de logs incrementales, cada uno de ellos estará nombrado mediante el patrón de Blufeedme\_N.log, siendo N el orden de creación. Cada fichero de log podrá tener un tamaño máximo de 5MB, por lo que el tamaño máximo ocupado por la totalidad de los logs sería de 1000MB. Los logs son circulares, es decir, cuando se complete el fichero de log Blufeedme\_199.log, se sobrescribirá el fichero Blufeedme\_0.log con los nuevos datos y se continuará de forma ordenada. Por este hecho, el administrador deberá realizar un backup de los ficheros de logs generados por la aplicación Bluetooth antes de que sean sobrescritos si se quiere conservar dichos datos.

## 7.2 *Guía usuario*

En este apartado se mostrará un manual de usuario para cada una de las partes del sistema. Se plasmará la forma de interactuar con el sistema.

### 7.2.1 Web Service

En el capítulo 5.3 , Análisis y Diseño del sistema se listaron las operaciones disponibles en el servicio web. Para acceder a través de un sistema externo a cada una de las operaciones es necesario realizar una llamada mediante el protocolo SOAP. Si se va a integrar el servicio web en una plataforma web, es posible usar la clase cliente *clienteNewsService.php*, implementado en PHP y disponible en el cdrom adjunto. Veamos los detalles de cada una de las operaciones que ofrece el servicio web:

✓ **Añadir una noticia:**

```
Long addNoticia(String usuario, String firma,  
                 NoticiaWeb noticia)  
throws InvalidUserException, InvalidParamException
```

Añade una noticia al sistema. Los parámetros necesarios son:

- *usuario*: usuario gestor que va a crear la noticia
- *firma*: firma del mensaje para llevar a cabo la autentificación del usuario. La firma consiste en el SHA1 de la cadena formada por la concatenación de:

```
usuario + contraseña + noticia.autor + noticia.categoría +  
noticia.título + noticia.texto + noticia.subtitle +  
noticia.fechaPubli + noticia.fechaCaducidad
```

Donde *noticia.categoría* es el nombre de la categoría a la que pertenece, y las fechas siguen el formato "yyyy-MM-dd'T'HH:mm:ss".

- *noticia*: noticia a crear en el sistema.

El valor devuelto será el identificador de la noticia creada en caso de éxito, null en caso de error.

✓ **Eliminar una noticia:**

```
Boolean deleteNoticia(String usuario,  
                      String firma,  
                      Long id_noticia)  
throws InvalidUserException, InvalidParamException
```

Elimina una noticia del sistema. Los parámetros necesarios son:

- *usuario*: usuario que desea eliminar la noticia y por tanto gestor de la categoría a la que pertenece la noticia.
  - *firma*: firma del mensaje para llevar a cabo la autenticación del usuario. La firma consiste en el SHA1 de:  
usuario + contraseña + id\_noticia + selloTiempo
  - *id\_noticia*: identificador de la noticia a eliminar.
- El valor devuelto será TRUE en caso de éxito, FALSE en caso contrario.

✓ **Modificar una noticia:**

```
Boolean updateNoticia(String usuario, String firma, Long id_noticia,  
                      NoticiaWeb new_noticia)  
                      throws InvalidUserException,  
                      InvalidParamException
```

Modifica la información relativa a una noticia del sistema. Los parámetros necesarios son:

- *usuario*: usuario que desea modificar la noticia y por tanto gestor de la categoría de la noticia.
- *firma*: firma del mensaje. La firma consiste en el SHA1 de la cadena formada por la concatenación de:

usuario + contraseña + id\_noticia + new\_noticia.autor +  
new\_noticia.categoría + new\_noticia.título + new\_noticia.texto +  
new\_noticia.subtitle + new\_noticia.fechaPubli +  
new\_noticia.fechaCaducidad + selloTiempo

donde *new\_noticia.categoría* es el nombre de la categoría a la que pertenecerá la nueva noticia, y las fechas siguen el formato "yyyy-MM-dd'T'HH:mm:ss". El sello consiste en una marca de tiempo en el que se envió la petición al Web Service. El formato debe ser "yyyy-MM-dd'T'HH".

- *id\_noticia*: identificador de la noticia a modificar.
  - *new\_noticia*: noticia con la nueva información actualizada.
- El valor devuelto es TRUE en caso de éxito, FALSE en caso contrario.

## 7.2 .Guía usuario

---

### ✓ Obtener noticias:

```
ArrayList<blufeedme.modelo.Noticia>
getNoticias(java.lang.String usuario, String categoria,
            String firma)
            throws InvalidUserException
```

Obtiene las noticias correspondientes a la categoría/as que gestiona el usuario indicado. Parámetros:

- *usuario*: usuario del que se obtendrán las noticias
- *categoria*: categoría de la que se desea obtener las noticias.
- *firma*: firma del mensaje. La firma consiste en el SHA1 de la cadena resultante al concatenar:

usuario + contraseña + categoria

Donde *categoria* es el nombre de la categoría de la que se desean obtener las noticias.

La función devuelve la lista de noticias de la categoría indicada. En caso de que categoría sea null se devuelven todas las noticias de todas las categorías gestionadas por el usuario indicado. En caso de error devuelve null.

### ✓ Obtener categorías:

```
ArrayList<Categoria> getcategorias(String usuario,
                                      String firma)
                                      throws InvalidUserException
```

Obtiene las categorías gestionadas por el usuario indicado. Parámetros:

- *usuario*: usuario del que se desean obtener las categorías.
- *Firma*: firma del mensaje. La firma consiste en el SHA1 de la cadena formada por la concatenación de

usuario + contraseña.

La función devuelve la lista de categorías. En caso de error devuelve null.

### ✓ Asociar dispositivo a una categoría:

```
Boolean asociar(String usuario, String MAC, String categoria,
                  String firma)
                  throws InvalidUserException,
```

[InvalidParamException](#)

Asocia un dispositivo a una categoría. Una asociación provoca que se envíen las noticias de la categoría al dispositivo. Parámetros:

- *usuario*: usuario gestor de la categoría.
- *MAC*: dirección MAC del dispositivo.
- *categoria*: categoría a asociar.
- *firma*: firma del mensaje. La firma consiste en el SHA1 de la cadena formada por la concatenación de

usuario + contraseña + mac + categoria +  
selloTiempo

Donde *categoria* es el nombre de la misma.

La función devuelve TRUE en caso de éxito, FALSE en caso contrario.

✓ **Desasociar un dispositivo de una categoría:**

```
Boolean desasociar(String usuario, String MAC, String categoria,  
                   String firma)  
throws InvalidUserException,  
InvalidParamException
```

Deshace la asociación entre un dispositivo y una categoría. Una vez desasociados el dispositivo no recibirá noticias de la categoría.

Parámetros:

- *usuario*: usuario gestor de la categoría.
- *MAC*: dirección MAC del dispositivo.
- *categoria*: categoría a desasociar.
- *firma*: firma del mensaje. La firma consiste en el SHA1 de la cadena formada por la concatenación de

usuario + contraseña + mac + categoria + selloTiempo

Donde *categoria* es el nombre de la misma.

La función devuelve TRUE en caso de éxito, FALSE en caso contrario.

✓ **Añadir un dispositivo móvil:**

```
Long addDispositivoMovil(String usuario,  
                         Dispositivo dispositivo,  
                         String firma)
```

## 7.2 .Guía usuario

---

```
throws InvalidUserException
```

Da de alta un dispositivo en el sistema. Parámetros:

- *usuario*: usuario que da de alta el dispositivo. Debe ser el gestor de todas las categorías a las que está asociado el dispositivo.
- *dispositivo*: dispositivo a dar de alta en el sistema.
- *firma*: firma del mensaje. La firma consiste en el SHA1 de la cadena formada por la concatenación de

```
usuario + contraseña + mac + pin
```

Donde *mac* y *pin* son correspondientes al dispositivo pasado como parámetro

La función devuelve TRUE en caso de éxito, FALSE en caso contrario.

✓ **Eliminar un dispositivo móvil:**

```
Boolean  
deleteDispositivoMovil(String usuario, String MAC, String firma)  
throws InvalidUserException,  
InvalidParamException
```

Eliminar un dispositivo del sistema. El dispositivo no puede tener ninguna categoría asociada. Parámetros:

- *usuario*: usuario registrado en el sistema que elimina el dispositivo.
- *MAC*: dirección MAC del dispositivo.
- *firma*: firma del mensaje. La firma consiste en el sha1 de la cadena formada por la concatenación de

```
usuario + contraseña + mac + selloTiempo
```

Donde *mac* es la MAC pasada como parámetro.

La función devuelve TRUE en caso de éxito, FALSE en caso contrario.

✓ **Modificar un dispositivo móvil:**

```
Boolean updateDispositivoMovil(String usuario, String MAC,  
String MACN, String pinN,
```

## 7.2 .Guía usuario

---

```
String firma)
throws InvalidUserException,
InvalidParamException
```

Actualiza la información relativa a un dispositivo. Parámetros:

- *usuario*: usuario que modifica el dispositivo. Debe de ser el gestor de al menos una de las categorías a las que está asociado el móvil.
- *MAC*: dirección MAC del dispositivo a modificar.
- *MACN*: nueva dirección MAC del dispositivo.
- *pinN*: nuevo número pin del dispositivo.
- *firma*: firma del mensaje. La firma consiste en el sha1 de la cadena formada por la concatenación de

```
usuario + contraseña + MAC + MACN + pinN +
selloTiempo
```

*selloTiempo* es una marca temporal del momento en el que se realiza la llamada a la función del web service. El sello de tiempo debe tener el formato "yyyy-MM-dd'T'HH".

La función devuelve TRUE en caso de éxito, FALSE en caso contrario.

✓ **Obtener la lista de dispositivos:**

```
ArrayList<Dispositivo> getDispositivos(String usuario,
                                         String firma)
throws InvalidUserException,
InvalidParamException
```

Obtiene los todos los dispositivos registrados en la BD. Parámetros:

- *usuario*: usuario con el que se va a realizar la consulta.
- *firma*: firma del mensaje enviado. La firma consiste en el sha1 de la cadena formada por la concatenación de

```
usuario + contraseña
```

La función devuelve la lista de dispositivos registrados en el sistema o null en caso de error.

## 7.2 .Guía usuario

---

✓ **Obtener lista de dispositivos asociados a una categoría personal:**

```
ArrayList<Dispositivo> getMyDispositivos(String usuario,  
                                         String firma)  
                                         throws InvalidUserException
```

Obtiene los dispositivos registrados en la base de datos asociados a las categorías gestionadas por el usuario que realiza la consulta.  
Parámetros:

- *usuario*: usuario con el que se realiza la consulta.
- *firma*: firma del mensaje. La firma consiste en el sha1 de la cadena formada por la concatenación de

usuario + contraseña

La función devuelve la lista de dispositivos en caso de éxito, null en caso contrario.

✓ **Obtener lista dispositivos asociados a una categoría:**

```
ArrayList<Dispositivo>  
getDispositivosWhereCategoria(String usuario, String categoria,  
                               String firma)  
                               throws InvalidParamException,  
                                     InvalidUserException
```

Obtiene los dispositivos asociados a una determinada categoría.  
Parámetros:

- *usuario*: usuario que va a realizar la consulta.
- *categoria*: categoría de la que se desean obtener los dispositivos asociados.
- *firma*: firma del mensaje. La firma consiste en el sha1 de la cadena formada por la concatenación de

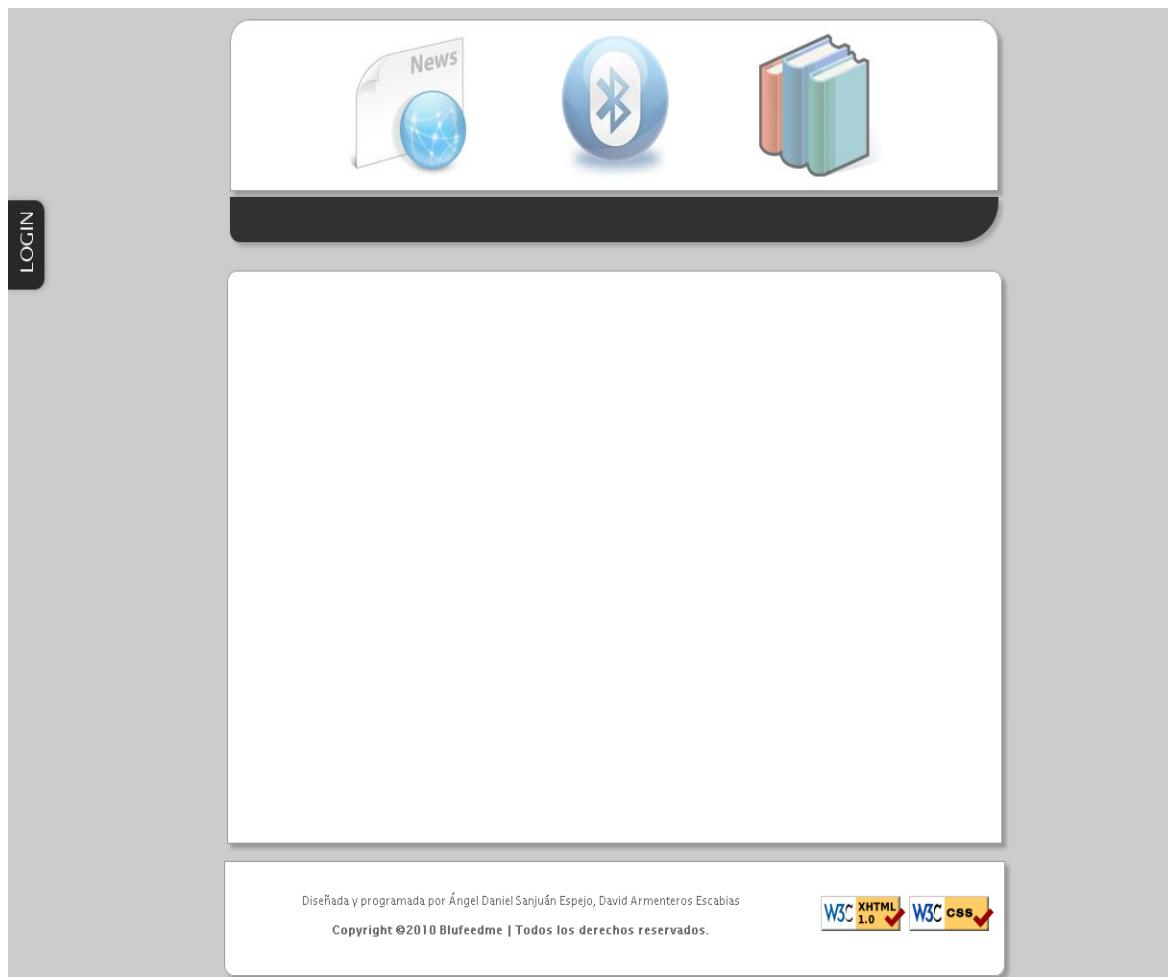
usuario + contraseña + categoria

Donde *categoria* es la categoría indicada como parámetro.

La función devuelve la lista de dispositivos asociados a la categoría indicada como parámetro, null en caso de error.

### **7.2.2 Cliente Web Service**

Cuando se accede a la plataforma web cliente del servicio web disponible en el sistema aparece una pantalla principal con el aspecto mostrado en la figura 7.2.1, Manual cliente Web Service; Pantalla principal.



*Ilustración 7.2.1: Manual cliente Web Service; Pantalla principal*

Lo primero que se debe hacer es logearse. Para ello dispone de un panel en el lateral izquierdo de la pantalla. Es necesario logearse porque cada una de las operaciones que efectuamos en el Web Service necesita autentificación, por lo que sería muy incomodo estar constantemente introduciendo usuario y password. Una vez logeado en el sistema aparecerá debajo del menú el login del usuario y la posibilidad de cerrar sesión e iniciar sesión con un usuario diferente.

## 7.2 .Guía usuario

---



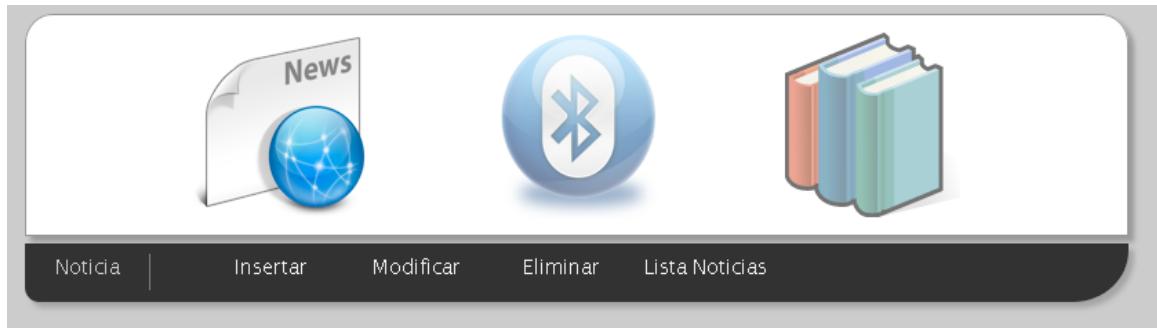
*Ilustración 7.2.2: Manual cliente Web Service; Login*

Se dispone de un menú principal por el que navegar para llevar a cabo diferentes operaciones. El menú principal está compuesto por iconos relacionados con las operaciones disponibles en el sistema:

- Un ícono para las operaciones sobre noticias.
- Un ícono para las operaciones sobre Bluetooth.
- Un ícono para las operaciones sobre categorías.

Para navegar por el menú simplemente basta con pasar el cursor por encima del ícono que corresponda a las operaciones que se desean realizar y estas aparecerán en el submenú horizontal situado justo debajo de los íconos.

Dicho menú tiene la apariencia mostrada en la figura Ilustración 7.2.2: Manual cliente Web Service; Login.



*Ilustración 7.2.3: Manual cliente Web Service; Menú*

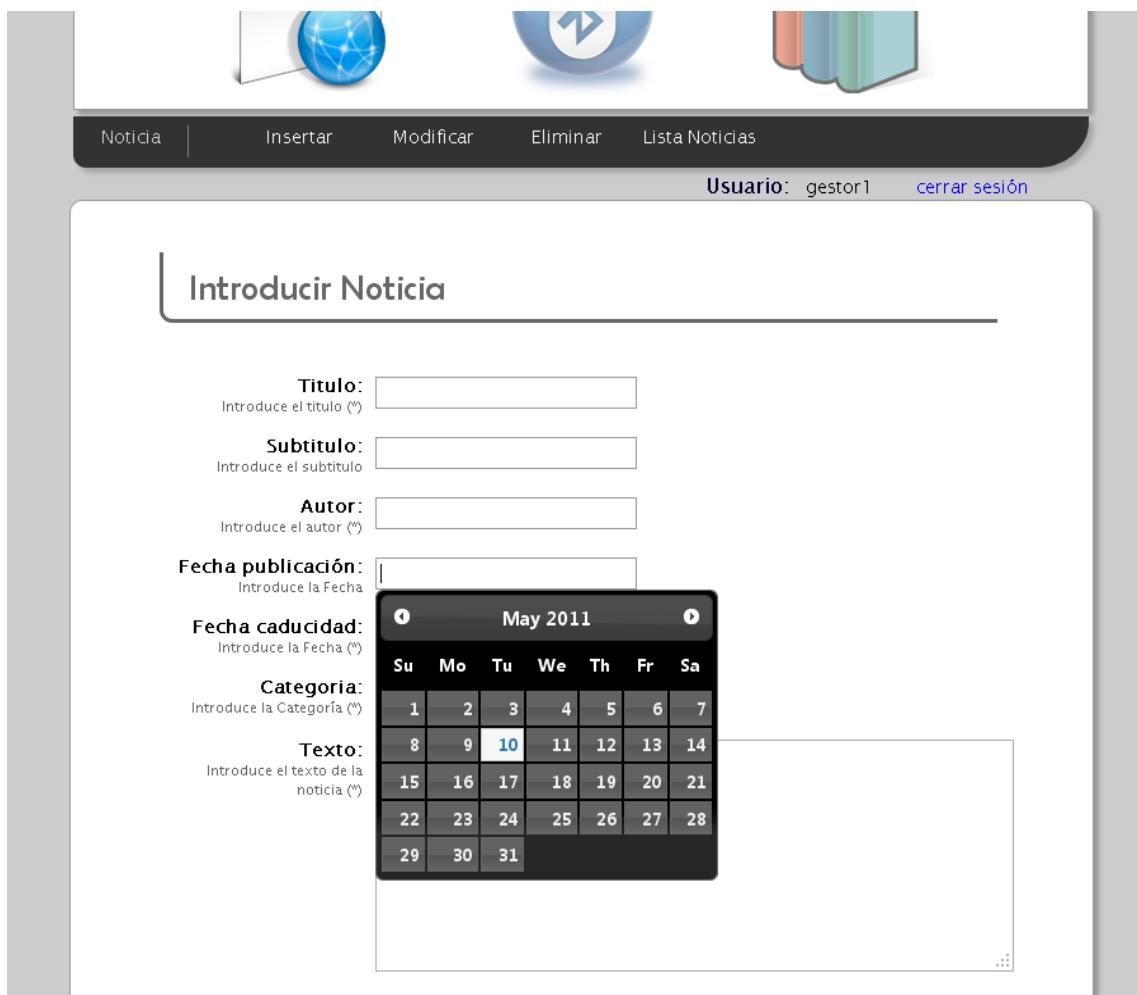
Mediante el submenú se acceden a cada una de las operaciones concretas disponibles. Cuando se selecciona una de las opciones del submenú, en el panel central aparece el formulario para introducir los datos necesarios. Veamos la interfaz para cada una de las funcionalidades disponibles y especificadas en el apartado 5.2.1, Especificación de requisitos.

## 7.2 .Guía usuario

- ◆ Operaciones con noticias:

Para acceder a las operaciones con noticias se debe seleccionar el primero de los tres grandes iconos. Las operaciones disponibles para las noticias son:

- Insertar Noticia:



The screenshot shows a web-based news insertion form. At the top, there are three large icons: a blue globe with a white flag, a blue arrow, and a blue and green shield. Below the icons is a navigation bar with links: Noticia, Insertar, Modificar, Eliminar, and Lista Noticias. On the right of the navigation bar are the user information 'Usuario: gestor1' and a 'cerrar sesión' (log out) link. The main content area is titled 'Introducir Noticia'. It contains several input fields with placeholder text: 'Título:' (Introduce el título (\*)), 'Subtítulo:' (Introduce el subtítulo), 'Autor:' (Introduce el autor (\*)), 'Fecha publicación:' (Introduce la Fecha), 'Fecha caducidad:' (Introduce la Fecha (\*)), 'Categoría:' (Introduce la Categoría (\*)), and 'Texto:' (Introduce el texto de la noticia (\*)). A date picker calendar is overlaid on the 'Fecha publicación:' field, showing the month of May 2011. The calendar grid is as follows:

Su	Mo	Tu	We	Th	Fr	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

Ilustración 7.2.4: Manual cliente Web Service: Formulario Introducir Noticia

En este caso el formulario consta de los campos necesarios para poder insertar una noticia:

- Título: campo para poder introducir el título de la noticia.
- Subtítulo: campo para introducir el subtítulo de la noticia.
- Autor: campo para introducir el autor de la noticia.
- Fecha publicación: campo para indicar la fecha en la que se desea que la noticia sea publicada. Para facilitar la inserción de este campo se muestra un calendario, de forma que, al picar sobre el

## 7.2 .Guía usuario

día deseado ser rellena el campo de forma automática.

- Fecha caducidad: campo para indicar la fecha en la que la noticia dejará de ser publicada por los distintos medios de publicación disponibles en el sistema. Al igual que con el campo fecha de publicación, se facilita la tarea mostrando un calendario.
- Categoría: campo para indicar la categoría a la que será asociada la noticia. Este campo ya contiene la lista de categorías posibles a asignar en función del usuario que ha iniciado la sesión.
- Texto: es el campo para poder introducir todo el cuerpo de la noticia.

Una vez introducidos los campos (obligatorios campos con marca (\*)) pulsar el botón *Aceptar*.

- Modificar Noticia:

Noticia | Insertar | Modificar | Eliminar | Lista Noticias

Usuario: gestor1 | cerrar sesión

### Modificar Noticia

**ID Noticia:** 1

Introduce el identificador de la noticia a modificar (\*)

**Título:** Solicitud de adaptación a los nuevos

Introduce el nuevo título (\*)

**Subtítulo:** Subtitulo de la noticia

Introduce el nuevo subtítulo

**Autor:** Secretaría

Introduce el nuevo autor (\*)

**Fecha publicación:** 2010-11-10

Introduce la nueva fecha de publicación (\*)

**Fecha caducidad:** 2013-12-16

Introduce la nueva fecha de caducidad (\*)

**Categoría:** UGR

Introduce la nueva categoría (\*)

**Texto:**

Introduce el nuevo texto (\*)

Se ha abierto un plazo preferente para solicitar la adaptación a los nuevos grados hasta el 29 de Julio de 2010. Los alumnos de este centro interesados pueden iniciar el trámite a través de la siguiente aplicación:  
<https://etsiit.ugr.es/app/static/SolicitadorDeCambioAGrado>  
E-mail de contacto: jbernier@ugr.es

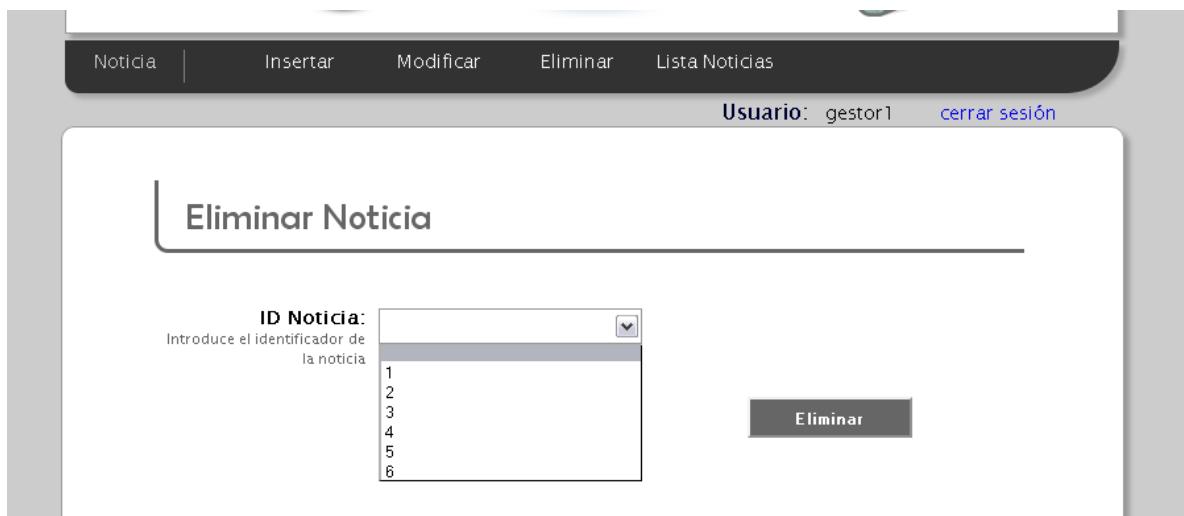
Modificar

Ilustración 7.2.5: Manual cliente Web Service; Formulario Modificar Noticia

## 7.2 .Guía usuario

Para modificar una noticia es necesario seleccionar a través del primer campo del formulario, *id noticia*, la noticia a modificar. Una vez seleccionada automáticamente se rellenan todos los campos del formulario con la información actual de la noticia a modificar. Tras modificar los campos pulsar el botón *Modificar*.

- Eliminar Noticia:



Noticia | Insertar | Modificar | Eliminar | Lista Noticias

Usuario: gestor1 | cerrar sesión

### Eliminar Noticia

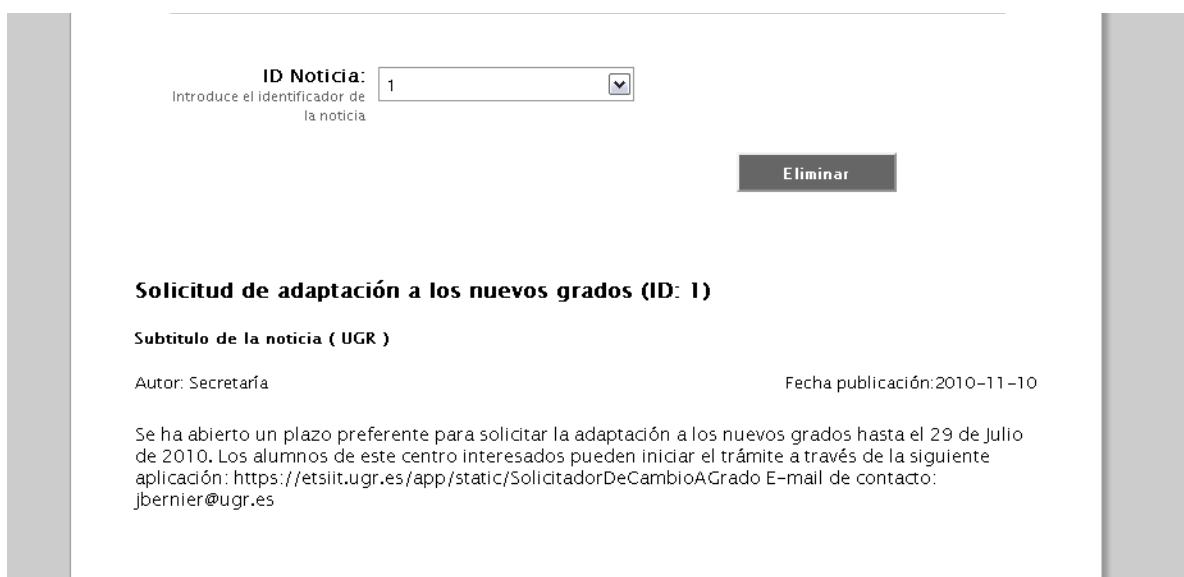
ID Noticia: Introduce el identificador de la noticia

1  
2  
3  
4  
5  
6

Eliminar

Ilustración 7.2.6: Manual cliente Web Service; Formulario Eliminar Noticia

A la hora de eliminar una noticia basta con indicar el identificador de la noticia. Una vez seleccionada la interfaz mostrará la información de la noticia que ha seleccionado. De esta forma podrá verificar que se trata de la noticia deseada.



ID Noticia: Introduce el identificador de la noticia

1

Eliminar

**Solicitud de adaptación a los nuevos grados (ID: 1)**

**Subtítulo de la noticia ( UGR )**

Autor: Secretaría Fecha publicación: 2010-11-10

Se ha abierto un plazo preferente para solicitar la adaptación a los nuevos grados hasta el 29 de julio de 2010. Los alumnos de este centro interesados pueden iniciar el trámite a través de la siguiente aplicación: <https://etsiit.ugr.es/app/static/SolicitadorDeCambioAGrado> E-mail de contacto: [jbernier@ugr.es](mailto:jbernier@ugr.es)

Ilustración 7.2.7: Manual cliente Web Service; Formulario Eliminar Noticia(continuación)

## 7.2 .Guía usuario

Una vez verificada la información pulsar el botón *Eliminar*.

- **Lista Noticias:**



Ilustración 7.2.8: *Manual cliente Web Service; Lista Noticias*

Se trata de un carrusel que muestra todas las noticias. En primera instancia muestra el título y subtítulo de la noticia. Si se desea obtener más información de una de las noticias bastaría con hacer click sobre la noticia y el carrusel desplegará toda la información.

## 7.2 .Guía usuario

- ◆ Bluetooth:
  - Introducir Dispositivo:

Introducir Dispositivo Bluetooth

MAC:  Introduce la MAC (^)

PIN:  Introduce el PIN (^)

Registrar

Dispositivos existentes en el sistema

00:23:AF:CD:2F:29
UGR (Universidad de Granada -- <a href="http://www.ugr.es">www.ugr.es</a> )
Secretaría E.T.S.I. (Secretaría de la Escuela Técnica Superior Ingenierías Informática y Telecomunicación -- <a href="http://etsiit.ugr.es">etsiit.ugr.es</a> )
ISI (Departamento de Lenguajes y Sistemas Informáticos -- <a href="http://isi.ugr.es">isi.ugr.es</a> )

Ilustración 7.2.9: Manual cliente Web Service; Formulario Introducir Dispositivo

El formulario muestra los campos necesarios para introducir un dispositivo móvil en el sistema:

- MAC: campo para introducir la dirección ethernet del dispositivo móvil que se va a registrar.
- PIN: campo para indicar el código que se usará para realizar el pairing en las conexiones que se establecerán entre el dispositivo y el sistema para el envío de noticias.

Además de los campos mencionados, la interfaz muestra los dispositivos existentes en el sistema (Todos independientemente del gestor). Para cada uno de los dispositivos muestra la lista de categorías a las que se encuentra asociado en ese momento.

## 7.2 .Guía usuario

Una vez introducidos los campos pulsar el botón *Registrar*.

- **Modificar:**

Modificar Dispositivo Bluetooth

**MAC:** 00:23:AF:CD:2F:29  
Introduce la MAC del dispositivo (\*)

**Nueva MAC:**   
Introduce la nueva MAC (\*)

**Nuevo pin:**   
Introduce el nuevo PIN (\*)

**Modificar**

Ilustración 7.2.10: Manual cliente Web Service; Formulario Modificar Dispositivo

Para modificar un dispositivo se debe seleccionar uno de los dispositivos registrados en el sistema sobre los que el gestor que ha iniciado sesión puede realizar operaciones. Una vez seleccionado el dispositivo basta con indicar los datos a modificar:

- Nueva MAC: campo para indicar la nueva dirección ethernet que se desea asignar al dispositivo móvil.
- Nuevo pin: campo para introducir el nuevo código que se usará para establecer la conexión entre el dispositivo móvil y el sistema.

Una vez introducida toda los datos pulsar el botón *Modificar*.

- **Eliminar:**

Eliminar Dispositivo Bluetooth

**MAC:**   
Introduce la MAC (\*)

00:23:AF:CD:2F:29  
4C:54:99:06:4A:4D

**Eliminar**

Ilustración 7.2.11: Manual cliente Web Service; Formulario Eliminar Dispositivo

## 7.2 .Guía usuario

Para eliminar un dispositivo móvil se debe seleccionar uno de los existentes en el campo MAC. Una vez seleccionado pulsar el botón *Eliminar*.

- Asociar:

Noticia | Insertar | Modificar | Eliminar | Lista Noticias

Usuario: gestor1 | cerrar sesión

### Asociar DispositivoBluetooth-Categoría

MAC:

Categoría:

**Asociar**

**Dispositivos existentes en el sistema**

00:23:AF:CD:2F:29
UGR (Universidad de Granada -- <a href="http://www.ugr.es">www.ugr.es</a> )
Secretaría E.T.S.I. (Secretaría de la Escuela Técnica Superior Ingenierías Informática y Telecomunicación -- <a href="http://etsiit.ugr.es">etsiit.ugr.es</a> )

Ilustración 7.2.12: *Manual cliente Web Service: Formulario Asociar Dispositivo*

El formulario muestra la lista de dispositivos existentes en el sistema, campo MAC.

Para indicar la categoría a la que se desea asociar el dispositivo móvil la interfaz muestra una segunda lista, campo Categoría, en la que aparecen solamente las categorías que son gestionadas por el gestor con el que se ha iniciado la sesión.

Además, la interfaz muestra los dispositivos existentes asociados a categorías del gestor que va a realizar la operación. Para cada dispositivo muestra las categorías a las que está asociado actualmente.

Una vez introducidos los datos pulsar el botón *Asociar*.

## 7.2 .Guía usuario

- Desasociar:

The screenshot shows a web-based application interface for managing device associations. At the top, there is a navigation bar with links: 'Dispositivo' (selected), 'Introducir', 'Modificar', 'Eliminar', 'Asociar Categoría', and 'Desasociar Categoría'. The user is identified as 'Usuario: gestor1' with a 'cerrar sesión' (Logout) link. The main content area is titled 'Desasociar DispositivoBluetooth-Categoría'. It contains two dropdown menus: 'MAC' (with placeholder 'Introduce la MAC del dispositivo(必)') and 'Categoría' (with placeholder 'Introduce la categoría(必)'). A 'Desasociar' (Unpair) button is located below these fields. Below this form, a section titled 'Mis dispositivos registrados' (My registered devices) lists two entries: '00:23:AF:CD:2F:29' (UGR (Universidad de Granada -- www.ugr.es)) and '4C:54:99:06:4A:4D'.

Ilustración 7.2.13: Manual cliente Web Service; Formulario Desasociar Dispositivo

Al igual que el caso anterior, el formulario muestra dos campos:

- MAC: lista con las direcciones ethernet de los dispositivos registrados en el sistema que están asociados a alguna de las categorías gestionadas por el gestor con el que se ha iniciado la sesión.
- Categoría: lista con las categorías a las que está asociado el dispositivo móvil seleccionado mediante el campo MAC del formulario.

Como en el resto de operaciones, se mostrarán los dispositivos registrados que están asociados a categorías del gestor y las categorías a las que se encuentra asociado el dispositivo pertenecientes, claro está, al gestor que realiza la operación.

Una vez introducidos los datos pulsar el botón *Desasociar*.

## 7.2 .Guía usuario

---

- ◆ Categoría:
  - Lista categorías:



*Ilustración 7.2.14: Manual cliente Web Service; Formulario Lista Categorías*

Mostrará un carrusel con título y descripción de cada categoría gestionada por el gestor. En primera instancia se mostrarán solo los títulos de las categorías. Para poder obtener la información de la descripción de la categoría será necesario hacer click sobre la categoría deseada.

### **7.2.3 Pantallas**

La web para la publicación de noticias por las pantallas es dinámica. Actualiza la información de forma periódica y automática y muestra y cada una de las noticias a publicar.

La web permite al usuario interactuar, de forma que puede detener la animación y explorar las noticias de forma personal, volver a reanudar la animación automática de la web, y modificar la velocidad de la animación que muestra las sucesivas noticias.

Para detener/reanudar la animación basta con colocar el cursor sobre/fuera de alguna de las noticias publicadas en el carrusel.

Para modificar la velocidad de la animación se debe usar la rueda de scroll.

## 7.2.4 Bluetooth

### 7.2.4.1 Administrador

La aplicación Bluetooth se encuentra en el fichero ejecutable BlufeedmeBT.jar que podrá ser lanzada a través del comando:

**“java -jar BlufeedmeBT.jar”.**

Una vez ejecutada la aplicación se nos mostrará la siguiente pantalla:



Ilustración 7.2.15: Pantalla ficheros de propiedades bluetooth

En esta ventana se solicitan tres ficheros que son de suma importancia para el funcionamiento de la aplicación, como son:

- fichero de configuración de la base de datos: en este fichero se define el nombre u ruta de la base de datos, el usuario con el que accederá la aplicación y su contraseña. Un fichero de ejemplo sería:

```
URL:jdbc:mysql://localhost/blufeedme_db
user:root_blufeedme
password:1234
```

- fichero de configuración de la pila Bluetooth: en este fichero se indican las propiedades de la pila Bluetooth, como son los distintos timeouts de conexión, el número máximo de conexiones simultáneas etc. Un fichero de ejemplo sería:

```
PROPERTY_OBEX_TIMEOUT:180000
PROPERTY_INQUIRY_DURATION_DEFAULT:20
PROPERTY_INQUIRY_DURATION:20
PROPERTY_CONNECT_UNREACHABLE_RETRY:3
PROPERTY_CONNECT_TIMEOUT:180000
```

## 7.2 .Guía usuario

- fichero de plantilla para el envío de noticias: en este fichero se almacena la plantilla html que servirá como base para la generación del fichero de noticias para los dispositivos.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
    <title>Noticias UGR</title>
    <style type="text/css"> /* Código para el estilo */</style>
  </head>
  <body>
```

Una vez seleccionados los tres ficheros de configuración, se deberá pulsar el botón de aceptar y se nos mostrará la siguiente ventana:

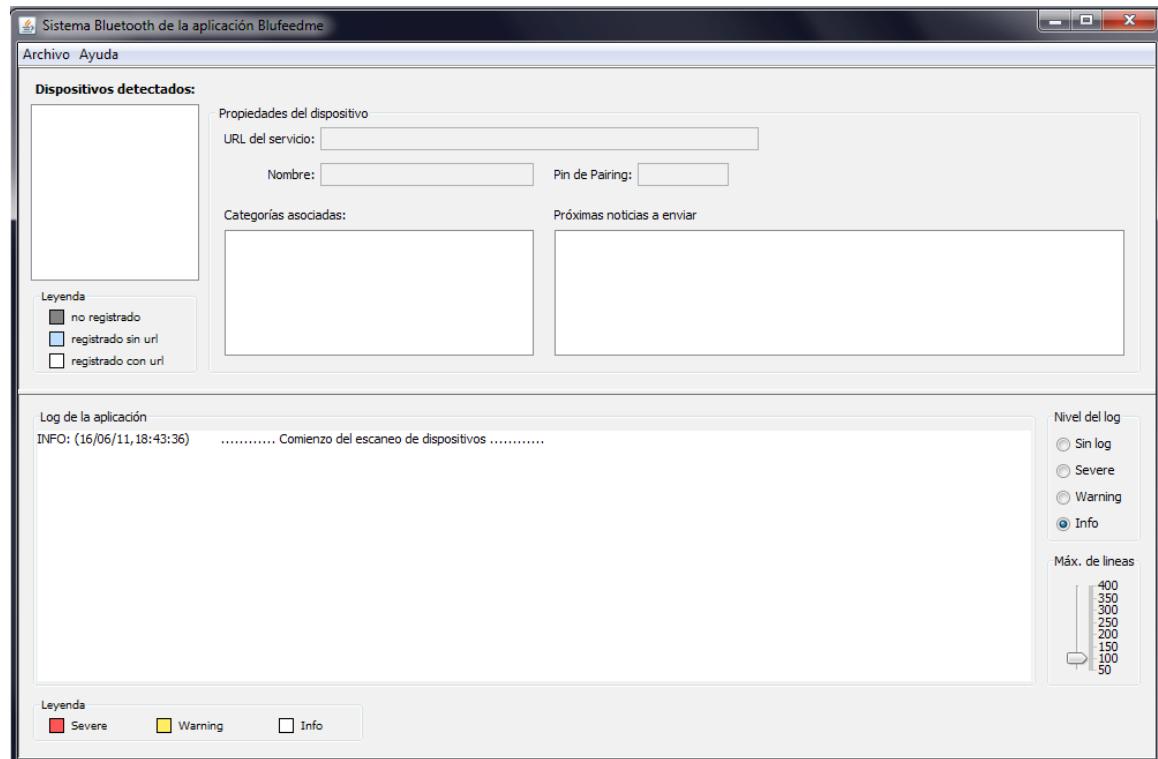


Ilustración 7.2.16: Pantalla de auditoría del proceso de envío bluetooth

En esta ventana será donde se reflejarán todos los dispositivos que están siendo escaneados en tiempo real por la antena Bluetooth y la información

## 7.2 .Guía usuario

---

proporcionada por la base de datos.

- En la sección superior aparece un listado con las direcciones MAC de cada uno de los dispositivos detectados. Cada línea estará coloreada con alguno de los tres colores indicados en la leyenda (gris, azul, blanco) dependiendo del modo en el que esté el dispositivo almacenado en la base de datos:
  - Gris: Dispositivo no registrado en la base de datos. Se le enviarán únicamente las noticias más importantes ya que no posee categoría de noticias asignadas.
  - Azul: Dispositivo registrado pero sin URL de conexión conocida. Se trata de un dispositivo cuya MAC está registrada en el sistema, sin embargo, no se conoce la URL de conexión para el servicio OBEX PUSH por lo que tendrá que realizarse una búsqueda de sus servicios disponibles.
  - Blanco: Dispositivo registrado y con URL de conexión conocida. Se trata de un dispositivo registrado en el sistema y del que además poseemos la URL del servicio OBEX PUSH.

A la derecha del listado aparecen una serie de campos que nos mostrarán todos los datos relativos al dispositivo seleccionado en la lista de la izquierda. Se mostrará el nombre del dispositivo (en caso de que posea), el URL de conexión (si ya disponemos de él), el PIN de conexión (es 0000 para dispositivos no registrados) y un listado con las categorías asociadas junto con las noticias que están pendientes de envío al dispositivo.

- En la sección inferior aparece una lista con la traza de todas las operaciones del sistema. En dicha lista se irán añadiendo líneas conforme vayan sucediendo eventos en el sistema . Cada una de las líneas poseerá uno de los colores que aparecen en la leyenda inferior, indicando así la gravedad del evento. Los eventos en amarillo indican que ha habido algún tipo de inconveniente pero que no posee excesiva importancia, por ejemplo, se ha cumplido algún timeout para una transacción o habido un error al obtener el nombre del dispositivo. Los eventos en rojo mostrarán los errores importantes como puede ser un fallo durante el envío del fichero de noticias, sin embargo, el sistema tendrá en cuenta dicho fallo y retransmitirá aquellos ficheros que no se hayan completado satisfactoriamente.

Las opciones de la derecha sirven para filtrar o modificar la visualización de la lista de eventos, indicando el número máximo de líneas almacenadas o mostrando únicamente aquellos eventos que posean un nivel de prioridad por debajo del indicado.

En la lista de logs aparecerá una barra de scroll si se supera el número de entradas que pueden ser visibles en la zona dedicada para ello. El programa permite avanzar dicha lista de forma automática si no hay ninguna fila seleccionada, de este modo, se visualizarán siempre los últimos eventos ocurridos. Si se selecciona una o más filas de la lista la función de autoscroll se detendrá.

A parte del listado de eventos parcial obtenido en pantalla, recordamos que todas las operaciones y eventos serán almacenadas simultáneamente de forma permanente en los ficheros de logs (en formato xml) establecidos por la aplicación.

Para finalizar la ejecución del programa se deberá picar sobre la cruz de la esquina superior derecha o elegir la opción salir en la barra superior de tareas.

### 7.2.4.2 Usuario

El usuario de la aplicación será cualquier persona que disponga de un dispositivo móvil con Bluetooth y que se encuentre en el radio de cobertura de las antenas emisoras de Blufeedme. Para que una persona reciba las noticias más importantes o personalizadas, en caso de estar registrado, deberá activar el Bluetooth en su dispositivo móvil en primer lugar y establecerlo como visible durante el período de tiempo en el que quiera recibir las noticias. Una vez que está activado el Bluetooth y se ha hecho visible, se recibirá una notificación en el dispositivo móvil, preguntándole si autoriza la conexión con el Hot-Spot. Si acepta la conexión se le pedirá, por motivos de seguridad, que introduzca un código PIN para la realización del pairing entre ambos dispositivos. Dicho código PIN será la combinación que se estableciera en su registro como usuario de Blufeedme o la combinación **0000** para dispositivos que no estén registrados en el sistema. Si el código PIN introducido es correcto se le transferirá a su dispositivo móvil un fichero con las noticias pertinentes, para ello deberá autorizar la recepción de dicho fichero.

Los pasos de forma resumida son los siguientes:

- Activar Bluetooth en dispositivos
- Hacer visible el dispositivo a través de bluetooth.
- Aceptar la petición de conexión.
- Ingresar código su código PIN o 0000 si no está registrado.
- Aceptar la recepción del fichero de noticias.

Una vez recibido el fichero html con la lista de noticias, podrá visualizarlo con la aplicación destinada a tal fin que tenga instalada en su dispositivo móvil.

## 8 CONCLUSIONES

BLUFEEDME es el resultado de 8 meses de trabajo, de un desarrollo en el que se han aplicado muchos de los conocimientos, en materia de Ingeniería del Software y otras muchas disciplinas, adquiridos durante los años de estudio de Ingeniería Informática en la Escuela Técnica Superior de Ingeniería Informática y de Telecomunicación.

El sistema obtenido final satisface los objetivos principales planteados para la publicación y divulgación de noticias: integrabilidad, accesibilidad y sistema cercano al usuario. Por tanto, podemos decir que el sistema aporta la solución al problema actual de publicación de noticias, convirtiéndose en una alternativa válida a los sistemas de publicación y divulgación de noticias actuales. No obstante, todo es mejorable y en nuestro caso concreto, todavía queda trabajo por hacer para aumentar las funcionalidades ofrecidas, consiguiendo un sistema más flexible. Por ejemplo, añadir jerarquización en las categorías de noticias, la obtención de noticias desde el dispositivo móvil bajo demanda o parallelización entre varios Hotspot situados una misma Java Virtual Machine (Piconet).

En el desarrollo del sistema se toma como premisa el utilizar, exclusivamente, herramientas libres o, en su defecto, gratuitas para intentar abaratar los costes de desarrollo lo máximo posible. Este objetivo se cumple en todo momento.

Además de la premisa de utilizar herramientas libres, o en su defecto, gratuitas, el software que se desarrolla, se desarrolla con la idea de, tras su finalización, poder proceder a su liberación. El sistema desarrollado, finalmente, emplea licencia GPL v3, o en su defecto licencias compatibles, caso de Apache v2, y creando toda la estructura necesario para ponerlo a disposición de la comunidad. Por tanto, satisface otro de los principales objetivos planteados al comienzo del proceso de desarrollo del software.

No solo cumple con los principales objetivos citados, el sistema obtenido satisface el resto de objetivos fijados y funciona tal y como se planeó al comienzo del desarrollo: mejorando en muchos aspectos las expectativas que se tenían.

Desde el punto de vista personal de los autores, el desarrollo de un proyecto de una amplitud y complejidad considerable, ha supuesto un gran reto personal, ya que hasta la presente no nos habíamos visto embarcados en un proyecto de la envergadura de BLUFEEDME, produciéndonos una gran satisfacción poder llegar a una solución válida y alternativa a los sistemas actuales.

El resultado final, no obstante, bien ha merecido el esfuerzo y la

## 8 .Conclusiones

dedicación invertidos. Resaltando también, desde un punto de vista didáctico, que el desarrollo de BLUFEEDME ha servido para conocer, aprender y utilizar nuevas herramientas y tecnologías que se desconocían antes de comenzar.

## 9 ANEXOS.

### ***9.1 Anexo I: Descripción detallada de los casos de uso***

Con la descripción detallada de los casos de uso, se pretende mostrar, con el mayor nivel de detalle posible, qué es lo que hace un caso de uso sin preocuparse de cómo lo hace. Esta descripción detalla consiste en narrar cuáles son las secuencias de interacciones, entre el usuario y el sistema, que se llevarán a cabo en los distintos escenarios que puedan presentarse durante la ejecución del caso de uso.

En UML no existe un estándar para una especificación de casos de uso. Sin embargo, con el objetivo de llenar este aspecto, existen multitud de plantillas propuestas por diferentes autores. En este caso, para la descripción de los casos de uso reflejados en los diagramas anteriores, utilizaremos la plantilla descrita a continuación:

<b>NOMBRE DEL CASO DE USO</b>
...
<b>RESUMEN</b>
...
<b>DEPENDENCIAS</b>
...
<b>ACTORES</b>
...
<b>PRECONDICIONES</b>
...
<b>POSTCONDICIONES</b>
...
<b>CURSO NORMAL</b>
<ol style="list-style-type: none"> <li>1. ...</li> <li>2. ...</li> <li>3. ...</li> </ol>
<b>CURSOS ALTERNATIVOS</b>
...
<b>OBSERVACIONES</b>
...

Tabla 4: Plantilla descripción Casos de Uso

Procedamos a la descripción detallada de cada uno de los casos de uso existentes en el sistema.

### 9.1.1 Subsistema Web

<b>NOMBRE DEL CASO DE USO</b>
Crear noticia
<b>RESUMEN</b>
Este caso de uso permite crear una nueva noticia en el sistema.
<b>DEPENDENCIAS</b>
Este caso de uso incluye el caso de uso <i>Validar Usuario</i> .
<b>ACTORES</b>

## 9.1 .Anexo I: Descripción detallada de los casos de uso

Gestor (sistema externo)
<b>PRECONDICIONES</b>
-
<b>POSTCONDICIONES</b>
Se ha creado una nueva noticia en el sistema.
<b>CURSO NORMAL</b>
<p>1 . El gestor envía al sistema la información de la nueva noticia a crear en el sistema (titulo, subtítulo, autor, categoría, texto, fecha publicación, fecha caducidad). Además de la información indicada el usuario indicará su lógín y firmará el mensaje. La firma del mensaje consistirá en el resultado de cifrar mediante <i>SHA</i> la cadena:</p> <p>login + password + autor + categoría + titulo + texto + subtítulo + fecha publicación + fecha caducidad</p> <p>2 . El sistema comprueba si el usuario es correcto. Se ejecuta el caso de uso <i>ValidarUsuario</i>.</p> <p>3 . El usuario es correcto el sistema comprueba que los parámetros son correctos.</p> <p>4 . Los parámetros son correctos y crea la nueva noticia. El sistema devuelve el identificador de la noticia creada en el sistema.</p>
<b>CURSOS ALTERNATIVOS</b>
3 . El usuario no es correcto. El sistema informa del error. Finaliza el caso de uso.
4. Los parámetros no son correctos. El sistema informa del error. Finaliza el caso de uso.
<b>OBSERVACIONES</b>
-

Tabla 5: Descripción caso de uso: Crear Noticia

<b>NOMBRE DEL CASO DE USO</b>	
	Modificar noticia
<b>RESUMEN</b>	
	Este caso de uso permite modificar una noticia existente en el sistema.
<b>DEPENDENCIAS</b>	
	Este caso de uso incluye el caso de uso <i>Validar Usuario</i>
<b>ACTORES</b>	
	Gestor (sistema externo)
<b>PRECONDICIONES</b>	
	-
<b>POSTCONDICIONES</b>	
	La noticia quedará modificada en el sistema.
<b>CURSO NORMAL</b>	
	<ol style="list-style-type: none"> <li>1. El gestor indica el identificador de la noticia a modificar y los parámetros a asignar a la noticia (titulo, subtítulo, autor, categoría, texto, fecha publicación, fecha caducidad). Además de la información indicada el usuario indicará su lógin y firmará el mensaje. La firma del mensaje consistirá en el resultado de cifrar mediante <i>SHA</i> la cadena:</li> </ol> <pre>login + password + identificador + autor + categoría + título + texto + subtítulo + fecha publicación + fecha caducidad</pre> <ol style="list-style-type: none"> <li>2. El sistema comprueba si el usuario es correcto. Se ejecuta el caso de uso <i>ValidarUsuario</i>.</li> <li>3. El usuario es correcto el sistema comprueba que los parámetros son correctos.</li> <li>4. Los parámetros son correctos y modifica la noticia. El sistema indica que la noticia ha sido modificada.</li> </ol>
<b>CURSOS ALTERNATIVOS</b>	
	<ol style="list-style-type: none"> <li>3 . El usuario no es correcto. El sistema informa del error. Finaliza el caso de uso.</li> <li>4. Los parámetros no son correctos. El sistema informa del error. Finaliza el caso de uso.</li> </ol>

9.1 .Anexo I: Descripción detallada de los casos de uso

---

OBSERVACIONES
...

*Tabla 6: Descripción caso de uso: Modificar Noticia*

<b>NOMBRE DEL CASO DE USO</b>	
	Eliminar noticia
<b>RESUMEN</b>	
	Este caso de uso permite eliminar una noticia del sistema.
<b>DEPENDENCIAS</b>	
	Este caso de uso incluye el caso de uso <i>Validar Usuario</i>
<b>ACTORES</b>	
	Gestor (sistema externo)
<b>PRECONDICIONES</b>	
	-
<b>POSTCONDICIONES</b>	
	La noticia no existirá en el sistema.
<b>CURSO NORMAL</b>	
1.	El gestor indica el identificador de la noticia a eliminar. Además de la información indicada el usuario indicará su lógín y firmará el mensaje. La firma del mensaje consistirá en el resultado de cifrar mediante <i>SHA</i> la cadena:
	<code>login + password + identificador</code>
2.	El sistema comprueba si el usuario es correcto. Se ejecuta el caso de uso <i>ValidarUsuario</i> .
3.	El usuario es correcto el sistema comprueba que los parámetros son correctos.
4.	Los parámetros son correctos y elimina la noticia. El sistema indica que la noticia ha sido eliminada.
<b>CURSOS ALTERNATIVOS</b>	
3 .	El usuario no es correcto. El sistema informa del error. Finaliza el caso de uso.
4.	Los parámetros no son correctos. El sistema informa del error. Finaliza el caso de uso.
<b>OBSERVACIONES</b>	
	-

Tabla 7: Descripción caso de uso: *Eliminar Noticia*

## 9.1 .Anexo I: Descripción detallada de los casos de uso

<b>NOMBRE DEL CASO DE USO</b>
Obtener noticias
<b>RESUMEN</b>
Este caso de uso permite obtener las noticias pertenecientes a una de las categorías gestionadas por el gestor.
<b>DEPENDENCIAS</b>
Este caso de uso incluye el caso de uso <i>Validar Usuario</i>
<b>ACTORES</b>
Gestor (sistema externo)
<b>PRECONDICIONES</b>
-
<b>POSTCONDICIONES</b>
-
<b>CURSO NORMAL</b>
<ol style="list-style-type: none"><li>1. El gestor indica el nombre de la categoría de las noticias a obtener. Además de la información indicada el usuario indicará su lógín y firmará el mensaje. La firma del mensaje consistirá en el resultado de cifrar mediante <i>SHA</i> la cadena:  login + password + categoría.</li><li>2. El sistema comprueba si el usuario es correcto. Se ejecuta el caso de uso <i>ValidarUsuario</i>.</li><li>3. El usuario es correcto el sistema comprueba que los parámetros son correctos.</li><li>4. Los parámetros son correctos y devuelve las noticias pertenecientes a la categoría.</li></ol>
<b>CURSOS ALTERNATIVOS</b>
<ol style="list-style-type: none"><li>3 . El usuario no es correcto. El sistema informa del error. Finaliza el caso de uso.</li><li>4. Los parámetros no son correctos. El sistema informa del error. Finaliza el caso de uso.</li></ol>
<b>OBSERVACIONES</b>

Tabla 8: Descripción caso de uso: *Obtener Noticias*

<b>NOMBRE DEL CASO DE USO</b>	
	Obtener categorías
<b>RESUMEN</b>	
	Este caso de uso permite obtener las categorías que gestiona el gestor.
<b>DEPENDENCIAS</b>	
	Este caso de uso incluye el caso de uso <i>Validar Usuario</i> .
<b>ACTORES</b>	
	Gestor (sistema externo)
<b>PRECONDICIONES</b>	
	-
<b>POSTCONDICIONES</b>	
	-
<b>CURSO NORMAL</b>	
	<ol style="list-style-type: none"> <li>1. El gestor indica el lógin y firmará el mensaje. La firma del mensaje consistirá en el resultado de cifrar mediante <i>SHA</i> la cadena:  login + password</li> <li>2. El sistema comprueba si el usuario es correcto. Se ejecuta el caso de uso <i>ValidarUsuario</i>.</li> <li>3. El usuario es correcto el sistema comprueba que los parámetros son correctos.</li> <li>4. Los parámetros son correctos y el sistema devuelve la lista de categorías.</li> </ol>
<b>CURSOS ALTERNATIVOS</b>	
	<ol style="list-style-type: none"> <li>3 . El usuario no es correcto. El sistema informa del error. Finaliza el caso de uso.</li> <li>4. Los parámetros no son correctos. El sistema informa del error. Finaliza el caso de uso.</li> </ol>
<b>OBSERVACIONES</b>	

Tabla 9: Descripción caso de uso: Obtener Categorías

## 9.1 .Anexo I: Descripción detallada de los casos de uso

<b>NOMBRE DEL CASO DE USO</b>
Dar de alta dispositivo
<b>RESUMEN</b>
Este caso de uso permite dar de alta un nuevo dispositivo en el sistema.
<b>DEPENDENCIAS</b>
Este caso de uso incluye el caso de uso <i>Validar Usuario</i> .
<b>ACTORES</b>
Gestor (sistema externo)
<b>PRECONDICIONES</b>
-
<b>POSTCONDICIONES</b>
Se ha registrado un dispositivo en el sistema.
<b>CURSO NORMAL</b>
<p>1 . El gestor envía al sistema la información del dispositivo a registrar en el sistema (mac, pin, categorías). Además de la información indicada el usuario indicará su lógin y firmará el mensaje. La firma del mensaje consistirá en el resultado de cifrar mediante <i>SHA</i> la cadena:</p> <p>login + password + mac + [nombre + descripción + texto + id_Gestor]*</p> <p>2 . El sistema comprueba si el usuario es correcto. Se ejecuta el caso de uso <i>ValidarUsuario</i>.</p> <p>3 . El usuario es correcto el sistema comprueba que los parámetros son correctos y que las categorías son gestionadas por el gestor que efectúa la operación.</p> <p>4 . Los parámetros son correctos y registra el dispositivo. El sistema devuelve el identificador del dispositivo creado en el sistema.</p>
<b>CURSOS ALTERNATIVOS</b>
<p>3 . El usuario no es correcto. El sistema informa del error. Finaliza el caso de uso.</p> <p>4. Los parámetros no son correctos. El sistema informa del error. Finaliza el caso de uso.</p>

---

9 .Anexo I: Descripción detallada de los casos de uso

---

OBSERVACIONES
-

*Tabla 10: Descripción caso de uso: Dar de alta dispositivo móvil*

## 9.1 .Anexo I: Descripción detallada de los casos de uso

<b>NOMBRE DEL CASO DE USO</b>
Dar de baja dispositivo móvil
<b>RESUMEN</b>
Este caso de uso permite dar de baja un dispositivo existente en el sistema.
<b>DEPENDENCIAS</b>
Este caso de uso incluye el caso de uso <i>Validar Usuario</i>
<b>ACTORES</b>
Gestor (sistema externo)
<b>PRECONDICIONES</b>
-
<b>POSTCONDICIONES</b>
El dispositivo no existirá en el sistema.
<b>CURSO NORMAL</b>
<ol style="list-style-type: none"><li>1. El gestor indica la MAC del dispositivo a eliminar. Además de la información indicada el usuario indicará su lógin y firmará el mensaje. La firma del mensaje consistirá en el resultado de cifrar mediante <i>SHA</i> la cadena:  login + password + MAC</li><li>2. El sistema comprueba si el usuario es correcto. Se ejecuta el caso de uso <i>ValidarUsuario</i>.</li><li>3. El usuario es correcto el sistema comprueba que los parámetros son correctos y si el dispositivo no tiene ninguna categoría asociada.</li><li>4. Los parámetros son correctos y da de baja el dispositivo. El sistema indica que el dispositivo ha sido dado de baja.</li></ol>
<b>CURSOS ALTERNATIVOS</b>
<ol style="list-style-type: none"><li>3 . El usuario no es correcto. El sistema informa del error. Finaliza el caso de uso.</li><li>4. Los parámetros no son correctos. El sistema informa del error. Finaliza el caso de uso.</li></ol>

---

9 .Anexo I: Descripción detallada de los casos de uso

---

OBSERVACIONES
...

*Tabla II: Descripción caso de uso: Dar de baja dispositivo móvil*

## 9.1 .Anexo I: Descripción detallada de los casos de uso

<b>NOMBRE DEL CASO DE USO</b>
Modificar dispositivo móvil
<b>RESUMEN</b>
Este caso de uso permite modificar un dispositivo del sistema.
<b>DEPENDENCIAS</b>
Este caso de uso incluye el caso de uso <i>Validar Usuario</i>
<b>ACTORES</b>
Gestor (sistema externo)
<b>PRECONDICIONES</b>
-
<b>POSTCONDICIONES</b>
El dispositivo quedará modificado en el sistema.
<b>CURSO NORMAL</b>
<ol style="list-style-type: none"><li>1. El gestor indica la MAC del dispositivo a modificar, la nueva MAC, y el nuevo pin. Además de la información indicada el usuario indicará su lógin y firmará el mensaje. La firma del mensaje consistirá en el resultado de cifrar mediante <i>SHA</i> la cadena:          login + password + MAC + nueva MAC + nuevo pin.</li><li>2. El sistema comprueba si el usuario es correcto. Se ejecuta el caso de uso <i>ValidarUsuario</i>.</li><li>3. El usuario es correcto el sistema comprueba que los parámetros son correctos.</li><li>4. Los parámetros son correctos y modifica el dispositivo. El sistema indica que el dispositivo ha sido modificado.</li></ol>
<b>CURSOS ALTERNATIVOS</b>
<ol style="list-style-type: none"><li>3 . El usuario no es correcto. El sistema informa del error. Finaliza el caso de uso.</li><li>4. Los parámetros no son correctos. El sistema informa del error. Finaliza el caso de uso.</li></ol>
<b>OBSERVACIONES</b>
-

Tabla 12: Descripción caso de uso: *Modificar dispositivo móvil*

<b>NOMBRE DEL CASO DE USO</b>
Modificar dispositivo móvil
<b>RESUMEN</b>
Este caso de uso permite modificar un dispositivo del sistema.
<b>DEPENDENCIAS</b>
Este caso de uso incluye el caso de uso <i>Validar Usuario</i>
<b>ACTORES</b>
Gestor (sistema externo)
<b>PRECONDICIONES</b>
-
<b>POSTCONDICIONES</b>
El dispositivo quedará modificado en el sistema.
<b>CURSO NORMAL</b>
<ol style="list-style-type: none"> <li>1. El gestor indica la MAC del dispositivo a modificar, la nueva MAC, y el nuevo pin. Además de la información indicada el usuario indicará su lógin y firmará el mensaje. La firma del mensaje consistirá en el resultado de cifrar mediante <i>SHA</i> la cadena:</li> </ol> <p style="padding-left: 40px;">login + password + MAC + nueva MAC + nuevo pin.</p> <ol style="list-style-type: none"> <li>2. El sistema comprueba si el usuario es correcto. Se ejecuta el caso de uso <i>ValidarUsuario</i>.</li> <li>3. El usuario es correcto el sistema comprueba que los parámetros son correctos.</li> <li>4. Los parámetros son correctos y modifica el dispositivo. El sistema indica que el dispositivo ha sido modificado.</li> </ol>
<b>CURSOS ALTERNATIVOS</b>
<ol style="list-style-type: none"> <li>3 . El usuario no es correcto. El sistema informa del error. Finaliza el caso de uso.</li> <li>4. Los parámetros no son correctos. El sistema informa del error. Finaliza el caso de uso.</li> </ol>
<b>OBSERVACIONES</b>
-

Tabla 13: Descripción caso de uso: Asociar categoría a dispositivo móvil

## 9.1 .Anexo I: Descripción detallada de los casos de uso

<b>NOMBRE DEL CASO DE USO</b>
Desasociar categoría a dispositivo
<b>RESUMEN</b>
Este caso de uso permite desasociar un dispositivo de una categoría.
<b>DEPENDENCIAS</b>
Este caso de uso incluye el caso de uso <i>Validar Usuario</i> .
<b>ACTORES</b>
Gestor (sistema externo)
<b>PRECONDICIONES</b>
-
<b>POSTCONDICIONES</b>
El dispositivo no está asociado a la categoría.
<b>CURSO NORMAL</b>
<ol style="list-style-type: none"><li>1. El gestor indica la MAC del dispositivo y la categoría a desasociar. Además indica el lógín y firmará el mensaje. La firma del mensaje consistirá en el resultado de cifrar mediante <i>SHA</i> la cadena:  login + password + MAC + categoría</li><li>2. El sistema comprueba si el usuario es correcto. Se ejecuta el caso de uso <i>ValidarUsuario</i>.</li><li>3. El usuario es correcto el sistema comprueba que los parámetros son correctos y la categoría es gestionada por el gestor.</li><li>4. Los parámetros son correctos y el sistema desasocia la categoría. El sistema indica que el dispositivo ha sido desasociado de la categoría.</li></ol>
<b>CURSOS ALTERNATIVOS</b>
<ol style="list-style-type: none"><li>3 . El usuario no es correcto. El sistema informa del error. Finaliza el caso de uso.</li><li>4. Los parámetros no son correctos. El sistema informa del error. Finaliza el caso de uso.</li></ol>
<b>OBSERVACIONES</b>

Tabla 14: Descripción caso de uso: Desasociar categoría de dispositivo móvil

<b>NOMBRE DEL CASO DE USO</b>	
	Obtener dispositivos
<b>RESUMEN</b>	
	Este caso de uso permite obtener los dispositivos existentes en el sistema.
<b>DEPENDENCIAS</b>	
	Este caso de uso incluye el caso de uso <i>Validar Usuario</i> .
<b>ACTORES</b>	
	Gestor (sistema externo)
<b>PRECONDICIONES</b>	
	-
<b>POSTCONDICIONES</b>	
	-
<b>CURSO NORMAL</b>	
	<ol style="list-style-type: none"> <li>1. El gestor indica la si desea obtener los dispositivos que desea obtener (asociados a sus categorías, todos los del sistema ó los asociados a una determinada categoría) . Además indica el lógín y firmará el mensaje. La firma del mensaje consistirá en el resultado de cifrar mediante <i>SHA</i> la cadena:</li> </ol> <p style="padding-left: 40px;">login + password</p> <ol style="list-style-type: none"> <li>2. El sistema comprueba si el usuario es correcto. Se ejecuta el caso de uso <i>ValidarUsuario</i>.</li> <li>3. El usuario es correcto el sistema comprueba que los parámetros son correctos.</li> <li>4. Los parámetros son correctos y el sistema devuelve la lista de dispositivos.</li> </ol>
<b>CURSOS ALTERNATIVOS</b>	
	<ol style="list-style-type: none"> <li>3 . El usuario no es correcto. El sistema informa del error. Finaliza el caso de uso.</li> <li>4. Los parámetros no son correctos. El sistema informa del error. Finaliza el caso de uso.</li> </ol>
<b>OBSERVACIONES</b>	

Tabla 15: Descripción caso de uso: Obtener dispositivos móviles

9.1 .Anexo I: Descripción detallada de los casos de uso

---

<b>NOMBRE DEL CASO DE USO</b>
Mostrar noticias pantallas
<b>RESUMEN</b>
Este caso de uso mostrar las noticias por las pantallas.
<b>DEPENDENCIAS</b>
-
<b>ACTORES</b>
Temporizador
<b>PRECONDICIONES</b>
-
<b>POSTCONDICIONES</b>
-
<b>CURSO NORMAL</b>
<ol style="list-style-type: none"><li>1. El temporizador lanza el caso de uso.</li><li>2. El sistema consulta en la base de datos las noticias existentes y actualiza el contenido de las pantallas (mostrando ó eliminando noticias de las mismas en función de las fechas) en caso de que haya cambiado dicho contenido.</li></ol>
<b>CURSOS ALTERNATIVOS</b>
<ol style="list-style-type: none"><li>2 . No se puede establecer conexión con la base de datos. Finaliza el caso de uso.</li></ol>
<b>OBSERVACIONES</b>
-

*Tabla 16: Descripción caso de uso: Mostrar noticias pantallas*

### **9.1.2 Subsistema Bluetooth**

<b>NOMBRE DEL CASO DE USO</b>
Inicializar Pila
<b>RESUMEN</b>
Este caso de uso realiza una llamada a la librería Bluetooth con una serie de parámetros para inicializar la pila Bluetooth y que se puedan ejecutar las operaciones sobre ésta.
<b>DEPENDENCIAS</b>
Este caso de uso extiende a los casos de uso Escanear dispositivos, Escanear servicios, Realizar Pairing y Enviar noticias.
<b>ACTORES</b>
Temporizador
<b>PRECONDICIONES</b>
La pila no ha sido inicializada previamente
<b>POSTCONDICIONES</b>
La pila está inicializada o en estado de error si ha habido algún error.
<b>CURSO NORMAL</b>
<ol style="list-style-type: none"><li>1 . El temporizador realiza una llamada a la librería Bluetooth con la configuración que desea para la pila.</li><li>2 . La librería inicializa la pila correctamente y termina el caso de uso.</li></ol>
<b>CURSOS ALTERNATIVOS</b>
2b. Ocurre algún error en la inicialización, la pila queda en estado de error y termina el caso de uso.
<b>OBSERVACIONES</b>
-

*Tabla 17: Descripción caso de uso: Inicializar Pila*

## 9.1 .Anexo I: Descripción detallada de los casos de uso

<b>NOMBRE DEL CASO DE USO</b>
Escanear Dispositivos
<b>RESUMEN</b>
Este caso de uso permite el escaneo de los dispositivos Bluetooth a nuestro alcance.
<b>DEPENDENCIAS</b>
Este caso de uso es extendido por Inicializar Pila
<b>ACTORES</b>
Temporizador
<b>PRECONDICIONES</b>
La pila Bluetooth ha sido inicializada previamente mediante el caso de uso Inicializar Pila.
<b>POSTCONDICIONES</b>
Se poseen todas las direcciones MAC de los dispositivos en estado visible y que estén a nuestro alcance
<b>CURSO NORMAL</b>
<ol style="list-style-type: none"><li>1 . El temporizador realiza una búsqueda de dispositivos mediante la librería que implementa la Pila Bluetooth.</li><li>2 . Se encuentra un dispositivo dado por su MAC y se añade a una lista y posteriormente a la base de datos si no lo estaba previamente.</li><li>3 . Finalizado el barrido finaliza el caso de uso.</li></ol>
<b>CURSOS ALTERNATIVOS</b>
2b. No se encuentra ningún dispositivo durante el tiempo de barrido (timeout).
<b>OBSERVACIONES</b>
-

Tabla 18: Descripción caso de uso: Escanear Dispositivos

<b>NOMBRE DEL CASO DE USO</b>	
	Escanear Servicios
<b>RESUMEN</b>	
	Este caso de uso permite buscar la dirección (URL) de conexión para el servicio de envío de ficheros de un dispositivo dado.
<b>DEPENDENCIAS</b>	
	Este caso de uso es extendido por Inicializar Pila e incluye al caso de uso.
<b>ACTORES</b>	
	Temporizador
<b>PRECONDICIONES</b>	
	<p>La pila Bluetooth ha sido inicializada previamente mediante el caso de uso Inicializar Pila.</p> <p>Se conoce la dirección MAC del dispositivo al que se le realizará el escaneo.</p>
<b>POSTCONDICIONES</b>	
	Se obtiene la URL de conexión del dispositivo o error en caso de que no exista o no se haya encontrado.
<b>CURSO NORMAL</b>	
	<ol style="list-style-type: none"> <li>1 . Se realiza una consulta a la base de datos para comprobar si ya tenemos almacenado el URL de conexión para el dispositivo.</li> <li>2 . No conocemos el URL del dispositivo, el temporizador realiza una búsqueda de servicios para dicho dispositivo Bluetooth.</li> <li>3 . Se encuentra el URL de conexión para el servicio solicitado</li> <li>4 . Se añade dicho URL a la base de datos para las próximas consultas, se devuelve dicho URL y finaliza el caso de uso.</li> </ol>
<b>CURSOS ALTERNATIVOS</b>	
	<p>2b. El URL del servicio para el dispositivo ya lo conocemos, se devuelve dicho URL y finaliza el caso de uso.</p> <p>3b. Ocurre un error en la búsqueda o el dispositivo no posee dicho servicio, se devuelve error y finaliza el caso de uso</p>
<b>OBSERVACIONES</b>	
	-

Tabla 19: Descripción caso de uso: Escanear Servicios

## 9.1 .Anexo I: Descripción detallada de los casos de uso

<b>NOMBRE DEL CASO DE USO</b>
Realizar Pairing
<b>RESUMEN</b>
Este caso de uso permite la el emparejamiento validado entre dos dispositivos bluetooth.
<b>DEPENDENCIAS</b>
Este caso de uso es extendido por Inicializar Pila y es incluido por Enviar noticias.
<b>ACTORES</b>
Dispositivo Registrado, Dispositivo no Registrado, Temporizador
<b>PRECONDICIONES</b>
La pila Bluetooth ha sido inicializada previamente mediante el caso de uso Inicializar Pila. Se conoce la URL de conexión del dispositivo y su PIN en caso de estar registrado.
<b>POSTCONDICIONES</b>
Los dos dispositivos quedan emparejados si el proceso tuvo éxito.
<b>CURSO NORMAL</b>
<ol style="list-style-type: none"><li>1. El temporizador envía petición de conexión a la URL de conexión del servicio.</li><li>2. Aparece una petición de conexión en el dispositivo destino, solicitándole el código pin .</li><li>3. El usuario del dispositivo (registrado o no registrado) introduce interactivamente su PIN en caso de estar registrado (está almacenado en nuestra BD) o un PIN genérico (0000) si no lo está.</li><li>4. El pin introducido es correcto, los dos dispositivos quedan emparejados y finaliza el caso de uso.</li></ol>
<b>CURSOS ALTERNATIVOS</b>
2b. La petición no llega al dispositivo destino, se cumple un timeout de espera o ocurre un error en dicho proceso. Se finaliza el caso de uso devolviendo el error ocurrido.
3b. El usuario cancela la petición de PIN o no lo introduce durante un tiempo determinado, se finaliza el caso de uso devolviendo el error ocurrido.
4b. El pin es incorrecto, el emparejamiento falla y finaliza el caso de uso.

OBSERVACIONES
-

Tabla 20: Descripción caso de uso: Realizar Pairing

NOMBRE DEL CASO DE USO
Generar noticias Dispositivo
RESUMEN
Este caso los datos con las noticias que tienen que ser enviadas a un dispositivo.
DEPENDENCIAS
Este caso de uso es incluido por Enviar Noticias.
ACTORES
-
PRECONDICIONES
Se conoce la dirección MAC del dispositivo.
POSTCONDICIONES
Se obtienen el máximo número de noticias que pueden ser enviadas al dispositivo de forma ordenada y formateadas adecuadamente para su dispositivo.
CURSO NORMAL
<ol style="list-style-type: none"> <li>1. El dispositivo está registrado, se obtiene una lista de la BD con las siguientes noticias a enviar para el dispositivo.</li> <li>2. Se genera una cadena o fichero con toda la información obtenida y formateada para el dispositivo.</li> <li>3. Se devuelve dicho fichero y finaliza el caso de uso.</li> </ol>
CURSOS ALTERNATIVOS
2b. El dispositivo no está registrado, se obtiene una lista de la Bd con las noticias correspondientes a la categoría pantalla.
OBSERVACIONES
-

Tabla 21: Descripción caso de uso: Generar Noticias Dispositivo

## 9.1 .Anexo I: Descripción detallada de los casos de uso

<b>NOMBRE DEL CASO DE USO</b>
Enviar Noticias
<b>RESUMEN</b>
Este caso de uso se encarga de enviar un fichero de noticias al dispositivo indicado.
<b>DEPENDENCIAS</b>
Este caso de uso incluye a los casos de uso Realizar Pairing y Generar noticias Dispositivos. Es extendido por el caso de uso Inicializar Pila.
<b>ACTORES</b>
Temporizador
<b>PRECONDICIONES</b>
Se conoce la dirección MAC del dispositivo Se ha realizado el emparejamiento Se dispone de la lista de noticias a enviar formateadas para el dispositivo.
<b>POSTCONDICIONES</b>
El dispositivo recibe las noticias que le correspondan. En la base de datos queda reflejado si se ha realizado el envío correctamente de todas las noticias implicadas.
<b>CURSO NORMAL</b>
<ol style="list-style-type: none"><li>1. El dispositivo está registrado, se realiza el caso de uso Generar noticias Dispositivo para obtener el listado de noticias formateadas.</li><li>2. Se establece la conexión o emparejamiento con el dispositivo mediante el caso de uso Realizar Pairing.</li><li>3. Se envía petición de envío del fichero al dispositivo.</li><li>4. Aparece un mensaje de confirmación en el dispositivo de destino para confirmar la recepción.</li><li>5. El usuario acepta el fichero.</li><li>6. Se transfiere el fichero</li><li>7. Se almacena en la base de datos las noticias que han sido enviadas correctamente al dispositivo y finaliza el caso de uso.</li></ol>
<b>CURSOS ALTERNATIVOS</b>
2b. El pairing falla y por tanto no se pueden enviar noticias, finaliza el caso de uso. 4b. La petición de envío no ha llegado o se ha cumplido el timeout para confirmar la recepción del fichero, finaliza el caso de uso. 5b. El usuario cancela el fichero, finaliza el caso de uso.

---

9 .Anexo I: Descripción detallada de los casos de uso

---

OBSERVACIONES
-

*Tabla 22: Descripción caso de uso: Enviar Noticias*

## 9.2 Anexo II: Diccionario de datos análisis y diseño.

En este anexo se muestran todos y cada uno de los atributos de las clases mencionadas en el capítulo 5.3 , Análisis y Diseño del sistema. Los métodos correspondientes a las clases pueden consultarse en el código implementado o, en algunos casos, en la documentación JAVADOC en el cdrom adjunto al presente documento. Las clases implementadas finalmente estarán constituidas por los atributos indicados en el diccionario más los atributos resultantes de las relaciones existentes entre clases (referencias a uno o varios objetos de otras clases). Las clases y los atributos se muestran en orden alfabético ascendente.

Clase (pkg)	Atributo	Tipo	Descripción
BD_BLUFEEDME (Configuración)	host	String	Dirección del host donde se encuentra la Base de Datos
	database	String	Base de Datos a la que se realizará la conexión
	user	String	Usuario para realizar la conexión con la Base de Datos.
	password	String	Password para realizar la conexión con la Base de Datos.
BLUFEEDMEBTABOUT BOX (vista)			
BLUFEEDMEBTVIEW (vista)	aboutBox	JDialog	Dialogo con la información referente a la aplicación.
	handlerListaLog	ListStreamHandler	Manejador para añadir filas al log
	jDialogconfiguracion	JDialog	Dialogo inicial para seleccionar los archivos de configuración necesarios para el funcionamiento de la aplicación
	LOGGER	Logger	log general de la aplicación
	rendererFilasDispositivos	RendererListaDispositivos	Renderer de las filas de la lista de dispositivos

9 .Anexo II: Diccionario de datos análisis y diseño.

Clase (pkg)	Atributo	Tipo	Descripción
	rendererFilas Log	RendererLista Log	Renderer de las filas de la lista de log
BDBLUFEEDMEMYSQL (basededatos.implementación)	con	Connection	Objeto para establecer la conexión con una Base de Datos MySQL.
	password	String	Password del usuario a utilizar para establecer la conexión con la Base de Datos.
	UrlBD	String	Dirección url de la base de datos para establecer la conexión.
	user	String	Usuario a usar para establecer la conexión con la Base de Datos.
BLUFEEDMEBT (vista)			
CATEGORIA (Modelo)	descripcion	String	Breve descripción de la categoría: nombre completo, información de contacto, etc.
	id	long	Identificador único de categoría.
	nombre	String	Nombre de la categoría que se aparecerá en la publicación de las noticias asociadas a la categoría.
DISPOSITIVO (Modelo)	id	long	Identificador único de dispositivo.
	mac	String	Dirección ethernet del dispositivo. Formato Hexadecimal sin separadores: XXXXXXXXXXXX.
	pin	String	Código usado para llevar a cabo el pairing en la comunicación iniciada entre el dispositivo móvil y el sistema.

9.2 .Anexo II: Diccionario de datos análisis y diseño.

Clase (pkg)	Atributo	Tipo	Descripción
	url	String	Dirección url del servicio disponible en el dispositivo móvil para realizar el envío de los archivos que contienen las noticias a enviar.
DISPOSITIVOBT (BlufeedmeBT)	baseDatos	BdBlufeedme MySQL	Base de datos (una conexión por hebra o dispositivo de emisión)
	buscador	DiscoveryAgent	Buscador de nuestra hebra
	CATEGORIA_PANTALLA	Long	Define la categoría de noticias que se mostraran en pantalla y por tanto se enviaran a todos los dispositivos
	fichero_bd_prop	File	Fichero con las propiedades de conexión a la Base de datos
	LOGGER	Logger	Log general de la aplicación
	mapDispositivosURL	ConcurrentHashMap	Tabla hash concurrente para almacenar los dispositivos y su URL de conexión
	OBEX_FILE_TRANSFER	UUID	Define el protocolo obex file transfer
	OBEX_PUSH	UUID	Define el protocolo obex file transfer
	PIN_DEFAULT	String	Define el pin por defecto de conexión para dispositivos que no hayan sido registrados
	semaforoMaxBusqServ	Semaphore	Controla el número máximo de hebras de búsqueda simultáneas
	semaforoMaxConexiones	Semaphore	Controla el número máximo de hebras de envío

9 .Anexo II: Diccionario de datos análisis y diseño.

Clase (pkg)	Atributo	Tipo	Descripción
	semaforoEsca neo	Semaphore	Controla y bloquea la ejecución del escaneo.
	vectdispositiv os	HashMap	MAC, PIN
DISPOSITIVOVISTA (vista)	categorias	ArrayList	Lista de categorías pertenecientes al modelo que tiene asociado el dispositivo
	mac	String	Dirección ethernet del dispositivo
	nombre	String	Nombre del dispositivo
	noticias	ArrayList	Lista de noticias del dispositivo
	registrado	boolean	Indica si el dispositivo esta registrado en el sistema o no.
	pin	String	Código usado para establecer la conexión mediante Bluetooth.
	URL_Servicio	String	URL del servicio que se usa para intercambiar la información mediante la conexión Bluetooth.
ENVIARADISPOSITIVO (BlufeedmeBT)	baseDatos	BdBlufeedme MySql	Base de Datos a la que se realizará la conexión
	cabeceraDesc onectar	HeaderSet	Cabecera del mensaje para la desconexión entre dispositivos
	cabeceraOper acion	HeaderSet	Cabecera del mensaje para indicar la operación a realizar
	clientSession	ClientSession	Sesión creada por el cliente para conexión
	deviceId	String	Identificador del dispositivo local
	dispositivo	RemoteDevice	Dispositivo de destino para el envío

9.2 .Anexo II: Diccionario de datos análisis y diseño.

Clase (pkg)	Atributo	Tipo	Descripción
	fichero_bd_prop	File	Fichero con las propiedades de conexión con la BD
	fichero_html	File	Fichero con la plantilla que se utilizará en el archivo a enviar
	LOGGER	Logger	Variable para el log
	MAC	String	Dirección física del dispositivo
	maxConexiones	Semaphore	Controla el número máximo de hebras de envío
	MAX_NOTICIAS	int	Número máximo de noticias a realizar en un solo envío
	MAX_TAM_NOTICIAS	int	Tamaño máximo del fichero de noticias a enviar
	PIN	String	Pin del dispositivo de destino
	putOperation	Operation	Operación de envío de fichero
	streamSalida	OutputStream	Buffer de datos de salida
EVENTODISPOSITIVO (eventos)	URL_Servicio	String	URL de conexión con el dispositivo de destino
	dispositivo	Dispositivo	
	DISPOSITIVO_ENCONTRADO	int	Constante que define el tipo de evento en caso de encontrar un dispositivo
	DISPOSITIVO_REGISTRADO_ENCONTRADO	int	Constante que define el tipo de evento en caso de encontrar un dispositivo registrado
	nombreDispositivo	String	Nombre del dispositivo
	noticias	ArrayList	Lista de noticias del

9 .Anexo II: Diccionario de datos análisis y diseño.

Clase (pkg)	Atributo	Tipo	Descripción
			dispositivo
	NOTICIAS_A_E NVIAR	int	Número de noticias a enviar al dispositivo
	NOTICIAS_EN VIADAS	int	Número de noticias enviadas al dispositivo
	NO_VALIDO	int	Evento no válido
	NUEVA_BUSQ UEDA	int	Constante que define el tipo de evento en caso de nueva búsqueda
	SERVICIO_EN CONTRADO	int	Constante que define el tipo de evento en caso de encontrar un servicio
	tipoEvento	int	Almacena el tipo de evento
GESTOR (Modelo)			
GESTORNOTICIAS (Lógica)	bd	BD_BLUFEED ME	
INVALIDPARAMEXCEP TION (service.exception)	code	int	Código de excepción.
INVALIDUSEREXCEPTI ON (service.exception)	code	int	Código de excepción.
LISTENERDISPOSITIV O (eventos)			
LISTSTREAMHANDLER (vista)	listaLog	DefaultListMo del	Modelo de datos del JList
	n_lineasMax	int	Número de líneas máximas a mostrar en el JList
MIFORMATTER (vista)	formatoFecha	SimpleDateFor mat	Formato para la impresión de una fecha
MISHA (service.util)			
NEWSERVICE (service)	bd	BdBlufeedme MySQL	Objeto para gestionar las operaciones con la Base de Datos.
	CATEGORÍA_P ANTALLA	final long	Identificador de la categoría usada para

9.2 .Anexo II: Diccionario de datos análisis y diseño.

Clase (pkg)	Atributo	Tipo	Descripción
			mostrar noticias por las pantallas
	DATEERROR	final int	Código de error para indicar error con las fechas.
	CATEGORYER ROR	final int	Código error para indicar un error en la categoría indicada.
	NEWSERROR	final int	Código para indicar un error relacionado con la noticia indicada en la operación.
	DISPERROR	final int	Código para indicar un error relacionado con el dispositivo indicada en la operación.
	LOGINERROR	final int	Código para indicar un error al realizar la autentificación del usuario.
NOTICIA (Modelo)	autor	String	Autor que aparecerá en la publicación de la noticia.
	fecha	Calendar	Fecha en la que la noticia se crea en el sistema.
	fechaCaducidad	Calendar	Fecha a partir de la cual la noticia dejará de ser publicada por los medios de publicación de noticias.
	fechaPubli	Calendar	Fecha a partir de la cual la noticia comenzará a ser publicada en el sistema.
	id	long	Identificador único de noticia
	subtitulo	String	Subtitulo que aparecerá para la noticia cuando sea publicada.

9 .Anexo II: Diccionario de datos análisis y diseño.

Clase (pkg)	Atributo	Tipo	Descripción
	titulo	long	Titulo de la noticia que aparecerá cuando sea publicada.
NOTICIA (Lógica)	autor	String	Autor que aparecerá en la publicación de la noticia.
	categoria	String	Nombre de la categoría a la que la noticia esta asociada.
	fechaPubli	Calendar	Fecha a partir de la cual la noticia comenzará a ser publicada en el sistema.
	subtitulo	String	Subtitulo que aparecerá para la noticia cuando sea publicada.
	titulo	long	Titulo de la noticia que aparecerá cuando sea publicada.
NOTICIAWEB (service)	autor	String	Autor que aparecerá en la publicación de la noticia.
	fecha	Calendar	Fecha en la que la noticia se crea en el sistema.
	fechaCaducidad	Calendar	Fecha a partir de la cual la noticia dejará de ser publicada por los medios de publicación de noticias.
	fechaPubli	Calendar	Fecha a partir de la cual la noticia comenzará a ser publicada en el sistema.
	subtitulo	String	Subtitulo que aparecerá para la noticia cuando sea publicada.
	titulo	long	Titulo de la noticia que aparecerá cuando sea publicada.

9.2 .Anexo II: Diccionario de datos análisis y diseño.

---

Clase (pkg)	Atributo	Tipo	Descripción
RENDERERLISTADISP OSITIVOS (vista)			
RENDERERLISTALOG (vista)			
USUARIO (modelo)	contrasenia	String	Contraseña que se usará para autenticar al usuario.
	id	long	Identificador único de usuario.
	nombre	String	Nombre que se usará para identificar y autenticar al usuario.

Tabla 23: Diccionario datos Análisis

### **9.3 Anexo III: Diccionario de datos Base de Datos.**

En este anexo se describe, en detalle, cada uno de los atributos de las entidades y las relaciones del diagrama Entidad-Relación obtenido para nuestra Base de Datos en el capítulo 5.3.3, Base de Datos.

Para ello realizamos dos tablas; una para mostrar las entidades y sus atributos y en la que describiremos las relaciones junto con las distintas entidades participantes de las relaciones y la cardinalidad con la que participan.

Los campos de los que constan las dos tablas son:

- ◆ Concepto: nombre de la entidad, relación o atributo que se va a describir.
- ◆ Descripción: breve descripción del concepto correspondiente.
- ◆ Tipo: en el caso de los atributos de las entidades y las relaciones, indicará el tipo de dato utilizado para su representación interna en la Base de Datos.
- ◆ Dominio: conjunto válido de valores que puede tomar el atributo al que hace referencia.
- ◆ PK: indica si el atributo actúa como clave primaria. Para una misma entidad o relación puede haber un conjunto de atributos que actúen como clave primaria.
- ◆ Nulo: indica si el atributo puede, o no, ser nulo.

El listado de entidades y atributos se realiza en orden alfabético ascendente.

Concepto	Descripción	Tipo	Dominio	PK	Nulo
Categoría	Entidad que representa a una de las categorías en las que se agrupan las noticias existentes en nuestro sistema				
descripción	Descripción detallada de la categoría; nombre completo, información de interés, contacto...	Cadena(80)		No	Si
id_categoria	Identificador único de la tabla.	BigInt(20)		Si	No
nombre	Nombre de la categoría que aparecerá en las	Cadena(80)		No	No

9.3 Anexo III: Diccionario de datos Base de Datos.

Concepto	Descripción	Tipo	Dominio	PK	Nulo
	publicaciones de las noticias.				
Dispositivo	Entidad que representa a uno de los dispositivos móviles registrados en nuestro sistema para recibir noticias vía Bluetooth				
id_dispositivo	Identificador único de la tabla.	BigInt(20)		Si	No
MAC	Dirección ethernet del dispositivo móvil en hexadecimal.	Cadena(12)	XXXXXXXXXXX XXX (sin separadores)	No	No
pin	Código que se usara para establecer la comunicación vía Bluetooth (Pairing) entre el dispositivo y nuestro sistema.	Cadena(7)		No	No
URL	Dirección del servicio para el envío de datos en las comunicaciones establecidas entre el dispositivo móvil y nuestro sistema.	Cadena(200)		No	Si
Gestor	Entidad que representa a uno de los usuarios gestores registrados en nuestro sistema.				
contraseña	Cadena de caracteres que constituirá la contraseña que permitirá autenticarse al usuario gestor en nuestro sistema	Cadena(20)		No	No
id_gestor	Identificador único de la tabla.	BigInt(20)		Si	No
nombre	Nombre que el usuario gestor usará para	Cadena(20)		No	No

9 .Anexo III: Diccionario de datos Base de Datos.

Concepto	Descripción	Tipo	Dominio	PK	Nulo
	autentificarse en el sistema.				
Noticia	Entidad que representa a una de las noticias creadas en nuestro sistema por los gestores.				
autor	Nombre del autor de la noticia.	cadena(80)		No	No
fecha	Fecha en la que la noticia fue creada en el sistema	DateTime		No	No
fecha_caducidad	Fecha en la que la noticia dejará de aparecer publicada por los diferentes medios.	DateTime		No	No
fecha_publicacion	Fecha a partir de la cual la noticia aparecerá publicada en los diferentes medios de publicación de noticias	DateTime		No	No
id_noticia	Identificador único de la tabla	BigInt(20)		Si	No
subtitulo	Subtitulo de la noticia.	cadena(100)		No	Si
texto	Cuerpo de la noticia.	text		No	No
titulo	Titulo de la noticia.	cadena(100)		No	No

Pasamos a definir las distintas relaciones recogidas en el Diagrama Entidad-Relación, así como las entidades que participan en ellas y la cardinalidad con la que lo hacen.

Concepto	Descripción	Entidad 1	Entidad 2	Cardinalidad
Asociado	Relación que indica las categorías a las que se encuentra asociado un dispositivo móvil	Dispositivo	Categoría	Muchos a muchos
Gestionar	Relación que indica que Gestor		Categoría	Uno a muchos

9.3 Anexo III: Diccionario de datos Base de Datos.

<b>Concepto</b>	<b>Descripción</b>	<b>Entidad 1</b>	<b>Entidad 2</b>	<b>Cardinalidad</b>
	gestor gestiona cada una de las categorías existentes en nuestro sistema.			
Pertenece	Relación que indica la categoría a la que pertenece cada una de las noticias creadas en el sistema.	Noticia	Categoría	Muchos a uno
Historial	Relación que indica las noticias que han sido enviadas a un dispositivo móvil en el tiempo de vida del sistema.	Noticia	Dispositivo	Muchos a muchos

## ***9.4 Anexo IV: Web Service.***

### **Introducción**

Un servicio web (en inglés, Web Service) es una pieza de software que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos. Las organizaciones OASIS y W3C son los comités responsables de la arquitectura y reglamentación de los servicios Web. Para mejorar la interoperabilidad entre distintas implementaciones de servicios Web se ha creado el organismo WS-I, encargado de desarrollar diversos perfiles para definir de manera más exhaustiva estos estándares.

### **Aplicaciones clásicas frente a aplicaciones orientadas a servicios SOA.**

Cuando queremos diseñar una aplicación empresarial podemos adoptar dos estrategias de desarrollo claramente diferenciadas: diseño orientado a objetos o un diseño orientado a servicios. Aunque estos paradigmas son en cierto sentido antagónicos, ambos son perfectamente válidos para construir aplicaciones, cada uno con sus ventajas e inconvenientes.

La programación orientada a objetos (POO), consiste en intentar modelar el sistema mediante objetos que representan cosas del mundo real mediante el encapsulamiento de unos datos y un comportamiento relacionados con el dominio del problema que queremos resolver. Estos objetos se conocen como objetos de dominio de la aplicación, que tendrán más o menos complejidad en función de las características del proyecto que estemos desarrollando. Este patrón de diseño es conocido como Domain-Driven Design (DDD).

Una alternativa a DDD es la Arquitectura Orientada a Servicios (SOA). Podríamos decir que la mayoría de las aplicaciones JavaEE que se construyen hoy en día tienen este tipo de arquitectura, que pone el foco en la definición y abstracción de funcionalidades en componentes más sencillos, con interfaces bien definidos, y promueven la separación (desacoplamiento) de las capas específicas de la aplicación.

En multitud de textos se realiza una relación directa de Web Services con SOA (Arquitectura Orientada a Servicios), ya que los Web Services son considerados los servicios bases para su funcionamiento, dejando claro que se podrían utilizar otros.

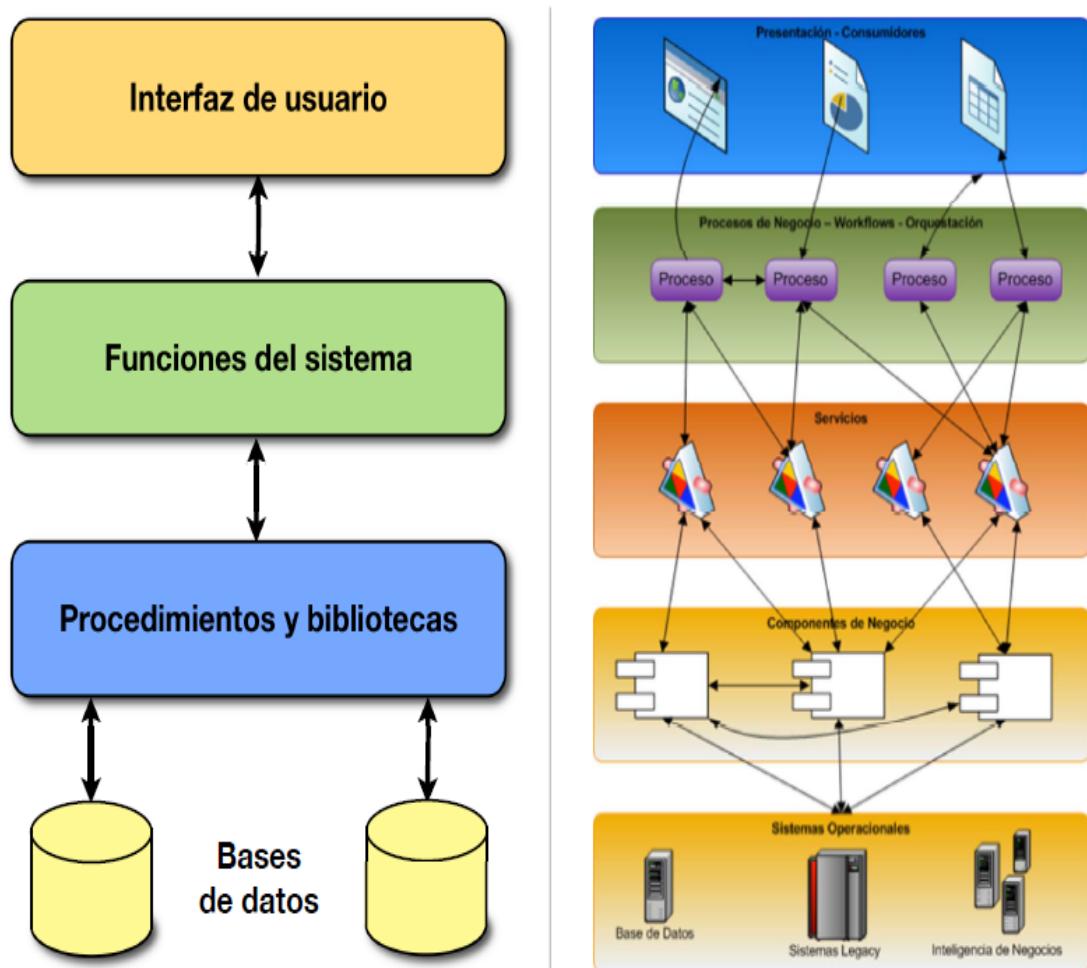


Ilustración 9.4.1: Aplicaciones clásicas VS Aplicaciones SOA

### Arquitectura SOA y modelo conceptual (JAVA)

La definición de la arquitectura SOA describe una serie de patrones o guías para crear servicios que sean independientes, relacionados, y alineados con el negocio. Dada la separación entre la descripción, la implementación y el “binding” del servicio, es posible tener gran flexibilidad en lo que se refiere a nuevas oportunidades de negocio – B2B.

Para poder obtener estas características, se deben de cumplir con una serie de lineamientos que nos van a permitir llegar a tener una arquitectura orientada a servicios dentro de la organización.

Una arquitectura SOA se puede ver de manera parcial y abstracta a través de una arquitectura n-layer de servicios que se alinean con los procesos del negocio. La siguiente figura nos muestra esta arquitectura n-layer.

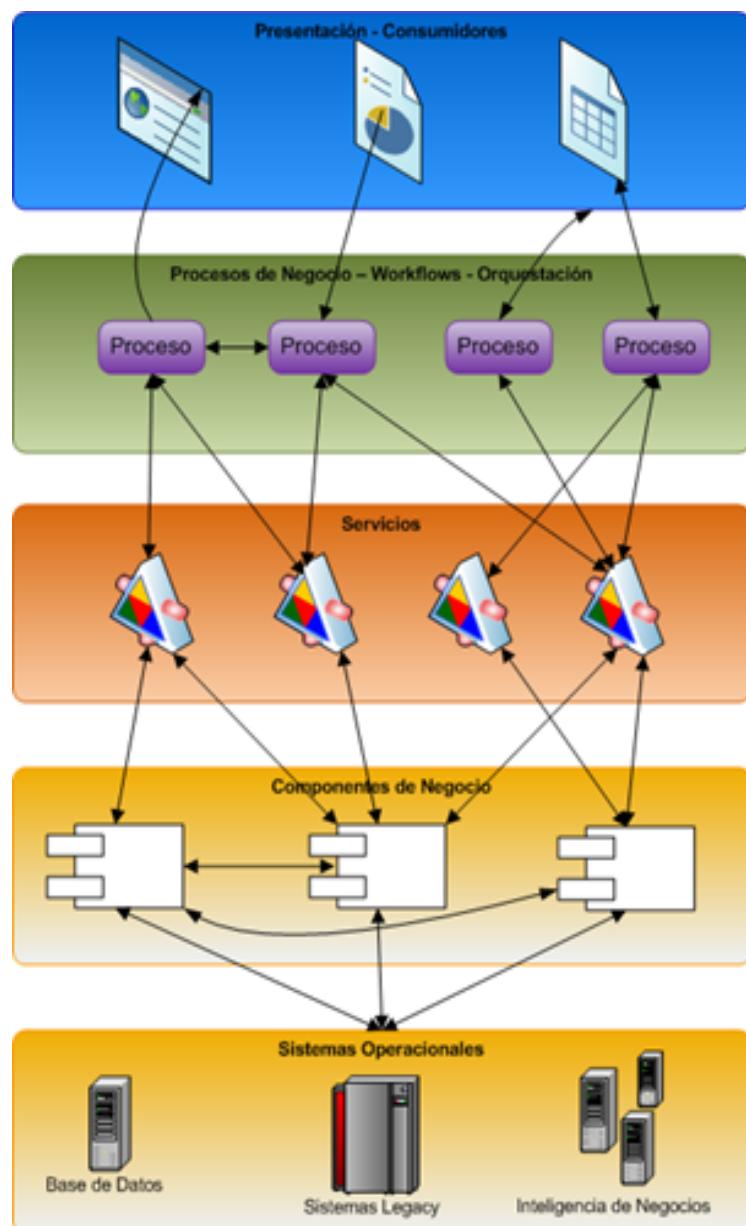


Ilustración 9.4.2: Arquitectura n-layer

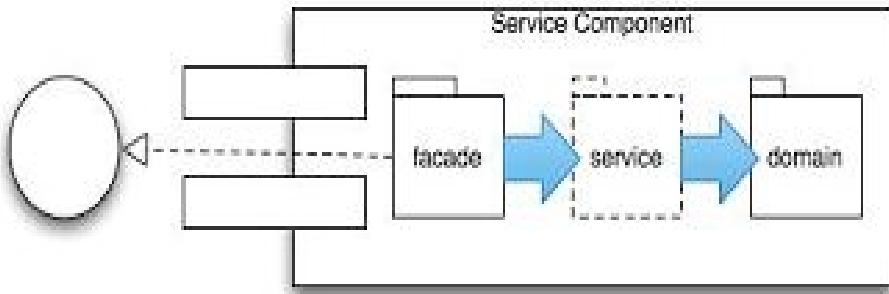
La primera aproximación al desarrollo de componentes para una típica arquitectura SOA implementada en JavaEE, consiste en implementar la lógica de negocio en clases e interfaces claramente definidos.

Un componente orientado a servicios consta de:

- Fachada: nos da acceso a una funcionalidad determinada ocultando su implementación (desacoplamiento)
- Servicio: clase en la que se implementa la lógica de negocio

#### 9.4 .Anexo IV: Web Service.

- Dominio: conjunto de clases que almacenan el estado de los servicios (sin encapsulación)



*Ilustración 9.4.3: Componente orientado a servicios*

Aquí hay una diferencia importante entre la orientación a objetos clásica que conocemos de C++ y Java: en SOA, los servicios no tienen estado, sino que se almacena en los objetos de dominio planos (POJOs) que son clases java que tienen únicamente variables, getters y setters. Dicho con otras palabras, en la arquitectura orientada a servicios se separa la funcionalidad (métodos) de los datos de los objetos (variables). Esto es contrario a los principios de la programación orienta a objetos y por lo tanto es considerado por algunos puristas como un antipatrón, como una especie de anticristo que solo hace maldades.

Entonces, ¿por qué se ha vuelto tan popular en entornos JEE?. La respuesta tiene que ver con las particularidades de los entornos web, y con la orientación de las aplicaciones empresariales hacia un modelo muy centrado en los datos. Podríamos decir que los datos son la sangre vital de las aplicaciones de hoy en día, y que esa sangre en forma de objetos DTO (Data Transfer Objects), fluye a través de distintas capas dando vida a los componentes y módulos funcionales que las integran. Estas capas se suelen estructurar de forma que cada una cumpla con una serie responsabilidades y atribuciones bien definidas:

- Capa de Presentación (web). Compuesta por controladores (clases que manejan una petición desde el navegador web hasta el servidor)
- Capa de Negocio (service). Implementa la lógica y reglas de negocio (clases que realizan los cálculos y los procesos de la app)
- Capa de Persistencia (persistence). Transacciones con la base de datos (clases que realizan operaciones CRUD sobre la BD)

El código de los distintos componentes se suele organizar en paquetes que

contienen a su vez clases para cada una de las capas independientes; típicamente web, service, y persistence.

La transferencia de información entre estas capas se hace a través de objetos DTOs planos. A este modelo de objetos (sin funcionalidad encapsulada) le han colgado el Sanbenito de Anemic Domain Model, esta etiqueta, aparte de dar la impresión de tratarse de una grave enfermedad, viene a reflejar la mala prensa que tienen entre algunos puristas de la programación, infundada a mi juicio. Estos objetos se organizan dentro un paquete domain para cada componente y se usan en todas las capas de la aplicación para transferir información, y en las operaciones transaccionales CRUD contra la base de datos.

Esta forma ‘procedural’ de diseñar y desarrollar aplicaciones Java EE ha resultado ser muy eficaz y sencilla, de tal modo que se ha convertido en la arquitectura “estándar”, sobre todo cuando se trata de aplicaciones web.

### **Fundamentos Arquitectura Orientada a servicios. Web Services.**

Los objetivos fundamentales que persigue un web service son:

- Independencia del lenguaje y de la plataforma
  - Separación de especificación de la implementación
- Interoperabilidad
  - Utilización de estándares: XML, SOAP, WSDL, UDDI...
- Acoplamiento débil: Sistemas basados en mensajes
  - Interacciones síncronas y asíncronas
- A través de Internet
  - Sin control centralizado
  - Utilización de Protocolos establecidos
  - Consideraciones de seguridad
- Modularidad y Reusabilidad de servicios
- Escalabilidad: Uno-a-uno frente a uno-a-muchos

Para poder cumplir todos esos objetivos un web service emplea estándares abiertos. Al conjunto de servicios y protocolos de los servicios Web se le denomina Web Services Protocol Stack. Los estándares empleados:

- **XML** (Extensible Markup Language): Es el formato estándar para los datos que se vayan a intercambiar.

#### 9.4 .Anexo IV: Web Service.

- **SOAP** (Simple Object Access Protocol) o **XML-RPC** (XML Remote Procedure Call): Protocolos sobre los que se establece el intercambio.
- Otros protocolos: los datos en XML también pueden enviarse de una aplicación a otra mediante protocolos normales como **HTTP** (Hypertext Transfer Protocol), **FTP** (File Transfer Protocol), o **SMTP** (Simple Mail Transfer Protocol).
- **WSDL** (Web Services Description Language): Es el lenguaje de la interfaz pública para los servicios Web. Es una descripción basada en XML de los requisitos funcionales necesarios para establecer una comunicación con los servicios Web.
- **UDDI** (Universal Description, Discovery and Integration): Protocolo para publicar la información de los servicios Web. Permite comprobar qué servicios web están disponibles.
- **WS-Security** (Web Service Security): Protocolo de seguridad aceptado como estándar por OASIS (Organization for the Advancement of Structured Information Standards). Garantiza la autenticación de los actores y la confidencialidad de los mensajes enviados.

En los siguientes apartados se profundizará más en cada uno de los estándares comentados.

La estructura de un servicio web consta de una descripción de una interfaz mediante la que acceder a la funcionalidad implementada por las distintas operaciones ofrecidas por dicha interfaz.

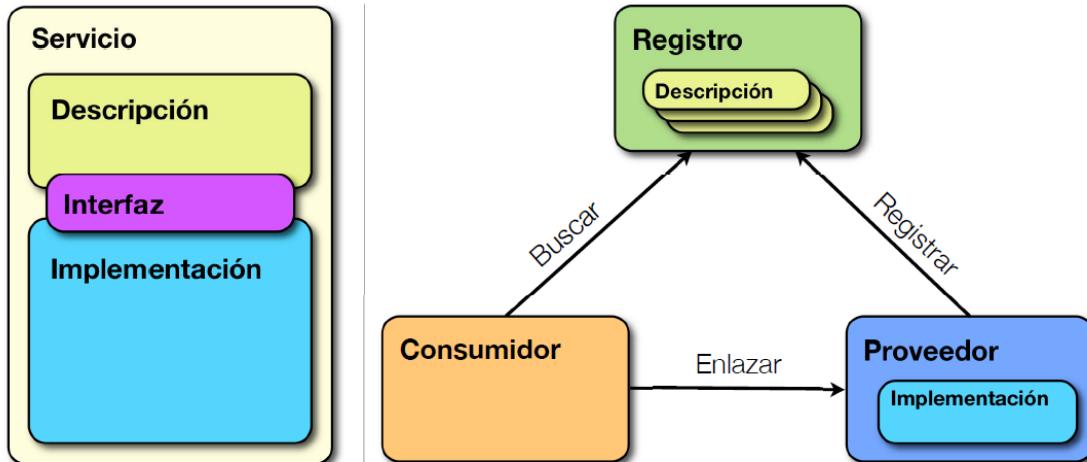


Ilustración 9.4.4: Estructura Web Service

Cuando un consumidor desea usar un servicio web puede acceder al

registro de servicios publicados mediante UDDI y comprobar que servicios web están disponibles. En caso de existir un servicio web de interés para el consumidor se le facilitará la descripción del mismo y este podrá enlazar con el proveedor y usar las operaciones disponibles en la interfaz que proporcione. Este proceso puede observarse en la figura 9.4.4.

Para llevar a cabo este proceso se emplean los estándares comentados anteriormente. Todas las comunicaciones se basan en el empleo de mensajes construidos sobre el lenguaje de marcas XML. El registro y consulta de los de los servicios web se realiza mediante el protocolo UDDI. La descripción de la interfaz pública del servicio web se escribe en WSDL, basado en XML. En la figura 9.4.5. El proceso de comunicación comentado anteriormente entre el proveedor del servicio web y el consumidor se lleva a cabo mediante mensajes SOAP, solicitud y respuesta.

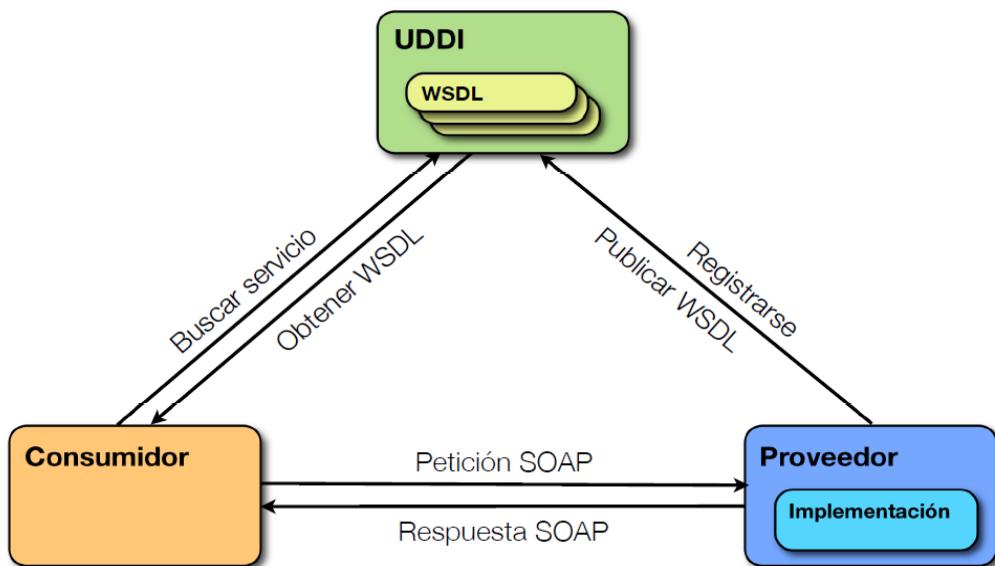
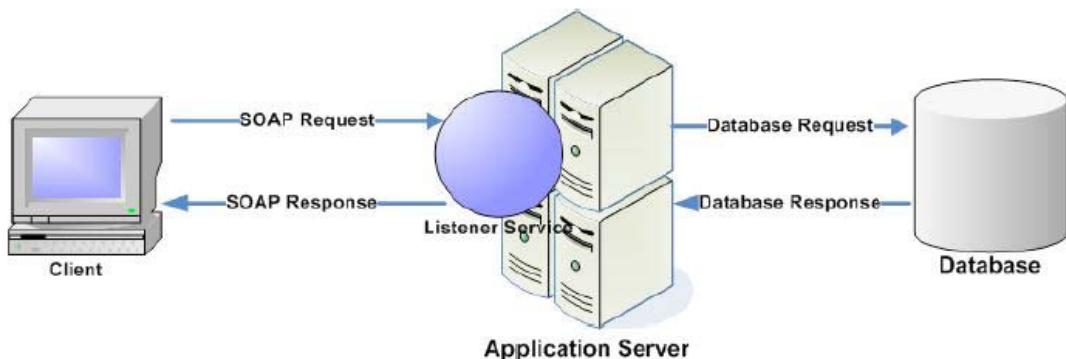


Ilustración 9.4.5: Estándares empleados en proceso de comunicación

Para una mejor comprensión vamos a detallar los procesos mostrados en la figura 9.4.5.

### SOAP

Los mensajes utilizados para llevar a cabo la comunicación entre el consumidor y el proveedor del servicio se basan en el protocolo SOAP.



*Ilustración 9.4.6: Comunicación consumidor-proveedor web services*

SOAP (siglas de Simple Object Access Protocol) es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. Este protocolo deriva de un protocolo creado por David Winer en 1998, llamado XML-RPC. SOAP fue creado por Microsoft, IBM y otros y está actualmente bajo el auspicio de la W3C. Es uno de los protocolos utilizados en los servicios Web.

SOAP proporciona un mecanismo estándar de empaquetar mensajes. SOAP ha recibido gran atención debido a que facilita una comunicación del estilo RPC entre un cliente y un servidor remoto. Pero existen multitud de protocolos creados para facilitar la comunicación entre aplicaciones, incluyendo RPC de Sun, DCE de Microsoft, RMI de Java y ORPC de CORBA. ¿Por qué se presta tanta atención a SOAP?

Una de las razones principales es que SOAP ha recibido un increíble apoyo por parte de la industria. SOAP es el primer protocolo de su tipo que ha sido aceptado prácticamente por todas las grandes compañías de software del mundo. Compañías que en raras ocasiones cooperan entre sí están ofreciendo su apoyo a este protocolo. Algunas de las mayores Compañías que soportan SOAP son Microsoft, IBM, SUN, Microsystems, SAP y Ariba.

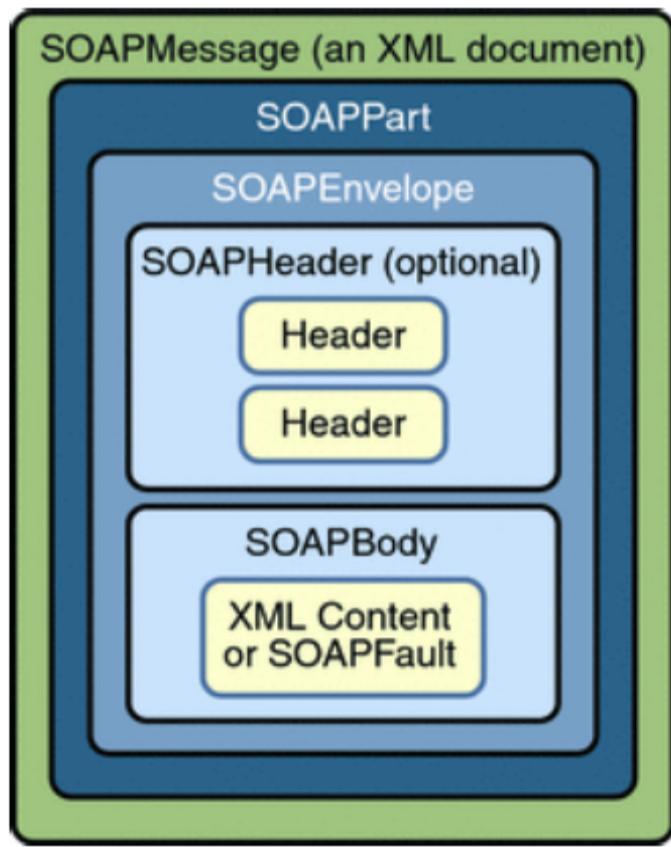
Algunas de las Ventajas de SOAP son:

- No está asociado con ningún lenguaje: los desarrolladores involucrados en nuevos proyectos pueden elegir desarrollar con el último y mejor lenguaje de programación que exista pero los desarrolladores responsables de mantener antiguas aflicciones heredadas podrían no poder hacer esta elección sobre el lenguaje de programación que utilizan. SOAP no especifica una API, por lo que la implementación de la API se deja al lenguaje de programación, como en Java, y la plataforma como Microsoft .Net.
- No se encuentra fuertemente asociado a ningún protocolo de transporte:

La especificación de SOAP no describe como se deberían asociar los mensajes de SOAP con HTTP. Un mensaje de SOAP no es más que un documento XML, por lo que puede transportarse utilizando cualquier protocolo capaz de transmitir texto.

- No está atado a ninguna infraestructura de objeto distribuido La mayoría de los sistemas de objetos distribuidos se pueden extender, y ya lo están alguno de ellos para que admitan SOAP.
- Aprovecha los estándares existentes en la industria: Los principales contribuyentes a la especificación SOAP evitaron, intencionadamente, reinventar las cosas. Optaron por extender los estándares existentes para que coincidieran con sus necesidades. Por ejemplo, SOAP aprovecha XML para la codificación de los mensajes, en lugar de utilizar su propio sistema de tipo que ya están definidas en la especificación esquema de XML. Y como ya se ha mencionado SOAP no define un medio de transporte de los mensajes; los mensajes de SOAP se pueden asociar a los protocolos de transporte existentes como HTTP y SMTP.
- Permite la interoperabilidad entre múltiples entornos: SOAP se desarrolló sobre los estándares existentes de la industria, por lo que las aplicaciones que se ejecuten en plataformas con dicho estándares pueden comunicarse mediante mensaje SOAP con aplicaciones que se ejecuten en otras plataformas. Por ejemplo, una aplicación de escritorio que se ejecute en una PC puede comunicarse con una aplicación del back-end ejecutándose en un mainframe capaz de enviar y recibir XML sobre HTTP.

SOAP proporciona un mecanismo estándar de empaquetar un mensaje. Un mensaje SOAP se compone de un sobre que contiene el cuerpo del mensaje y cualquier información de cabecera que se utiliza para describir el mensaje. A continuación se muestra la estructura de un mensaje SOAP:



*Ilustración 9.4.7: Estructura mensaje SOAP*

- Sobre (SOAPEnvelope): Elemento raíz del documento. Contiene dos subelementos: el Body y el Header. También puede contener otros elementos hijo.
- Cuerpo (SOAPBody): El cuerpo Body contiene la información principal del mensaje, es decir, la carga de datos del mensaje que se conoce como carga útil.
- Cabeceras (SOAPHeader): La cabecera Header es opcional y contiene información que describe el mensaje, datos adicionales que no necesariamente pertenecen al cuerpo del mensaje, y también metadatos sobre enrutamiento (routing), seguridad o transacciones.

Un ejemplo de mensaje de petición de un producto y respuesta de mensaje SOAP entre un consumidor y un proveedor sería:

```
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

  <soap:Body>
    <getProductDetails
      xmlns="http://warehouse.example.com/ws">
      <productId>827635</productId>
    </getProductDetails>
  </soap:Body>

</soap:Envelope>
```

*Ilustración 9.4.8: Mensaje SOAP request*

```
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

  <soap:Body>
    <getProductDetailsResponse
      xmlns="http://warehouse.example.com/ws">
      <getProductDetailsResult>
        <productName>Toptimate 3-Piece Set</productName>
        <productId>827635</productId>
        <description>3-Piece luggage set</description>
        <price>96.50</price>
        <inStock>true</inStock>
      </getProductDetailsResult>
    </getProductDetailsResponse>
  </soap:Body>

</soap:Envelope>
```

*Ilustración 9.4.9: Mensaje SOAP response*

## WSDL

Como ya se ha mencionado, WSDL es un formato XML para describir servicios Web.

La versión 1.0 fue la primera recomendación por parte del W3C y la versión 1.1 no alcanzó nunca tal estatus. La versión 2.0 se convirtió en la recomendación actual por parte de dicha entidad.

WSDL describe la interfaz pública a los servicios Web. Está basado en XML y describe la forma de comunicación, es decir, los requisitos del protocolo y los formatos de los mensajes necesarios para interactuar con los servicios listados en su catálogo. Las operaciones y mensajes que soporta se describen en abstracto y se ligan después al protocolo concreto de red y al formato del mensaje.

Un programa cliente que se conecta a un servicio web puede leer el WSDL para determinar qué funciones están disponibles en el servidor. Los tipos de datos especiales se incluyen en el archivo WSDL en forma de XML Schema.

El WSDL nos permite tener una descripción de un servicio web. Especifica la interfaz abstracta a través de la cual un cliente puede acceder al servicio y los detalles de cómo se debe utilizar.

**<definitions>**: Raíz del documento XML

**<types>**: Tipos de datos se intercambiarán

**<message>**: Formato de cada mensaje, tanto los parámetros como el resultado

**<portType>**: Operaciones (funciones) soportadas por el servicio

**<binding>**: Protocolo con el que se transmitirán los mensajes (SOAP, HTTP, MIME, etc.)

**<service>**: Dirección (URL) del servicio

*Ilustración 9.4.10: Estructura WSDL*

Como vemos en la figura la estructura del WSDL tiene los siguientes elementos:

- **Tipos de Datos**
  - <types>: Esta sección define los tipos de datos usados en los mensajes. Se utilizan los tipos definidos en la especificación de esquemas XML.
- **Mensajes**
  - <message>: Aquí definimos los elementos de mensaje. Cada mensaje puede consistir en una serie de partes lógicas. Las partes pueden ser de cualquiera de los tipos definidos en la sección anterior.
- **Tipos de Puerto**
  - <portType>: Con este apartado definimos las operaciones permitidas y los mensajes intercambiados en el Servicio.
- **Bindings**
  - <binding>: Especificamos los protocolos de comunicación usados.
- **Servicios**
  - <service>: Conjunto de puertos y dirección de los mismos. Esta parte final hace referencia a lo aportado por las secciones anteriores.

Con estos elementos no sabemos que hace un servicio pero si disponemos de la información necesaria para interactuar con él (funciones, mensajes de entrada/salida, protocolos...).

A continuación se muestra un ejemplo sencillo de un documento WSDL para el servicio Hello World y sus diferentes secciones.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="HelloService"
  targetNamespace="http://www.ecerami.com/wsdl/
HelloService.wsdl"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://www.ecerami.com/wsdl/
HelloService.wsdl"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <message name="SayHelloRequest">
    <part name="firstName" type="xsd:string"/>
  </message>
  <message name="SayHelloResponse">
    <part name="greeting" type="xsd:string"/>
  </message>

  <portType name="Hello_PortType">
    <operation name="sayHello">
      <input message="tns:SayHelloRequest"/>
      <output message="tns:SayHelloResponse"/>
    </operation>
  </portType>

```

Ilustración 9.4.11: Ejemplo WSDL

```

<binding name="Hello_Binding" type="tns:Hello_PortType">
  <soap:binding style="rpc"
    transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="sayHello">
    <soap:operation soapAction="sayHello"/>
    <input>
      <soap:body
        encodingStyle="http://schemas.xmlsoap.org/soap/
encoding/">
        namespace="urn:examples:helloservice"
        use="encoded"/>
    </input>
    <output>
      <soap:body
        encodingStyle="http://schemas.xmlsoap.org/soap/
encoding/">
        namespace="urn:examples:helloservice"
        use="encoded"/>
    </output>
  </operation>
</binding>

<service name="Hello_Service">
  <documentation>WSDL for HelloService</documentation>
  <port binding="tns:Hello_Binding" name="Hello_Port">
    <soap:address
      location="http://localhost/rpcrouter"/>
  </port>
</service>
</definitions>

```

Ilustración 9.4.12: Ejemplo WSDL (continuación)

En el anterior ejemplo podemos apreciar cada una de las partes descritas para un archivo WSDL.

Para poder hacer un uso más profundo de este estándar vamos a profundizar un poco más en algunos de los elementos que forman el archivo WSDL.

### Importación de documentos y espacio de nombres

A las definiciones dentro de un documento WSDL se les puede asignar un atributo nombre opcional de tipo NCNAME. Opcionalmente puede ir acompañado de un atributo *targetNamespace*. WSDL permite asociar un espacio de nombres con una localización de un documento usando una declaración de importación:

```
<Definitions .... >
  <import namespace="uri" location="uri"/> *
</ Definitions>
```

Las referencias a definiciones WSDL deben realizarse mediante la utilización de un QName. Los tipos de definiciones que pueden ser referenciadas en un WSDL son:

- WSDL definiciones: servicio, puerto, mensajes, enlaces y portType
- Otras definiciones: si las definiciones adicionales se añaden usando extensibilidad, deben utilizar un QName enlace.

### Tipos

El elemento de tipo encierra definiciones de tipos de datos que son relevantes para los mensajes intercambiados. Para la máxima interoperabilidad y la neutralidad, WSDL prefiere el uso de XSD como el sistema de tipo canónico, y lo trata como el sistema de tipo intrínseco.

```
<Definiciones .... >
  <types>
    <xsd: schema .... /> *
  </ types>
</ Definiciones>
```

El sistema de tipos XSD se puede utilizar para definir los tipos en un mensaje, independientemente de si el formato de conexión resultante es en realidad XML, o si el esquema XSD resultante valida el formato de conexión en particular.

Sin embargo, dado que es irracional esperar que una gramática de tipo de datos del sistema se pueda utilizar para describir todos los tipos abstractos presentes y futuros, WSDL permite que los esquemas de tipos de datos que se puedan añadir a través de elementos de extensibilidad. Un elemento de extensibilidad puede aparecer bajo el elemento de tipo para identificar la definición de tipo a usar en el sistema y para proporcionar un elemento contenedor XML para las definiciones de tipo. El papel de este elemento puede ser comparado a la del elemento de esquema del lenguaje Esquema XML.

```
<Definitions .... >  
  <types>  
    < type -system extensibility element-> *  
  </ Types>  
</ Definitions>
```

### Mensajes

Cada mensaje puede estar constituido por una o más partes lógicas. Cada una de las partes está asociada a uno de los tipos definidos en el elemento **types**. Para ello se usa el atributo **type**.

```
<definitions .... >  
  <message name="nmtoken"> *  
    <part name="nmtoken" element="qname"? type="qname"?/>> *  
  </message>  
</definitions>
```

Un ejemplo práctico:

```
<definitions .... >  
  <types>  
    <schema .... >  
      <element name="PO" type="tns:POType"/>  
      <complexType name="POType">  
        <all>  
          <element name="id" type="string"/>  
          <element name="name" type="string"/>  
          <element name="items">  
            <complexType>  
              <all>  
                <element name="item" type="tns:Item"  
minOccurs="0" maxOccurs="unbounded"/>  
              </all>  
            </complexType>  
          </element>  
        </all>  
      </complexType>  
    </schema>  
  </types>  
</definitions>
```

#### 9.4 .Anexo IV: Web Service.

```
<complexType name="Item">
  <all>
    <element name="quantity" type="int"/>
    <element name="product" type="string"/>
  </all>
</complexType>
<element name="Invoice" type="tns:InvoiceType"/>
<complexType name="InvoiceType">
  <all>
    <element name="id" type="string"/>
  </all>
</complexType>
</schema>
</types>

<message name="PO">
  <part name="po" element="tns:PO"/>
  <part name="invoice" element="tns:Invoice"/>
</message>
</definitions>
```

#### Tipos de Puerto

Un tipo de puerto es nombre para un conjunto de operaciones y los mensajes involucrados en ellas.

```
<wsdl:definitions .... >
  <wsdl:portType name="nmtoken">
    <wsdl:operation name="nmtoken" .... /> *
  </wsdl:portType>
</wsdl:definitions>
```

El atributo nombre debe de ser único dentro del conjunto de tipos de puertos definidos en el documento WSDL. Las operaciones son llamadas utilizando este atributo.

WSDL soporta cuatro primitivas de transmisión:

- **De un solo sentido (One-way).** El receptor recibe un mensaje.

- **Petición-respuesta (Request-response).** El receptor recibe un mensaje, y envía un mensaje correlacionado.
- **Petición-respuesta (Solicit-response).** El receptor envía un mensaje, y recibe un mensaje correlacionado.
- **Notificación.** El receptor envía un mensaje.

El formato para una operación One-way es:

```
<wsdl:definitions .... > <wsdl:portType .... > *
  <wsdl:operation name="nmtoken">
    <wsdl:input name="nmtoken"? message="qname"/>
  </wsdl:operation>
</wsdl:portType >
</wsdl:definitions>
```

El formato para una operación Request-response es:

```
<wsdl:definitions .... >
  <wsdl:portType .... > *
    <wsdl:operation name="nmtoken" parameterOrder="nmtokens">
      <wsdl:input name="nmtoken"? message="qname"/>
      <wsdl:output name="nmtoken"? message="qname"/>
      <wsdl:fault name="nmtoken" message="qname"/>*
    </wsdl:operation>
  </wsdl:portType >
</wsdl:definitions>
```

El formato para una operación Solicit-response es:

```
<wsdl:definitions .... >
  <wsdl:portType .... > *
    <wsdl:operation name="nmtoken" parameterOrder="nmtokens">
      <wsdl:output name="nmtoken"? message="qname"/>
      <wsdl:input name="nmtoken"? message="qname"/>
      <wsdl:fault name="nmtoken" message="qname"/>*
    </wsdl:operation>
  </wsdl:portType >
</wsdl:definitions>
```

El formato para una operación Notification es:

```
<wsdl:definitions .... >
  <wsdl:portType .... > *
    <wsdl:operation name="nmtoken">
      <wsdl:output name="nmtoken"? message="qname"/>
    </wsdl:operation>
  </wsdl:portType >
```

#### 9.4 .Anexo IV: Web Service.

```
</wsdl:definitions>
```

Un ejemplo práctico de definición de tipos de puerto puede ser el WSDL perteneciente al Web Service del presente proyecto:

```
<portType name="NewsService">
    <operation name="addNoticia">
        <input wsam:Action =
            "http://service.blufeedme/NewsService/addNoticiaRequest"
        message="tns:addNoticia"/>
        <output wsam:Action =
            "http://service.blufeedme/NewsService/addNoticiaResponse"
        message="tns:addNoticiaResponse"/>
        <fault message="tns:InvalidUserException"
        name="InvalidUserException" wsam:Action =
            "http://service.blufeedme/NewsService/addNoticia/Fault/InvalidUserException"/>
        <fault message="tns:InvalidParamException"
        name="InvalidParamException" wsam:Action =
            "http://service.blufeedme/NewsService/addNoticia/Fault/InvalidParamException"/>
    </operation>
    <operation name="deleteNoticia">
        <input wsam:Action =
            "http://service.blufeedme/NewsService/deleteNoticiaRequest"
        message="tns:deleteNoticia"/>
        <output wsam:Action =
            "http://service.blufeedme/NewsService/deleteNoticiaResponse"
        message="tns:deleteNoticiaResponse"/>
        <fault message="tns:InvalidUserException"
        name="InvalidUserException" wsam:Action =
            "http://service.blufeedme/NewsService/deleteNoticia/Fault/InvalidUserException"/>
        <fault message="tns:InvalidParamException"
        name="InvalidParamException" wsam:Action =
            "http://service.blufeedme/NewsService/deleteNoticia/Fault/InvalidParamException"/>
    </operation>
    .
    .
    .
</portType>
```

### Bindings

Un binding define el formato del mensaje y detalles del protocolo para las operaciones y mensajes definidos para un tipo de puerto concreto. El número de bindings debe de ser igual al número de tipos de puerto definido. La gramática para definir los bindings es:

```
<wsdl:definitions .... >
  <wsdl:binding name="nmtoken" type="qname"> *
    <!-- extensibility element (1) --> *
    <wsdl:operation name="nmtoken"> *
      <!-- extensibility element (2) --> *
      <wsdl:input name="nmtoken"? > ?
        <!-- extensibility element (3) -->
      </wsdl:input>
      <wsdl:output name="nmtoken"? > ?
        <!-- extensibility element (4) --> *
      </wsdl:output>
      <wsdl:fault name="nmtoken"> *
        <!-- extensibility element (5) --> *
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
</wsdl:definitions>
```

El nombre del *binding*, atributo *name*, debe de ser único entre todos los bindings definidos en el archivo WSDL. Para cada operación se especifica el formato de que adoptará. El protocolo se puede especificar el mismo para todos los mensajes usando:

```
<binding name="NewsServicePortBinding" type="tns:NewsService">
  <soap:binding    transport="http://schemas.xmlsoap.org/soap/http"
  style="document"/>
  <operation name="addNoticia">
    <soap:operation soapAction="" />
    <input><soap:body use="literal"/>
    </input>
    <output><soap:body use="literal"/>
    </output>
    <fault name="InvalidUserException">
      <soap:fault
      name="InvalidUserException"
      use="literal"/>
    </fault>
    <fault name="InvalidParamException">
```

## 9.4 Anexo IV: Web Service.

```
<soap:fault name="InvalidParamException"  
use="literal"/>  
    </fault>  
  </operation>  
  ...  
</binding>
```

Mediante el atributo *use* especificamos el tipo de formato del mensaje. Este puede ir sin ningún tipo de codificación o codificado. Para cada tipo de mensaje disponemos de:

- **soap:body:**

```
<soap:body parts="nmtokens"? use="literal|encoded"?  
encodingStyle="uri-list"? namespace="uri"?>
```

- **soap:fault:**

```
<definitions .... >  
  <binding .... >  
    <operation .... >  
      <fault>*  
        <soap:fault name="nmtoken" use="literal|encoded"  
          encodingStyle="uri-list"?  
          namespace="uri"?>  
        </fault>  
      </operation>  
    </binding>  
</definitions>
```

- **soap:header and soap:headerfault:**

```
<definitions...>  
  <binding .... >  
    <operation .... >  
      <input>  
        <soap:header message="qname" part="nmtoken" use="literal|  
        encoded"  
          encodingStyle="uri-list"? namespace="uri"?>*  
        <soap:headerfault message="qname" part="nmtoken"  
        use="literal|encoded"  
          encodingStyle="uri-list"?  
          namespace="uri"?>*  
        <soap:header message="qname" part="nmtoken"  
        use="literal|encoded"  
          encodingStyle="uri-list"? namespace="uri"?>  
      </input>  
      <output>  
        <soap:header message="qname" part="nmtoken"  
        use="literal|encoded"  
          encodingStyle="uri-list"? namespace="uri"?>  
      </output>  
    </operation>  
  </binding>  
</definitions>
```

```
use="literal|encoded" encodingStyle="uri-list"?
namespace="uri"?/*>
    <soap:header>
        </output>
    </operation>
</binding>
</definitions>
```

## UDDI

UDDI son las siglas del catálogo de negocios de Internet denominado Universal Description, Discovery and Integration. El registro en el catálogo se hace en XML. UDDI es una iniciativa industrial abierta (sufragada por la OASIS) entroncada en el contexto de los servicios Web. El registro de un negocio en UDDI tiene tres partes:

- Páginas blancas - dirección, contacto y otros identificadores conocidos.
- Páginas amarillas - categorización industrial basada en taxonomías.
- Páginas verdes - información técnica sobre los servicios que aportan las propias empresas.

La información contenida en cada una de las partes puede observarse en la figura 9.4.13.

- **Business Entity:** Información sobre la compañía que publica un servicio.
- **Business Service:** Información sobre el servicio.
- **Binding Template:** Descripción técnica sobre la implementación de un servicio (un servicio puede tener varias implementaciones, y por tanto varias binding templates)
- **tModel:** Modelo técnico para definir las diferentes estructuras de datos almacenadas en el registro.

UDDI es uno de los estándares básicos de los servicios Web cuyo objetivo es ser accedido por los mensajes SOAP y dar paso a documentos WSDL, en los que se describen los requisitos del protocolo y los formatos del mensaje solicitado para interactuar con los servicios Web del catálogo de registros.

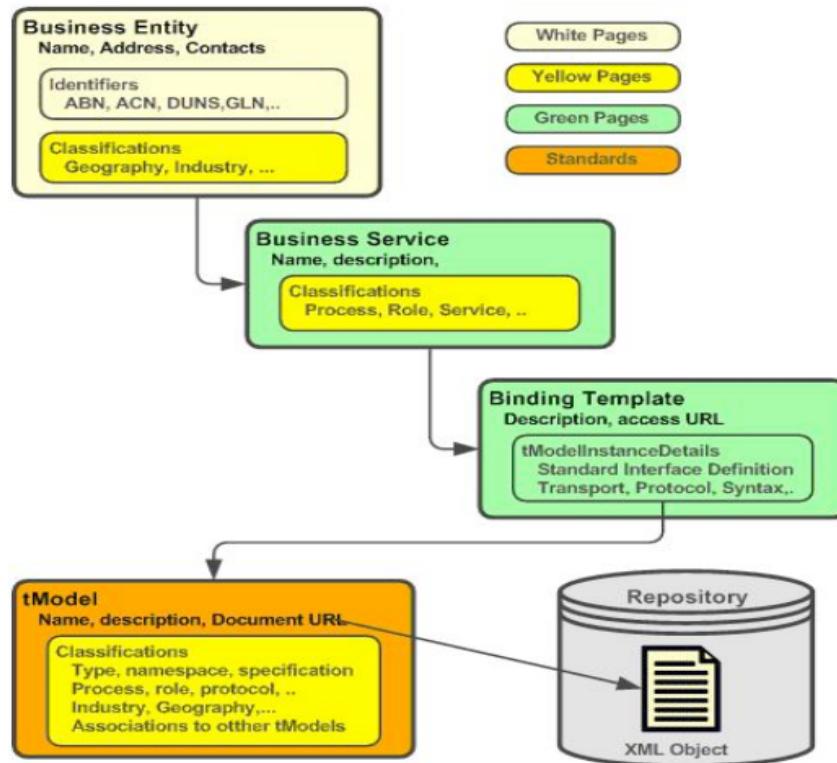


Ilustración 9.4.13: Estructura UDDI

### Web Services JAVA

Las aplicaciones típicamente están compiladas en un bytecode, un código intermedio más abstracto que el lenguaje máquina. En el tiempo de ejecución, el bytecode es normalmente interpretado o compilado a código nativo para la ejecución. Una aplicación java es empaquetada en un archivo .jar. En el caso de aplicaciones web se utilizan para empaquetarla archivos .war. Un archivo .war es un archivo JAR utilizado para distribuir una colección de JavaServer Pages, servlets, clases Java, archivos XML, librerías de tags y páginas web estáticas (HTML y archivos relacionados).

## 9 .Anexo IV: Web Service.

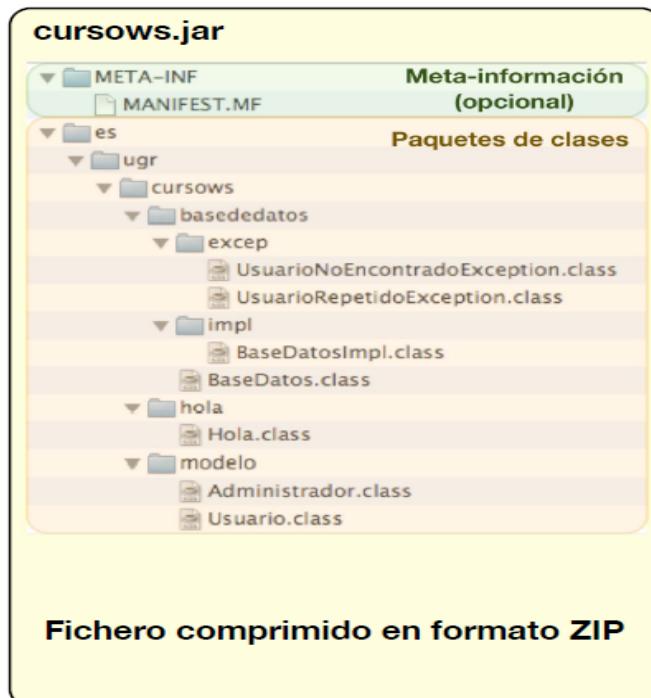


Ilustración 9.4.14: Archivo .jar



Ilustración 9.4.15: Archivo .war

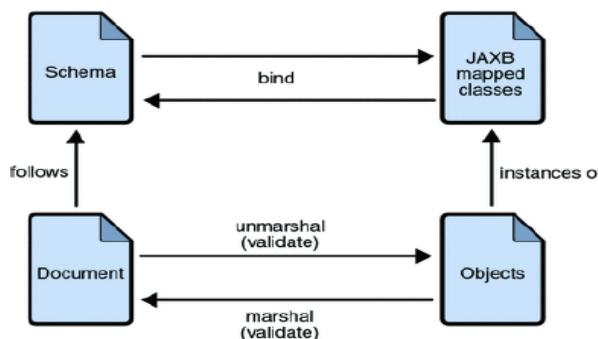
### JAXB

Como se ha indicado en apartados anteriores los servicios web se especifican usando archivos basados en XML. Mediante el WSDL se obtiene la descripción del servicio web. En esta descripción se describen tipos de datos siguiendo el formato de los schemas XML.

Java facilita una API para ligar archivos de esquemas XML a representaciones de código Java. Esta API es JAXB (Java Architecture for XML Binding).

JAXB simplifica el acceso a un documento XML de un programa java representando el documento XML en un programa en formato Java, esto es, provee a los desarrolladores de aplicaciones Java, una forma rápida y conveniente para enlazar o vincular esquemas XML a representaciones Java.

JAXB provee de métodos para desorganizar (unmarshal) documentos instancias de XML en árboles de contenido, para después utilizar los mismos y generar mediante el método organizar (marshal) instancias XML de las que fueron generados.



*Ilustración 9.4.16: JAXB; Operaciones Unmarshal y Marshal*

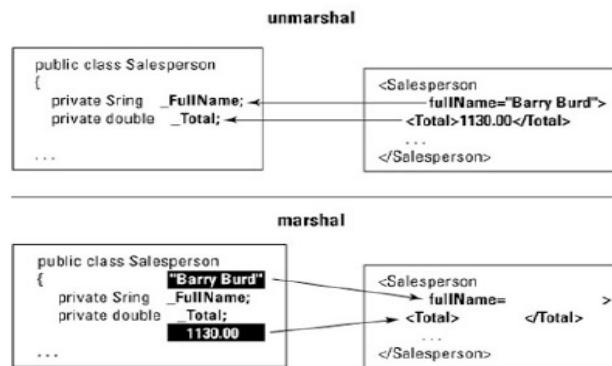


Ilustración 9.4.17: JAXB; Operaciones Unmarshal y Marshal II

Esto nos proporciona la flexibilidad de manejar datos XML en una plataforma “neutral” además de no requerir tratar o conocer las técnicas de programación de XML al ocultar ciertos detalles complejos de sus relaciones.

Las clases generadas JAXB describen solo relación real definida en los esquemas fuentes. El resultado de lo anterior son datos xml altamente portables que unido a un código java portable puede ser usado para crear flexibles y ligeras aplicaciones y servicios web.

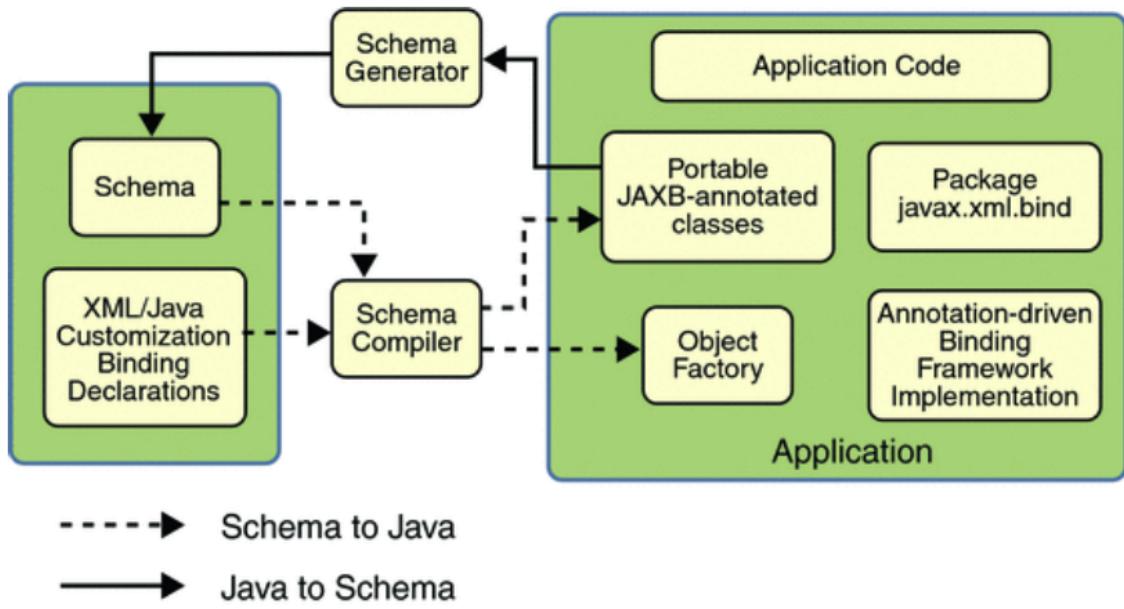


Ilustración 9.4.18: API JAXB

Para realizar la unión entre el schema y las clases de una aplicación Java, se usa el lenguaje de unión (binding declarations) y el compilador de esquema.

#### 9.4 .Anexo IV: Web Service.

---

El compilador de esquema genera un conjunto de ficheros fuente Java desde el schema basándose en las instrucciones en las instrucciones que proporcionamos en el esquema de unión. No necesitamos proporcionar una instrucción de unión para cada declaración del DTD para generar las clases Java. El compilador de esquema hace ciertas asunciones basadas en el DTD si nuestro esquema de unión no especifica completamente cómo debería unirse cada declaración del DTD al código. Por ejemplo, el compilador de esquema utiliza un algoritmo general de mapeo de nombres para unir nombres XML a nombres que son aceptables en el lenguaje de programación Java. En este caso, podemos utilizar el esquema de unión para hacer que el compilador de esquema genere nombres diferentes. Hay muchas otras personalizaciones que podemos hacer con el esquema de unión, incluyendo:

- Nombrar el paquete, las clases derivadas y los métodos.
- Asignar tipos a los métodos dentro de las clases derivadas.
- Elegir los elementos a unir a las clases.
- Decidir cómo unir cada declaración de atributo y de elemento a una propiedad en la clase de contenido apropiada.
- Crear constructores personalizados, interfaces, y enumeraciones.
- Elegir el tipo de cada valor de atributo o especificación de contenido.

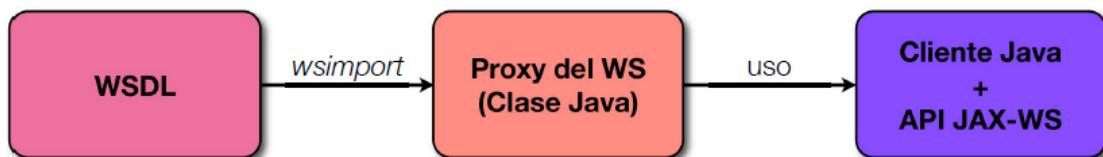
Para obtener los schemas a partir de las clases java se usaría el generador de esquema.

### JAX-WS

Es la API de Java para la creación de servicios web. Forma parte de la plataforma Java EE de Sun Microsystems. Como el resto de APIs de Java, JAX-WS emplea anotaciones, introducida en Java SE 5, para simplificar el desarrollo y despliegue de servicios web y clientes.



**Creación de un servicio Web en Java**



**Creación de un cliente de un servicio Web en Java**

## **9.5 Anexo V: Bluetooth:JSR 82**

### **Bluetooth**

Bluetooth es una tecnología de radio de corto alcance, que permite conectividad inalámbrica entre dispositivos remotos. Se diseñó pensando básicamente en tres objetivos: pequeño tamaño, mínimo consumo y bajo precio.

#### **Nociones sobre Bluetooth**

Bluetooth opera en la banda libre de radio ISM<sup>9</sup> a 2.4 Ghz. Su máxima velocidad de transmisión de datos es de 1 Mbps. El rango de alcance Bluetooth depende de la potencia empleada en la transmisión. La mayor parte de los dispositivos que usan Bluetooth transmiten con una potencia nominal de salida de 0 dBm, lo que permite un alcance de unos 10 metros en un ambiente libre de obstáculos.

#### **Salto de frecuencia:**

Debido a que la banda ISM está abierta a cualquiera, el sistema de radio Bluetooth deberá estar preparado para evitar las múltiples interferencias que se pudieran producir. Éstas pueden ser evitadas utilizando un sistema que busque una parte no utilizada del espectro o un sistema de salto de frecuencia.

En este caso la técnica de salto de frecuencia es aplicada a una alta velocidad y una corta longitud de los paquetes (1600 saltos/segundo). Con este sistema se divide la banda de frecuencia en varios canales de salto, donde, los transceptores, durante la conexión van cambiando de uno a otro canal de salto de manera pseudo-aleatoria.

Los paquetes de datos están protegido por un esquema ARQ (repetición automática de consulta), en el cual los paquetes perdidos son automáticamente retransmitidos.

#### **El canal:**

Bluetooth utiliza un sistema FH/TDD (salto de frecuencia/división de tiempo duplex), en el que el canal queda dividido en intervalos de 625  $\mu$ s, llamados slots, donde cada salto de frecuencia es ocupado por un slot.

Dos o más unidades Bluetooth pueden compartir el mismo canal dentro de una piconet (pequeña red que establecen automáticamente los terminales Bluetooth para comunicarse entre sí), donde una unidad actúa como maestra, controlando el tráfico de datos en la piconet que se genera entre las demás unidades, donde éstas actúan como esclavas, enviando y recibiendo señales hacia el maestro.

El salto de frecuencia del canal está determinado por la secuencia de la señal, es decir, el orden en que llegan los saltos y por la fase de esta secuencia.

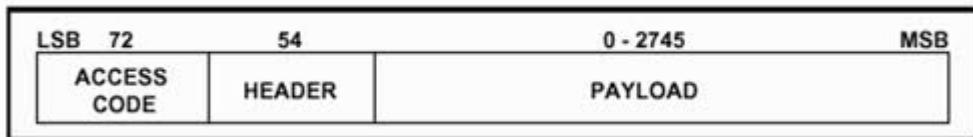
---

<sup>9</sup> Banda internacional médico-científica

En Bluetooth, la secuencia queda fijada por la identidad de la unidad maestra de la piconet (un código único para cada equipo), y por su frecuencia de reloj.

#### **Datagrama Bluetooth:**

La información que se intercambia entre dos unidades Bluetooth se realiza mediante un conjunto de slots que forman un paquete de datos. Cada paquete comienza con un código de acceso de 72 bits, que se deriva de la identidad maestra, seguido de un paquete de datos de cabecera de 54 bits. Éste contiene importante información de control, como tres bits de acceso de dirección, tipo de paquete, bits de control de flujo, bits para la retransmisión automática de la pregunta, y chequeo de errores de campos de cabecera. La dirección del dispositivo es en forma hexadecimal. Finalmente, el paquete que contiene la información, que puede seguir al de la cabecera, tiene una longitud de 0 a 2745 bits.

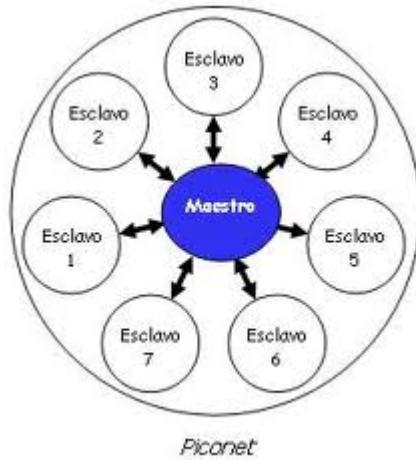


*Ilustración 9.5.1: Formato cabecera Bluetooth*

En cualquier caso, cada paquete que se intercambia en el canal está precedido por el código de acceso. Los receptores de la piconet comparan las señales que reciben con el código de acceso, si éstas no coinciden, el paquete recibido no es considerado como válido en el canal y el resto de su contenido es ignorado.

#### **Piconets:**

Como hemos citado anteriormente si un equipo se encuentra dentro del radio de cobertura de otro, éstos pueden establecer conexión entre ellos. Cada dispositivo tiene una dirección única de 48 bits, basada en el estándar IEEE 802.11 para WLAN. En principio sólo son necesarias un par de unidades con las mismas características de hardware para establecer un enlace. Dos o más unidades Bluetooth que comparten un mismo canal forman una piconet.



*Ilustración 9.5.2: Piconet*

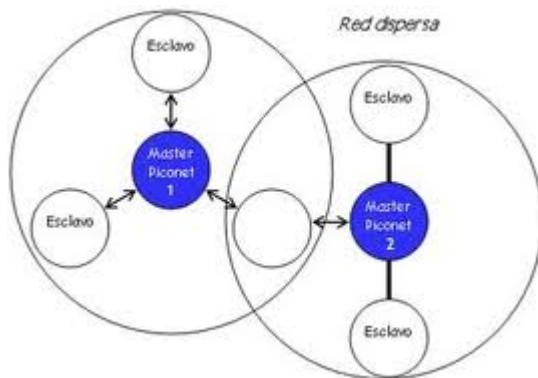
Para regular el tráfico en el canal, una de las unidades participantes se convertirá en maestra, pero por definición, la unidad que establece la piconet asume éste papel y todos los demás serán esclavos. Los participantes podrían intercambiar los papeles si una unidad esclava quisiera asumir el papel de maestra. Sin embargo sólo puede haber un maestro en la piconet al mismo tiempo. Hasta ocho usuarios o dispositivos pueden formar una piconet y hasta diez piconets pueden coexistir en una misma área de cobertura.

#### **Medios y velocidades:**

Además de los canales de datos, están habilitados tres canales de voz de 64 kbit/s por piconet. Las conexiones son uno a uno con un rango máximo de diez metros, aunque utilizando amplificadores se puede llegar hasta los 100 metros, pero en este caso se introduce alguna distorsión. Los datos se pueden intercambiar a velocidades de hasta 1 Mbit/s. El protocolo bandabase que utiliza Bluetooth combina las técnicas de circuitos y paquetes para asegurar que los paquetes llegan en orden.

#### **Scatternet:**

Los equipos que comparten un mismo canal sólo pueden utilizar una parte de su capacidad. Aunque los canales tienen un ancho de banda de un 1Mbit, cuantos más usuarios se incorporan a la piconet, disminuye la capacidad hasta unos 10 kbit/s más o menos. Para poder solucionar este problema se adoptó una solución de la que nace el concepto de scatternet.



Las unidades que se encuentran en el mismo radio de cobertura pueden establecer potencialmente comunicaciones entre ellas. Sin embargo, sólo aquellas unidades que realmente quieran intercambiar información comparten un mismo canal creando la piconet. Este hecho permite que se creen varias piconets en áreas de cobertura superpuestas.

A un grupo de piconets se le llama scatternet. El rendimiento, en conjunto e individualmente de los usuarios de una scatternet es mayor que el que tiene cada usuario cuando participa en un mismo canal de 1 Mbit. Además, estadísticamente se obtienen ganancias por multiplexación y rechazo de canales salto. Debido a que individualmente cada piconet tiene un salto de frecuencia diferente, diferentes piconets pueden usar simultáneamente diferentes canales de salto.

### Establecimiento de la conexión

La conexión con un dispositivo, se hace mediante un mensaje page. Si la dirección es desconocida, antes del mensaje page se necesitará un mensaje inquiry. Antes de que se produzca ninguna conexión, se dice que todos los dispositivos están en modo standby.

Un dispositivo en modo standby se despierta cada 1.28 segundos para escuchar posibles mensajes page/inquiry. Cada vez que un dispositivo se despierta, escucha una de las 32 frecuencias de salto definidas. Un mensaje de tipo page, será enviado en 32 frecuencias diferentes. Primero el mensaje es enviado en las primeras 16 frecuencias (128 veces), y si no se recibe respuesta, el maestro mandará el mensaje page en las 16 frecuencias restantes (128 veces). El tiempo máximo de intento de conexión es de 2.56 segundos.

En el estado conectado, el dispositivo Bluetooth puede encontrarse en varios modos de operación:

- *Active mode*: En este modo, el dispositivo Bluetooth participa activamente en el canal.
- *Sniff mode*: En este modo, el tiempo de actividad durante el cual el dispositivo esclavo escucha se reduce. Esto significa que el maestro sólo

puede iniciar una transmisión en unos slots de tiempo determinados.

- *Hold mode*: En el estado conectado, el enlace con el esclavo puede ponerse en espera. Durante este modo, el esclavo puede hacer otras cosas, como escanear en busca de otros dispositivos, atender otra piconet, etc.
- *Park mode*: En este estado, el esclavo no necesita participar en la piconet, pero aún quiere seguir sincronizado con el canal. Deja de ser miembro de la piconet. Esto es útil por si hay más de siete dispositivos que necesitan participar ocasionalmente en la piconet.

### APIs Java para Bluetooth

Mientras que el hardware Bluetooth había avanzado mucho, hasta hace relativamente poco no había manera de desarrollar aplicaciones java Bluetooth , hasta que apareció JSR 82 [15]<sup>10</sup>, que estandarizó la forma de desarrollar aplicaciones Bluetooth usando Java. Ésta esconde la complejidad del protocolo Bluetooth detrás de unos APIs que permiten centrarse en el desarrollo en vez de los detalles de bajo nivel del Bluetooth.

Estos APIs para Bluetooth están orientados para dispositivos que cumplan las siguientes características:

- Al menos 512K de memoria libre (ROM y RAM) (las aplicaciones necesitan memoria adicional).
- Conectividad a la red inalámbrica Bluetooth.
- Que tengan una implementación del J2ME CLDC.

### JSR 82

El objetivo de ésta especificación era definir un API estándar abierto, no propietario que pudiera ser usado en todos los dispositivos que implementen J2ME. Por consiguiente fue diseñado usando los APIs J2ME y el entorno de trabajo CLDC/MIDP.

Los APIs JSR 82 son muy flexibles, ya que permiten trabajar tanto con aplicaciones nativas Bluetooth como con aplicaciones Java Bluetooth. El API intenta ofrecer las siguientes capacidades:

- Registro de servicios.
- Descubrimiento de dispositivos y servicios.
- Establecer conexiones RFCOMM, L2CAP y OBEX entre dispositivos.

---

10 Ref. Bibliografía: JSR 82

- Usar dichas conexiones para mandar y recibir datos (las comunicaciones de voz no están soportadas).
- Manejar y controlar las conexiones de comunicación.
- Ofrecer seguridad a dichas actividades.

Los APIs Java para Bluetooth definen dos paquetes que dependen del paquete CLDC `javax.microedition.io`:

- `javax.bluetooth`
- `javax.obex`

### **Programación de aplicaciones Bluetooth**

La anatomía de una aplicación Bluetooth está dividida en cuatro partes:

- Inicialización de la pila.
- Descubrimiento de dispositivos y servicios.
- Manejo del dispositivo.
- Comunicación.

### **Inicialización**

#### **BCC (Bluetooth Control Center)**

Los dispositivos Bluetooth que implementen este API pueden permitir que múltiples aplicaciones se estén ejecutando concurrentemente. El BCC previene que una aplicación pueda perjudicar a otra. El BCC es un conjunto de capacidades que permiten al usuario o al OEM<sup>11</sup> resolver peticiones conflictivas de aplicaciones definiendo unos valores específicos para ciertos parámetros de la pila Bluetooth.

El BCC puede ser una aplicación nativa, una aplicación en un API separado, o sencillamente un grupo de parámetros fijados por el proveedor que no pueden ser cambiados por el usuario. Hay que destacar, que el BCC no es una clase o un interfaz definido en esta especificación, pero es una parte importante de su arquitectura de seguridad.

#### **Inicialización de la Pila**

La pila Bluetooth es la responsable de controlar el dispositivo Bluetooth, por lo que es necesario inicializarla antes de hacer cualquier otra cosa. El proceso de inicialización consiste en un número de pasos cuyo propósito es dejar el dispositivo listo para la comunicación inalámbrica.

Desafortunadamente, la especificación deja la implementación del BCC a

---

11 Original Equipment Manufacturer

los vendedores, y cada vendedor maneja la inicialización de una manera diferente. En un dispositivo puede haber una aplicación con un interfaz GUI, y en otra puede ser una serie de configuraciones que no pueden ser cambiados por el usuario. Un ejemplo sería<sup>12</sup>:

```
...
// Configuramos el puerto
BCC.setPortNumber("COM1");
// Configuramos la velocidad de la conexión
BCC.setBaudRate(50000);
//Configuramos el modo conectable
BCC.setConnectable(true);
//Configuramos el modo discovery a LIAC13
BCC.setDiscoverable(DiscoveryAgent.LIAC);
...
```

## **Discovery**

Dado que los dispositivos inalámbricos son móviles, necesitan un mecanismo que permita encontrar, conectar, y obtener información sobre las características de dichos dispositivos.

En este apartado, vamos a tratar como el API de Bluetooth permite realizar todas estas tareas.

### **Descubrir dispositivos (Device discovery)**

Cualquier aplicación puede obtener una lista de dispositivos a los que es capaz de encontrar, usando, o bien ***startInquiry()*** (no bloqueante) o ***retrieveDevices()*** (bloqueante). ***startInquiry()*** requiere que la aplicación tenga especificado un *listener*, el cual es notificado cuando un nuevo dispositivo es encontrado después de haber lanzado un proceso de búsqueda. Por otra parte, si la aplicación no quiere esperar a descubrir dispositivos (o a ser descubierta por otro dispositivo) para comenzar, puede utilizar ***retrieveDevices()***, que devuelve una lista de dispositivos encontrados en una búsqueda previa o bien unos que ya conozca por defecto.

#### **Clases del Device Discovery**

**interface javax.bluetooth.DiscoveryListener**

---

12 Los APIs invocados aquí no son parte del JSR 82

13 Limited Inquiry Access Code

Este interfaz permite a una aplicación especificar un evento en el *listener* que reaccione ante eventos de búsqueda. También se usa para encontrar dispositivos. El método **deviceDiscovered()** se llama cada vez que se encuentra un dispositivo en un proceso de búsqueda. Cuando el proceso de búsqueda se ha completado o cancelado, se llama al método **inquiryCompleted()**. Este método recibe un argumento, que puede ser INQUIRY\_COMPLETED, INQUIRY\_ERROR o INQUIRY\_TERMINATED, dependiendo de cada caso.

#### **interface javax.bluetoothDiscoveryAgent**

Esta interfaz provee métodos para descubrir dispositivos y servicios. Para descubrir dispositivos, esta clase provee del método **startInquiry()** para poner al dispositivo en modo de búsqueda, y el método **retrieveDevices()** para obtener la información de dispositivos previamente encontrados. Además provee del método **cancelInquiry()** que permite cancelar una operación de búsqueda.

#### **Descubrir servicios (Service Discovery)**

En este apartado vamos a ver la parte del API que usa el cliente para descubrir servicios disponibles en los dispositivos servidores encontrados. La clase DiscoveryAgent provee de métodos para buscar servicios en un dispositivo servidor Bluetooth e iniciar transacciones entre el dispositivo y el servicio. Este API no da soporte para buscar servicios que estén ubicados en el propio dispositivo.

Para descubrir los servicios disponibles en un dispositivo servidor, el cliente primero debe recuperar un objeto que encapsule funcionalidad SDAP<sup>14</sup> (SDP<sup>15</sup> + GAP<sup>16</sup>, se verá más adelante).

Este objeto es del tipo DiscoveryAgent, cuyo pseudocódigo viene dado por:

```
DiscoveryAgent da =  
LocalDevice.getLocalDevice().getDiscoveryAgent();
```

#### **Clases del Service Discovery**

##### **class javax.bluetooth.UUID**

Esta clase encapsula enteros sin signo que pueden ser de 16, 32 ó 128

---

14 SDAP=Service Discovery Application Profile

15 SDP=Service Discovery Profile

16 GAP=Generic Access Profile

bits de longitud.

Estos enteros se usan como un identificador universal cuyo valor representa un atributo del servicio. Sólo los atributos de un servicio representados con UUID están representados en la Bluetooth SDP.

```
class javax.bluetooth.DataElement
```

Esta clase contiene varios tipos de datos que un atributo de servicio Bluetooth puede usar. Algunos de estos son:

- String
- boolean
- UUID
- Enteros con signo y sin signo, de uno, dos, cuatro o seis bytes de longitud
- secuencias de cualquiera de los tipos anteriores.

Esta clase además presenta una interfaz que permite construir y recuperar valores de un atributo de servicio.

```
class javax.bluetooth.DiscoveryAgent
```

Esta clase provee métodos para descubrir servicios y dispositivos.

```
interface javax.bluetooth.ServiceRecord
```

Este interfaz define el Service Record de Bluetooth, que contiene los pares (atributo ID, valor). El atributo ID es un entero sin signo de 16 bits, y valor es de tipo DataElement. Además, este interfaz tiene un método populateRecord() para recuperar los atributos de servicio deseados (pasando como parámetro al método el ID del atributo deseado).

```
interface javax.bluetooth.DiscoveryListener
```

Este interfaz permite a una aplicación especificar un *listener* que responda a un evento del tipo Service Discovery o Device Discovery. Cuando un nuevo servicio es descubierto, se llama al método servicesDiscovered(), y cuando la transacción ha sido completada o cancelada se llama a

serviceSearchCompleted()).

### **Registro del servicio(Service Registration)**

Las responsabilidades de una aplicación servidora de Bluetooth son:

1. Crear un Service Record que describa el servicio ofrecido por la aplicación.
2. Añadir el Service Record al SDDB del servidor para avisar a los clientes potenciales de este servicio.
3. Registrar las medidas de seguridad Bluetooth asociadas a un servicio.
4. Aceptar conexiones de clientes que requieran el servicio ofrecido por la aplicación.
5. Actualizar el Service Record en el SDDB del servidor si las características del servicio cambian.
6. Quitar o deshabilitar el Service Record en el SDDB del servidor cuando el servicio no está disponible.

A las tareas 1,2,5 y 6 se las denominan registro del servicio (Service Registration), que comprenden unas tareas relacionadas con advertir al cliente de los servicios disponibles.

### **Responsabilidades del Registro de Servicio**

En la figura 9.5.3 vemos, que cuando la aplicación llama a **Connector.open()** con un String conexión URL, la implementación crea un **ServiceRecord**. El correspondiente registro del servicio es añadido a la SDDB por la implementación cuando la aplicación servidora llama a **acceptAndOpen()**. La aplicación servidora puede acceder a dicho **ServiceRecord** llamando a **getRecord()** y hacer las modificaciones pertinentes. Las modificaciones se hacen también en el **ServiceRecord** de la SDDB cuando la aplicación llama a **updateRecord()**. Finalmente el **ServiceRecord** es eliminado de la SDDB cuando la aplicación servidora manda un **close** al **notifier**.

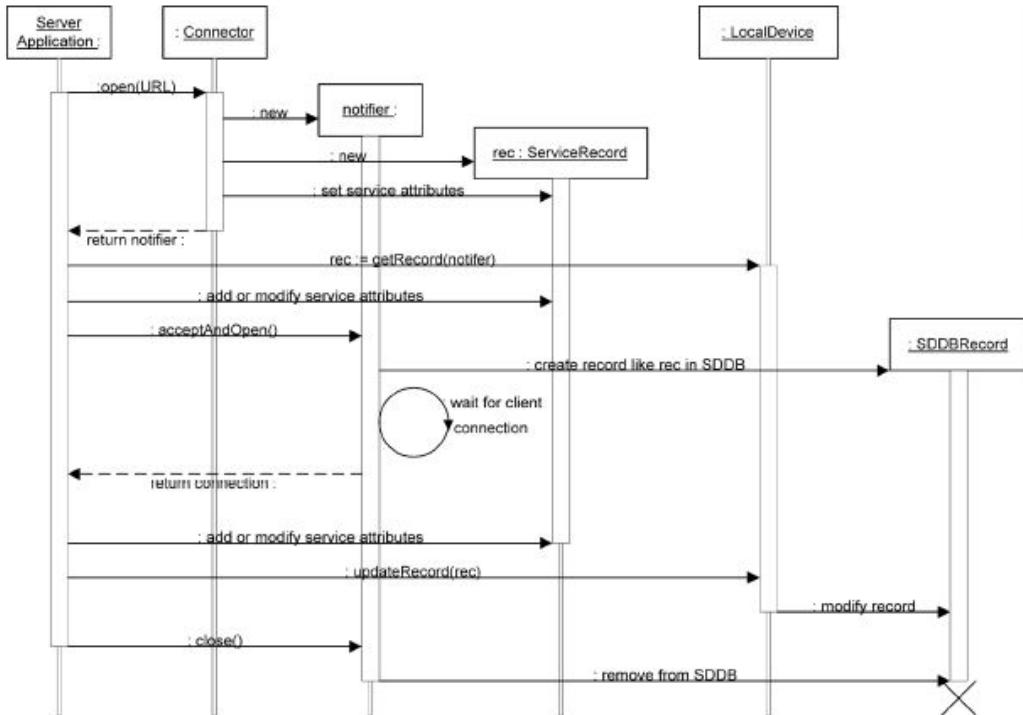


Ilustración 9.5.3: Colaboración entre la implementación y la aplicación servidora para el registro del servicio

### Modos conectable y no conectable

- *Modo conectable*: un dispositivo en este modo escucha periódicamente intentos de iniciar una conexión de un dispositivo remoto.
- *Modo no-conectable*: un dispositivo en este modo no escucha intentos de iniciar una conexión de un dispositivo remoto.

Para el correcto funcionamiento de una aplicación servidora, es necesario que el dispositivo servidor esté en *modo conectable*. Es por esto, que en la implementación de `acceptAndOpen()`, ésta debe asegurarse que el dispositivo local esté en *modo conectable* (dado que depende del usuario el tener o no el dispositivo en un modo u otro). La implementación hace una petición al BCC para hacer al dispositivo local conectable, si ésta no es posible, se lanzará una excepción `BluetoothStateException`.

Cuando todos los servicios en la SDDB han sido eliminados o deshabilitados, la implementación puede decidir opcionalmente pedir al dispositivo servidor que pase al *modo no-conectable*. Aunque un dispositivo esté en el *modo no-conectable* (no responde a intentos de conexión), puede iniciar un intento de conexión. Por esto, un dispositivo en *modo no-conectable* puede ser un cliente, pero no un servidor. Por lo tanto la implementación no necesita pedir al dispositivo que se ponga en *modo conectable* si no tiene ningún

ServiceRecord en su SDDB.

### Clases del Service Registration

```
interfaz javax.bluetooth.ServiceRecord
```

Un ServiceRecord describe un servicio Bluetooth a los clientes. Está compuesto de unos “cuantos” atributos de servicio. El SDP del servidor mantiene una base de datos de los *ServiceRecords*. Un servicio *run-before-connect* (la aplicación se ejecuta sin esperar a que haya una conexión establecida) añade su *ServiceRecord* a la SDDB llamando a **acceptAndOpen()**. El *ServiceRecord* provee suficiente información a un cliente SDP para poder conectarse al servicio Bluetooth del dispositivo servidor.

La aplicación servidora puede usar el método **setDeviceClasses()** para activar alguno de los bits de la clase servidora para reflejar la incorporación de un nuevo servicio.

```
class javax.bluetooth.LocalDevice
```

Esta clase provee un método **getRecord()** que la aplicación servidora puede usar para obtener el *ServiceRecord*. Una vez modificado, puede ser puesto en la SDDB llamando al método **notifier.acceptAndOpen()** o **updateRecord()** de *LocalDevice*.

```
class javax.bluetooth.ServiceRegistrationException extends  
java.io.IOException
```

La excepción *ServiceRegistrationException* se lanza cuando se intenta añadir o modificar un *ServiceRecord* en la SDDB y hay algún error. Estos errores pueden ocurrir:

- Durante la ejecución de **Connector.open()**.
- Cuando un servicio *run-before-connect* invoca a **acceptAndOpen()** y la implementación intenta añadir el *ServiceRecord* asociado al *notifier* en la SDDB.
- Después de la creación del *ServiceRecord*, cuando la aplicación servidora intenta modificar el *ServiceRecord* en la SDDB usando **updateRecord()**.

## **Comunicación**

Para usar un servicio en un dispositivo Bluetooth remoto, el dispositivo local debe comunicarse usando el mismo protocolo que el servicio remoto. Los APIs permiten usar RFCOMM, L2CAP u OBEX como protocolo de nivel superior.

El Generic Connection Framework (GCF) del CLDC provee la conexión base para la implementación de protocolos de comunicación. CLDC provee de los siguientes métodos para abrir una conexión:

- *Connection Connector.open(String name);*
- *Connection Connector.open(String name, int mode);*
- *Connection Connector.open(String name, int mode, boolean timeouts);*

La implementación debe soportar abrir una conexión con una conexión URL servidora o con una conexión URL cliente, con el modo por defecto READ\_WRITE.

## **Perfil del puerto serie (SPP)**

El protocolo RFCOMM provee múltiples emulaciones de los puertos serie RS-232 entre dos dispositivos Bluetooth. Las direcciones Bluetooth de los dos puntos terminales definen una sesión RFCOMM. Una sesión puede tener más de una conexión, el número de conexiones dependerá de la implementación. Un dispositivo podrá tener más de una sesión RFCOMM en tanto que esté conectado a más de un dispositivo.

### **Un vistazo al API**

Una aplicación que ofrezca un servicio basado en el perfil de puerto serie (SPP) es un servidor SPP. Una aplicación que inicie una conexión a un servicio SPP es un cliente SPP. Cliente y servidor residen en los extremos de una sesión RFCOMM. El servidor SPP registra su servicio en el SDDB, y como parte del proceso de registro, se añade un identificador de canal (channel identifier) al ServiceRecord por la implementación. En este apartado vamos a ver las capacidades que una implementación SPP tiene que tener para los interfaces StreamConnection y StreamConnectionNotifier del CLDC.

### **Conexiones URL de un cliente y un servidor SPP**

A continuación vamos a mostrar algunos de los argumentos (ABNF) necesarios para la conexión URL entre clientes y servidores.

```
srvString = protocol colon slashes srvHost 0*5(srvParams)
cliString = protocol colon slashes cliHost 0*3(cliParams)
protocol = btspp
btspp = %d98.116.115.112.112 // define el literal btspp
```

```
cliHost = address colon channel
srvHost = "localhost" colon uuid
channel = %d1 -30
uidd = 1*32(HEXDIG)
colon = ":" 
slashes = "//"
bool = "true" / "false"
address = 12*12(HEXDIG)
text = 1*( ALPHA/ DIGIT / SP / "-" / "_" )
name = ";name=" text
master = ";master=" bool
encrypt = ";encrypt=" bool
authorize = ";authorize=" bool
authenticate = ";authenticate=" bool
cliParams = master / encrypt / authenticate
servParams = name / master / encrypt / authorize / authenticate
```

SP se usa para espacios, ALPHA para letras alfabéticas mayúsculas y minúsculas, DIGIT se usa para números de cero a nueve y HEXDIG para números hexadecimales (0-9, a-f,A-F).

### Registro del servicio del puerto serie

Un SPP debe inicializar los servicios que ofrece y registrarlos en el SDDDB. Un servicio

de puerto serie viene representado por un par de objetos emparentados:

1. Un objeto que implementa el interfaz **javax.microedition.io.StreamConnectorNotifier**. Este objeto escucha conexiones clientes que demanden este servicio.
2. Un objeto que implemente el interfaz **javax.bluetooth.ServiceRecord**. Este objeto describe el servicio y como puede ser accedido por dispositivos remotos.

Para crear estos objetos la aplicación servidora usa el método **Connector.open()** con un argumento de conexión URL, del siguiente modo:

```
StreamConnectionNotifier service =
(StreamConnectionNotifier)Connector.open
```

```
("btspp://localhost:102030405060708090A1B1C1D1D1E100;name=SPPEx"
);
```

Invocando **Connector.open()** con un argumento conexión URL, éste devuelve un StreamConnectionNotifier que representa el servicio SPP. La implementación de **Connector.open()** además crea un nuevo registro de servicio (*service record*) que representa el servicio SPP. Una implementación de un SPP debe realizar los siguientes pasos cuando crea el registro de servicio.

1. Crear un identificador de un canal servidor RFCOMM, *chanN* y asignarlo.
2. *chanN* es añadido al *ProtocolDescriptorList* en el registro de servicio.
3. El UUID (102030...) usado en el *connection string* para describir el tipo de servicio ofrecido es añadido al *ServiceClassIDList*.
4. El atributo *ServiceName* es añadido al registro de servicio con el valor "SPPEx".

En el caso de un servicio *run-before-connect*, el registro de servicio es añadido a la SDDB la primera vez que la aplicación servidora llama a **acceptAndOpen()** en el StreamConnectionNotifier asociado. El registro de servicio se hace visible a potenciales clientes SPP cuando es añadida a la SDDB.

## Establecimiento de la conexión

### Establecimiento de la conexión del servidor

Un servidor SPP crea un objeto *StreamConnectionNotifier* del siguiente modo:

- Usando el apropiado string para un servidor SPP como argumento de **Connector.open()**
- Haciendo un casting del resultado de **Connector.open()** al interfaz *StreamConnectionNotifier*.

```
StreamConnectionNotifier service =
(StreamConnectionNotifier)Connector.open
("btspp://localhost:102030405060708090A1B1C1D1D1E100;name=SPPEx"
);
StreamConnection con = (StreamConnection)
service.acceptAndOpen();
```

Un servicio SPP puede aceptar múltiples conexiones de diferentes clientes llamando a **acceptAndOpen()** repetidamente. Cada cliente accede al mismo registro de servicio y se conecta al servicio usando el mismo canal servidor RFCOMM. Si el sistema Bluetooth no soporta múltiples conexiones, el

**acceptAndOpen()** lanzará un *BluetoothStateException*.

El método **close()** en el objeto *StreamConnection* representa que se ha usado una conexión SPP servidora para cerrar la conexión.

Cuando un servicio run-before-connect manda un mensaje **close()** al *StreamConnectionNotifier*, el registro de servicio asociado a ese notifier se vuelve inaccesible a los clientes que estén usando el servicio discovery. La implementación debe eliminar el registro de servicio de la SDDB. El mensaje **close()** además hace que la implementación desactive los bits de clase que fueron activados por **setServiceClasses()** (excepto si otro notifier activó esos bits y aún está activo).

#### Establecimiento de la conexión del cliente:

Antes de que un cliente SPP pueda establecer una conexión con un servicio SPP, éste debe previamente haber descubierto el servicio mediante el servicio discovery. Una conexión URL del cliente incluye la dirección Bluetooth del dispositivo servidor y el identificador de canal del servidor. El método **getConnectionURL()** en el interfaz *ServiceRecord* se usa para obtener la conexión URL del cliente.

Invocando el método **Connector.open()** con una conexión URL del cliente, devuelve un objeto *StreamConnection* que representa la conexión SPP del lado del cliente.

```
StreamConnection con =  
(StreamConnection)  
Connector.open("btspp://dirección:identificador_canal");
```

Los registros de servicio consisten en una colección de pares (*attrID*, *attrValue*). Cada par describe un atributo del servicio. La aplicación servidora puede opcionalmente añadir otros atributos de servicio al *ServiceRecord*. Es posible incluso añadir atributos definidos por el usuario.

Con el método **updateServiceAvailability()**, la aplicación servidora puede obtener el *ServiceRecord* que fue creado por:

```
ServiceRecord record = localDev.getRecord(notifier);
```

La aplicación servidora modificará el atributo *ServiceAvailability* basándose en el número de conexiones de cliente actuales. Las modificaciones que la aplicación servidora hace al *ServiceRecord* no se reflejan inmediatamente en la SDDB, si no que para ello se usará:

```
localDev.updateRecord(record);
```

### Restricciones en la modificación de los registros de servicio

Las aplicaciones sólo pueden acceder a sus propios notifiers, no es posible para una aplicación modificar el *ServiceRecord* de otra aplicación en el SDDB servidor .

El *ProtocolDescriptorList* le dice a una aplicación cliente como conectarse al servicio. *Protocol0* *Protocol1* representan la pila (L2CAP, RFCOMM) que normalmente se usa para conectar a un servicio de puerto serie. Esos atributos son fijos para asegurarse que esa pila esté siempre en el *ProtocolDescriptorList*.

**ProtocolSpecificParameter0** es el identificador del canal servidor. Es un atributo fijo para asegurarse que la implementación de RFCOMM puede manejar la asignación de los valores del canal servidor.

### L2CAP

En este apartado vamos a ver el protocolo de adaptación y control lógico del enlace (Logical Link Control and Adaptation Protocol). L2CAP soporta dos tipos de conexiones, orientadas a conexión (bidireccionales) y no orientadas a conexión (unidireccionales). Todas las conexiones hechas usando la primitiva de servicio connect de la capa L2CAP son orientadas a conexión, este API no soporta comunicaciones en grupo, y por lo tanto no soporta canales no orientados a conexión.

#### Un vistazo al API

Este API soporta sólo canales L2CAP orientados a conexión. El *L2CAPConnectionNotifier* le indica a un servidor L2CAP cuando un cliente inicia una conexión. Una vez que la conexión está establecida, se devuelve un objeto *L2CAPConnection*. El interfaz *L2CAPConnection* y *L2CAPConnectionNotifier* extienden el interfaz *Connection*. Este interfaz puede ser usado para enviar o recibir datos de un dispositivo remoto usando el protocolo L2CAP.

#### Configuración del canal

Los canales orientados a conexión necesitan ser configurados una vez que se ha establecido la conexión. Los parámetros de configuración del canal que se negocian entre los dispositivos Bluetooth son:

- Unidad máxima de transferencia (MTU): Es el tamaño del *payload* que el que envía la petición es capaz de atender. El valor por defecto es de 672 bytes (DEFAULT\_MTU).
- Tiempo de descarte: Es la cantidad de tiempo durante el cual el administrador del canal intenta transmitir satisfactoriamente el paquete antes de descartarlo. El valor 0xFFFF (valor por defecto) indica que el paquete será retransmitido hasta que llegue un asentimiento o hasta que el enlace ACL termine.

- Calidad del servicio (QoS): esta opción describe el flujo de tráfico. Este parámetro no está soportado en el API.

### Unidad máxima de transferencia

La implementación es responsable de configurar el canal con la MTU pedida o usando la que tiene por defecto, antes de cualquier operación de lectura o escritura. El parámetro *ReceiveMTU* es el número máximo de bytes que el dispositivo local puede recibir en el payload. El parámetro *TransmitMTU* es el número máximo de bytes que el dispositivo local puede mandar al dispositivo remoto en el payload.

Si  $MTU_A < TransmitMTU_B$  o  $TransmitMTU_A < ReceiveMTU_B$  la conexión falla.

Vamos a ver como la implementación configura la MTU de acuerdo a la petición hecha.

Hay diferentes posibilidades:

1. La aplicación especifica el *ReceiveMTU* y el *TransmitMTU*. En este caso la implementación advierte del valor del *ReceiveMTU* al dispositivo remoto. Si el dispositivo remoto responde con una respuesta de configuración negativa la conexión falla, si es positivo, la implementación espera a la petición de configuración del dispositivo remoto. Cuando la recibe compara el valor que le ha venido de *ReceiveMTU* con el *TransmitMTU* que ha mandado, si *TransmitMTU* es menor o igual, la conexión es posible, sino la conexión falla.
2. La aplicación especifica *ReceiveMTU* pero no *TransmitMTU*. La aplicación puede usar el método *getTransmitMTU()* de la clase *L2CAPConnection* para obtener la MTU de salida para evitar enviar muchos datos.
3. La aplicación especifica *TransmitMTU*. En este caso la aplicación avisa al dispositivo remoto que el *ReceiveMTU* va a ser el *DEFAULT\_MTU* (672bytes).
4. Si la aplicación no especifica ninguno de los dos parámetros, estamos en un caso similar al caso 2 y 3.

### Interfaz de conexión L2CaP

#### Conexiones URL de un cliente y servidor L2CAP

```
srvString = protocol colon slashes srvHost 0*7(srvParams)
cliString = protocol colon slashes cliHost 0*5(cliParams)
protocol = bt12cap
```

## 9.5 .Anexo V: Bluetooth:JSR 82

```
btspp = %d98.116.108.50.99.97.112 // define el literal l2cap
cliHost = address colon psm
srvHost = "localhost" colon uuid
psm = 4*4 (HEXDIG)
uidd = 1*32 (HEXDIG)
colon = ":" 
slashes = "//"
bool = "true" / "false"
address = 12*12 (HEXDIG)
text = 1*( ALPHA/ DIGIT / SP / "-" / "_" )
name = ";name=" text
master = ";master=" bool
encrypt = ";encrypt=" bool
authorize = ";authorize=" bool
authenticate = ";authenticate=" bool
receiveMTU = ";receiveMTU=" 1*(DIGIT)
transmitMTU = "transmitMTU=" 1*(DIGIT)
cliParams = master / encrypt / authenticate / receiveMTU /
transmitMTU
servParams = name / master / encrypt / authorize / authenticate/
receiveMTU /
transmitMTU
```

SP se usa para espacios, ALPHA para letras alfabéticas mayúsculas y minúsculas, DIGIT se usa para números de cero a nueve y HEXDIG para números hexadecimales (0-9, a-f,A-F).

El string psm es un descriptor de la conexión, y representa el *Protocol Service Multiplexor*. De este modo las aplicaciones servidoras L2CAP se pueden diferenciar entre sí. Este valor está comprendido entre (0x1001..0xFFFF), siendo el bit menos significativo impar y todos los demás pares.

El pseudocódigo para abrir una conexión cliente L2CAP es:

```
try{
    L2CAPConnection client = (L2CAPConnection)
    Connctor.open("bt12cap://.....;ReceiveMTU=xxx;TransmitMTU=yyy"
);
} catch(...)
```

El pseudocódigo para abrir una conexión servidor L2CAP es:

```
try{
    L2CAPConnectionNotifier server = (L2CAPConnectionNotifier)
    Connctor.open("bt12cap://localhost:...;name=L2CAPEx");
    L2CAPConnection con = (L2CAPConnection) server.acceptAndOpen();
} catch (...)
```

En el caso de que haya habido algún fallo durante la conexión y se haya lanzado alguna excepción, se puede obtener el motivo del fallo usando el método `getStatus()`.

### Registro de servicio L2CAP

Cuando una aplicación servidora L2CAP llama a `Connector.open()`, se crea un registro de servicio de manera similar a como se hacía en los servicios de puerto serie.

### Clases de conexión L2CAP

```
interface javax.bluetooth.L2CAPConnection extends
javax.microedition.io.Connection
```

Este interfaz representa las conexiones L2CAP. Contiene métodos para obtener las MTUs usadas en la conexión y métodos para enviar y recibir datos.

```
interface javax.bluetooth.L2CAPConnectionNotifier extends
javax.microedition.io.Connection
```

El único método de este interfaz es `acceptAndOpen()`, que es usado por los servidores L2CAP para escuchar conexiones de clientes.

```
Class javax.bluetooth.BluetoothConnectionException extends
java.io.IOException
```

Esta excepción se lanza cuando una conexión Bluetooth (RFCOMM o L2CAP) no puede ser establecida satisfactoriamente. El método `getStatus()` de esta clase indicará la razón del fallo de la conexión.

### Protocolo de intercambio de objetos (OBEX)

En este apartado vamos a tratar el protocolo de intercambio de objetos. En realidad, este tema aparentemente no se debería tratar, ya que dicho protocolo OBEX no está definido en el API Bluetooth, si no que tiene su propio API.

Este es un protocolo diseñado por el IrDA (Infrared Data Association) para intercambiar objetos entre clientes y servidores, mediante el establecimiento de una sesión OBEX. En vez de incluir esta funcionalidad en el API Bluetooth, se ha optado por extender el API OBEX y dar soporte a Bluetooth. Nosotros vamos a centrarnos únicamente en el soporte Bluetooth.

#### Un vistazo al API

OBEX implementa la transferencia de objetos estableciendo una sesión OBEX, mediante una petición CONNECT. Ésta termina mediante una petición DISCONNECT. Entre estas dos peticiones, el cliente puede traer objetos del servidor mediante GET, o enviarlos mediante PUT. Los objetos pueden ser archivos, vCards, arraíles de bytes,etc. El cliente puede además cambiar el archivo o carpeta en uso mediante la petición SETPATH. Otras operaciones permitidas son: ABORT, CREATE-EMPTY, PUT-DELETE.

OBEX, como HTTP, tiene métodos que le permiten pasar información adicional entre el cliente y el servidor mediante el uso de cabeceras.

Nombre de la cabecera	Cómo manipular la cabecera en la API
Count	HeaderSet.getHeader(), HeaderSet.setHeader()
Name	HeaderSet.getHeader(), HeaderSet.setHeader()
Type	HeaderSet.getHeader(), HeaderSet.setHeader()
Length	HeaderSet.getHeader(), HeaderSet.setHeader()
Time	HeaderSet.getHeader(), HeaderSet.setHeader()
Description	HeaderSet.getHeader(), HeaderSet.setHeader()
Target	HeaderSet.getHeader(), HeaderSet.setHeader()
HTTP	HeaderSet.getHeader(), HeaderSet.setHeader()
Body	Operation.openInputStream(), Operation.openDataInputStream(), Operation.openOutputStream(), Operation.openDataOutputStream()
End of Body	Operation.openInputStream(), Operation.openDataInputStream(), Operation.openOutputStream(),

	Operation.openDataOutputStream()
Who	HeaderSet.getHeader(), HeaderSet.setHeader()
Connection ID	ClientSession.setConnectionID(), ClientSession.getConnectionID(), ServerRequestHandler.setConnectionID(), ServerRequestHandler.getConnectionID()
Application parameters	HeaderSet.getHeader(), HeaderSet.setHeader()
Authentication Challenge	HeaderSet.createAuthenticationChallenge(), Authenticator.getPasswordAuthentication()
Authentication Response	Authenticator.getPasswordAuthentication(), Authenticator.validatePassword()
Object Class	HeaderSet.getHeader(), HeaderSet.setHeader()
User defined	HeaderSet.getHeader(), HeaderSet.setHeader()

Cuando se hace la llamada a **Connector.open()** se pueden dar las siguientes excepciones:

- *ConnectionNotFoundException* : se lanza cuando el entorno usado no es válido o cuando el tipo de protocolo no existe.
- *IllegalArgumentException*: se lanza cuando los parámetros del *connection string* no se reconocen.
- *IOException*: se lanza cuando no se puede conectar con el objetivo.

### Conexión del cliente

Para crear una conexión OBEX, el cliente le debe parar el string apropiado al **Connector.open()**, y este devolverá un objeto *javax.obex.ClientSession*. Para establecer la conexión OBEX el cliente crea un objeto *javax.obex.HeaderSet* usando el método **createHeaderSet()** del interfaz *ClientSession*. Finalmente el cliente facilita el objeto *HeaderSet* al método **connect()** de la interfaz *ClientSession*.

Para determinar si la petición ha tenido éxito o no, se usa el método **getResponseCode()** del interfaz *HeaderSet*, que devuelve un código de respuesta mandado por el servidor, que viene definido en la clase *javax.obex.ResponseCodes*.

Para la petición DISCONNECT, se procede del mismo modo, excepto que en vez de usar el método **connect()** se usa el método **disconnect()**. Para completar una operación SETPATH, el cliente llama al método **setPath()** en el objeto *ClientSession*. Para especificar el nombre del directorio destino, pone el

nombre llamando al método **setHeader()** del *HeaderSet*. Si la cabecera es muy larga se lanzará una excepción *java.io.IOException*.

Para completar una operación GET o PUT, el cliente crea un objeto *javax.obex.HeaderSet* con el método **createHeaderSet()**. Después de establecer los valores de cabecera, el cliente llama a los métodos **put()** o **get()** del objeto *javax.obex.ClientSession*.

Para abortar un PUT o un GET, el cliente llama al método **abort()** del objeto *javax.obex.Operation*. El método **abort()** llama además al método **close()** del objeto *Operation*.

### Conexión del servidor

Para crear una conexión servidora, el servidor invoca a *Connector.open()*, que le devuelve un objeto *javax.obex.SessionNotifier*. Este objeto espera a que el cliente cree una capa de transporte llamando a **acceptAndOpen()**.

El servidor debe crear una nueva clase que extienda la clase *javax.obex.serverRequestHandler*. El servidor deberá implementar aquellos métodos de OBEX a los que da soporte. Las aplicaciones servidoras no deben llamar al método **abort()**, ya que si no el argumento *javax.obex.Operation*, que es parte de los métodos **onGet()** y **onPut()**, lanzará una *java.io.IOException*.

### Autenticación

Para autenticar a un cliente o servidor OBEX, tanto el cliente como el servidor deben compartir un secreto o un password. Éste nunca es intercambiado durante el proceso de autenticación. Si el cliente quiere autenticar al servidor, éste le envía una cabecera con un reto (de 16 bytes). Con ésta, el servidor determina el password o el secreto. Entonces el servidor combina el password con el reto aplicando el algoritmo MD5. El resultado (llamado response digest) se le devuelve en la cabecera de autenticación. Entonces el cliente compara el reto que mandó combinado con el password y el algoritmo MD5 con el que ha recibido; si son el mismo, el servidor se ha autenticado.

El proceso de autenticación comienza con la llamada al método *createAuthenticationChallenge()*, el cual le indica a la implementación que incluya un reto de autenticación en la siguiente petición o respuesta.

Para facilitar el proceso de autenticación, el interfaz *Authenticator* provee de métodos que pueden ser implementados por la aplicación para responder retos. El método *onAuthenticationChallenge()* es invocado cuando una cabecera con un reto de autenticación es recibida. Cuando se recibe la respuesta de una autenticación, se llama al método *onAuthenticationResponse()* con el nombre de usuario (si está incluido en la cabecera de respuesta).

Si el proceso de autenticación falla, cuando el cliente invoque *connect()*, *setPath()*, *delete()*, *get()*, *put()* o *disconnect()* se producirá un *IOException* lanzado por el método. Si los valores no son iguales en el servidor OBEX, se

llamará al método `onAuthenticationFailure()` en el `ServerRequestHandler` del servidor.

### Clases OBEX

```
interface javax.obex.ClientSession extends  
javax.microedition.io.Connection
```

Este interfaz representa los objetos de conexión del lado del cliente. Provee de los métodos para las peticiones `CONNECT`, `DISCONNECT`, `SETPATH`, `PUTDELETE`, `CREATEEMPTY`, `PUT` y `GET`.

```
interface javax.obex.HeaderSet
```

Este interfaz define las cabeceras OBEX que deben ser implementadas en una operación. Provee de los métodos `get()` y `set()`. Los clientes pueden crear una objeto `HeaderSet` llamado al método `createHeader()` del objeto `javax.obex.ClientSession`.

```
class javax.obex.ResponseCodes
```

Esta clase implementa los códigos de respuesta válidos en un servidor OBEX

```
class javax.obex.ServerRequestHandler
```

Esta clase define el esquema de como el cliente OBEX maneja las peticiones. La aplicación que extiende esta clase sólo necesita reescribir aquellos métodos que soporta.

```
interface javax.obex.SessionNotifier extends  
javax.microedition.io.Connection
```

Este interfaz define el objeto de sesión notifier que es devuelto siguiendo a una llamada a `Connector.open()`. Provee además de métodos para esperar a un cliente para que establezca una conexión.

```
interface javax.obex.Operation extends  
javax.microedition.io.ContentConnection
```

Esta interfaz define un objeto de operación que es usado para las operaciones GET y PUT. Además provee del método ABORT.

```
interface Authenticator
```

Este interfaz maneja la autenticación y las cabeceras de respuesta de autenticación

```
class PasswordAuthentication
```

Esta clase encapsula el nombre de usuario y el password usados en la autenticación.

# Bibliografía

- [1] *McConnell, S. Desarrollo y Gestión de Proyectos Informáticos.*  
*McGraw-Hill, 2000.*
- [2] *Pressman, R. Ingeniería del Software, 5<sup>a</sup> edición.*  
*McGraw Hill, 2001.*
- [3] *Garví E., Holgado J. A., Rodríguez M.L. Ingeniería del Software II. Material de Clase.* 2008. Departamento De Lenguajes y Sistemas Informáticos de la Universidad de Granada.
- [4] *Booch, G., Rumbaugh, J., Jacobson, El lenguaje Unificado de modelado. Guía del usuario.*  
*I. Pearson Educación, 2006.*
- [5] *Jesús Gonzalez P., Miguel Angel López M., Pedro Antonio Castillo V., Pablo García S, Curso: Web 2.0:Arquitectura Orientada a Servicios en Java.*  
*Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Granada.*  
*2<sup>a</sup> Edición, 2010.*
- [6] *José Luis Bernier V., Mario J. Barchéin M., Fco. Javier Nievias M.,*  
*Curso:Programación de Servicios Web interactivos con PHP y MySQL.*  
*Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Granada.*  
*5<sup>a</sup> Edición, 2007.*
- [7] PHP, <http://www.webestilo.com/php>
- [8] PHP, <http://www.php.net/>
- [9] WSDL, <http://www.w3.org/TR/wsdl>
- [10] JQuery, <http://jquery.com/>
- [11] *Pedro Daniel Borchers J. , 2004. Java 2 Micro Edition: Soporte Bluetooth.*  
*Universidad Carlos III de Madrid.*
- [12] *Alberto Gimeno B., 2004. JSR-82: Bluetooth desde Java.*
- [13] *Alberto Moreno T., 2006. Seguridad en Bluetooth.*

- [14] *Bluecove*, <http://bluecove.org/>
- [15] *JSR 82*, <http://www.jcp.org/en/jsr/detail?id=82>
- [16] *GNU Licenses*, <http://www.gnu.org/licenses/licenses.es.html>
- [17] *Especificación Bluetooth*,  
<http://www.bluetooth.com/developer/specification/specification.asp>