

Oficina de Arquitectura

División de Servicios Digitales Departamentales
Ministerio de Justicia

ARQUITECTURA DE REFERENCIA

Sistemas de información basados en la
Plataforma JEE
Versión 2023



DOCUMENTO: *AR_PlataformaJEE_v2023.docx*

CATEGORIA: *Uso Interno DSDD*

Queda prohibida cualquier tipo de explotación y, en particular, la reproducción, distribución, comunicación pública y/o transformación, total o parcial, por cualquier medio, de este documento sin el previo consentimiento expreso y por escrito del Ministerio de Justicia.

CONTROL DOCUMENTAL

TITULO: Arquitectura de Referencia – Sistemas de información basados en la Plataforma JEE – Versión 2023

ENTIDAD DE DESTINO: DSDD / Ministerio de Justicia

FICHERO: AR_PlataformaJEE_v2023.docx

RESUMEN: Arquitectura de Referencia para el diseño, construcción y mantenimiento de sistemas de información basados en la Plataforma JEE, versión 2023.

HISTÓRICO DE VERSIONES

VERSIÓN	FECHA	AUTOR	DESCRIPCIÓN
1.0	pendiente	Arquitectura	Versión inicial

Índice

1 INTRODUCCIÓN.....	15
1.1 SISTEMAS DE INFORMACIÓN EN EL MINISTERIO DE JUSTICIA.....	16
1.1.1 Diseño centrado en los datos	16
1.1.2 Integración con sistemas compartidos	16
1.1.3 Categorías funcionales típicas	16
1.2 ALCANCE.....	17
1.3 ÁMBITO DE APLICACIÓN	17
1.3.1 Sistemas de información construidos desde cero.....	17
1.3.2 Re-ingeniería de sistemas de información ya implantados.....	18
1.3.3 Adaptación de sistemas de información externos.....	18
1.3.4 Fuera de ámbito	18
1.4 AUDIENCIA	19
1.5 CONFIDENCIALIDAD.....	19
1.6 MANTENIMIENTO Y REVISIONES.....	19
1.7 REFERENCIAS.....	19
1.7.1 Normativa	19
1.7.2 Guías de aplicación de normativa	20
1.7.3 WS-I.....	21
1.7.4 OWASP Top 10	21
1.7.5 eIDAS	21
2 ARQUITECTURA DEL SISTEMA.....	22
2.1 MODELO DE SISTEMA	22
2.1.1 Interfaces provistos	23
2.1.2 Interfaces requeridos.....	23
2.1.3 Puertos	23
2.1.4 Operativas de negocio.....	24
2.1.5 Datos	24
2.2 DESCOMPOSICIÓN DE SISTEMAS DE INFORMACIÓN.....	25
2.3 INTERACCIÓN ENTRE SISTEMAS	26
2.4 ÁMBITO LIMITADO DEL ENTORNO DE PRODUCCIÓN	27
2.5 CATÁLOGO DE ESTEREOTIPOS DE SUBSISTEMAS APlicATIVOS.....	28
2.5.1 Aplicación Web.....	29
2.5.2 Fachada Web	29
2.5.3 Aplicación de servicios Web.....	30
2.5.4 Aplicación Híbrida.....	31
2.6 ESTILOS ARQUITECTURALES	32
2.6.1 Arquitectura monolítica modular.....	33
2.6.2 Arquitectura de miniservicios	34
2.6.3 Arquitectura de microservicios	36
3 ARQUITECTURA FÍSICA	37
3.1 COMPONENTES FÍSICOS	37
3.1.1 Nodos	37
3.1.1.1 Servidores	38
3.1.1.2 Elementos de red	39
3.1.1.3 Sistemas externos.....	39
3.1.2 Conexiones.....	40
3.1.3 Artefactos	40
3.2 DETERMINACIÓN DEL DIAGRAMA DE RED	41
3.2.1 Primer nivel.....	41
3.2.1.1 Interfaz Web accesible desde el puerto público	42
3.2.1.2 Interfaz WS accesible desde el puerto público.....	42
3.2.1.3 Interfaz Web accesible desde el puerto restringido o privado	43

3.2.1.4	Interfaz WS accesible desde el puerto restringido o privado.....	44
3.2.2	Segundo nivel.....	44
3.2.3	Tercer nivel.....	45
3.3	DIMENSIONAMIENTO DE LA CAPACIDAD.....	46
3.3.1	Capacidad de las unidades de almacenamiento	46
3.3.2	Capacidad de la base de datos.....	46
3.3.3	Tiempo máximo de respuesta	46
3.3.4	Distribución de carga.....	47
3.3.5	Tamaño del clúster de servidores de aplicaciones	47
3.3.6	Disponibilidad	48
3.3.7	Revisión del dimensionamiento de la capacidad	48
4	ARQUITECTURA DE INTEGRACIÓN.....	49
4.1	MIDDLEWARE DE INTEGRACIÓN.....	49
4.1.1	Arquitectura física.....	50
4.1.2	Arquitectura lógica	52
4.2	DISEÑO DE SERVICIOS WEB INTEROPERABLES.....	53
4.2.1	WSDL-first	53
4.2.2	Conformidad con WS-I	54
4.2.3	MTOM.....	56
4.2.4	Activos semánticos	56
4.3	PLATAFORMA DE SERVICIOS COMPARTIDOS.....	57
4.3.1	Oracle APEX	57
4.4	INTEGRACIÓN CON EL SISTEMA DE BUSINESS INTELLIGENCE.....	58
5	ARQUITECTURA LÓGICA.....	60
5.1	PATRÓN LAYERED ARCHITECTURE	60
5.2	CAPAS LÓGICAS EN ESTEREOTIPOS APlicATIVOS.....	62
5.2.1	Aplicación Web.....	62
5.2.2	Fachada Web	63
5.2.3	Aplicación de servicios Web.....	63
5.2.4	Aplicación Híbrida.....	64
5.3	COMPONENTES LÓGICOS	65
5.3.1	Definidos por un interfaz.....	65
5.3.2	Patrón Singleton	66
5.3.3	Patrón Stateless	66
5.3.4	Patrón Business Facade	66
5.3.5	Manifestación de componentes por artefactos	67
5.4	CONTENEDOR DE COMPONENTES	68
5.4.1	Principio Inversion of Control	68
5.4.2	Patrón Dependency Injection	69
5.4.3	Framework Spring	69
5.5	MODELO DE DOMINIO	69
5.5.1	Patrón Value Object	69
5.5.2	Desacoplamiento entre el modelo de dominio y la capa de presentación.....	70
5.5.3	Patrón Domain Entity.....	70
5.6	INTERACCIÓN ENTRE COMPONENTES	71
5.6.1	Patrón Data Transfer Object	71
5.6.2	Excepciones	72
5.7	CATÁLOGO DE ESTEREOTIPOS DE COMPONENTE.....	73
5.7.1	DAO	73
5.7.2	Conector	74
5.7.3	Componente	74
5.7.4	Controlador.....	75
5.7.5	Servicio Web	76
5.7.6	Filtro Web	76
5.8	TAREAS DE MANTENIMIENTO.....	77
5.9	SOLICITUDES DE TRABAJO POR LOTES.....	78

6 ARQUITECTURA DE DATOS	80
6.1 GESTIÓN DE DATOS ESTRUCTURADOS.....	80
6.1.1 Modelo lógico	80
6.1.2 Modelo físico	81
6.1.3 Esquema	81
6.1.4 Control de acceso a la base de datos	83
6.1.5 JDBC data source	83
6.1.6 Entidades.....	84
6.1.7 Unidad de persistencia	85
6.1.8 Contexto de persistencia	85
6.1.9 Entity manager	86
6.1.10 Transaction manager.....	87
6.1.11 Delimitación de transacciones.....	88
6.1.12 Componentes DAO basados en Spring Data	88
6.1.13 JPQL.....	89
6.1.14 Consultas.....	89
6.1.15 Filtros de búsqueda	90
6.1.16 Consultas exactas y consultas laxas.....	90
6.1.17 Limitación de campos obtenidos	91
6.1.18 Consultas con resultados mixtos.....	92
6.1.19 Paginación	92
6.1.20 Depuración de consultas	93
6.1.21 Mantenimiento de datos	93
6.2 GESTIÓN DE DATOS NO ESTRUCTURADOS	94
6.2.1 Objetos binarios.....	94
6.2.2 Persistencia de objetos binarios.....	94
6.2.3 Procesamiento de objetos binarios	95
6.2.4 Ficheros temporales	95
6.2.5 Unidad de almacenamiento para ficheros finales	95
6.2.6 Control de acceso a la unidad de almacenamiento	96
6.2.7 Organización en carpetas	96
6.2.8 Componentes DAO de acceso a datos no estructurados	96
6.2.9 Mantenimiento de datos	97
6.2.10 Almacenamiento de objetos binarios en la base de datos.....	97
7 ARQUITECTURA DEL INTERFAZ DE USUARIO	99
7.1 CONTROL DE LA VISUALIZACIÓN DESDE EL SERVIDOR.....	99
7.2 DISEÑO RESTFUL DE INTERFACES DE USUARIO WEB	100
7.2.1 Tipos de recurso.....	100
7.2.2 Primitivas	101
7.2.3 Recursos anidados.....	101
7.2.4 Consultas y paginación	102
7.2.5 Jerarquías de clases de recurso	103
7.2.6 Diagrama de esquema de navegación.....	104
7.3 INTERNACIONALIZACIÓN	106
7.4 RESPONSIVIDAD	106
7.5 ACCESIBILIDAD	106
7.6 NORMALIZACIÓN DE LA APARIENCIA Y EXPERIENCIA DE USUARIO.....	107
7.6.1 Maqueta HTML.....	107
7.6.2 Motor de plantillas	108
8 ARQUITECTURA DE SEGURIDAD	109
8.1 SISTEMAS O SUBSISTEMAS FRONTERA	109
8.2 INTERFACES Y NIVELES DE ACCESO.....	110
8.3 OPEN WEB APPLICATION SECURITY PROJECT.....	111
8.3.1 Rotura de control de acceso	111
8.3.1.1 Control de autorización	111

8.3.2 Fallos criptográficos.....	112
8.3.2.1 Transport Layer Security	112
8.3.2.2 HTTP Strict Transport Security.....	113
8.3.2.3 Identificadores de recurso	113
8.3.3 Inyección	113
8.3.3.1 Validación sintáctica de los datos de entrada	114
8.3.3.2 Validación semántica de los datos de entrada	114
8.3.3.3 Validación de archivos	115
8.3.3.4 Codificación de datos	116
8.3.3.5 Cabecera Content-Security-Policy	116
8.3.4 Diseño inseguro.....	116
8.3.4.1 No mostrar información que facilite el fingerprinting.....	117
8.3.4.2 No exponer información en campos ocultos	117
8.3.4.3 Inventariar la información de carácter personal.....	117
8.3.4.4 Derecho de información acerca del tratamiento de datos personales.....	118
8.3.5 Configuración de seguridad incorrecta.....	118
8.3.5.1 No mostrar información técnica al usuario al tratar excepciones	118
8.3.5.2 Protección de las cookies.....	119
8.3.5.3 Inhabilitar el uso de entidades externas XML.....	119
8.3.6 Componentes vulnerables y anticuados	119
8.3.7 Fallos de identificación y autenticación	120
8.3.7.1 Interfaces anónimos y autenticados	120
8.3.7.2 Control de autenticación basado en Cl@ve	123
8.3.7.3 Control de autenticación basado en Autentica	127
8.3.7.4 Control de autenticación basado en Active Directory	129
8.3.7.5 Control de autenticación mediante TLS	130
8.3.7.6 Control de autenticación mediante WSS	131
8.3.7.7 Propagación del contexto de seguridad	132
8.3.7.8 Gestión de sesiones	133
8.3.8 Fallos de integridad de software y datos	134
8.3.8.1 Uso de librerías normalizadas	134
8.3.8.2 Impedir la deserialización insegura	134
8.3.9 Fallos de registro y monitorización de seguridad	135
8.3.9.1 Registros de actividad	135
8.3.9.2 Eventos de traza	136
8.3.9.3 Eventos de seguridad	138
8.3.9.4 Eventos de intercambio de datos	141
8.3.9.5 Identificadores de sesión y de petición.....	144
8.3.9.6 Recolección y análisis de eventos.....	145
8.3.9.7 Auditoría de cambios de estado de entidades.....	146
8.3.10 Falsificación de solicitudes en el servidor	146
8.3.10.1 Validación de URL.....	146
9 ARQUITECTURA DEL CÓDIGO FUENTE.....	148
9.1 IDENTIFICACIÓN DE SISTEMAS Y SUBSISTEMAS A EFECTOS DE GESTIÓN.....	148
9.1.1 Identificación de sistemas de información.....	148
9.1.2 Identificación de subsistemas aplicativos.....	148
9.2 ESTILOS DE NOMENCLATURA	149
9.2.1 Upper Camel Case	149
9.2.2 Lower Camel Case	150
9.2.3 Snake Case	150
9.2.4 Kebab Case	151
9.2.5 Dot Case	151
9.3 LICENCIAMIENTO.....	151
9.4 ARQUETIPOS DE PROYECTO APlicativo	152
9.4.1 Arquetipo de proyecto de Sistema de información	153
9.4.2 Arquetipo de proyecto de Aplicación Web	153
9.4.3 Arquetipo de proyecto de Fachada Web.....	154
9.4.4 Arquetipo de proyecto de Aplicación de servicios Web	155

9.4.5	Arquetipo de proyecto de Aplicación híbrida.....	155
9.4.6	Arquetipo de proyecto de Librería de componentes compartidos	156
9.4.7	Arquetipo de proyecto de Librería de especificación de servicios Web.....	156
9.5	CODIFICACIÓN	157
9.6	ARTEFACTOS.....	157
9.6.1	Artefactos POM	157
9.6.2	Artefactos WAR	158
9.6.3	Artefactos JAR.....	158
9.6.4	Artefactos de scripts de base de datos	160
9.6.5	Artefactos de recursos estáticos Web.....	160
9.7	APIs Y FRAMEWORKS BASE.....	161
9.8	PAQUETES	162
9.9	COMPONENTES.....	163
9.9.1	Interfaz.....	163
9.9.2	Implementación	164
9.9.3	Nombre	164
9.9.4	Estereotipo	165
9.10	CONFIGURACIÓN DEL CONTEXTO DE APLICACIÓN	165
9.11	MODELO DE DOMINIO	166
9.12	EXCEPCIONES	166
9.13	URL BASE	167
9.14	INTERFACES WEB	168
9.14.1	URL.....	168
9.14.2	Mecanismos de autenticación	168
9.14.3	Recursos estáticos Web compartidos	169
9.14.4	Recursos estáticos Web aplicativos	170
9.15	INTERFACES WS.....	170
9.15.1	URL.....	170
9.15.2	Espacios de nombres	170
9.15.3	Cabecera SOAPAction	171
9.16	OBJETOS DE LA BASE DE DATOS	172
9.17	DIRECCIONES DE CORREO.....	174
10	ARQUITECTURA DE LA CONFIGURACIÓN	176
10.1	CATEGORIZACIÓN DE LOS RECURSOS DE CONFIGURACIÓN	176
10.1.1	Ámbito.....	176
10.1.2	Visibilidad	176
10.1.3	Mutabilidad	177
10.2	PROPIEDADES DE CONFIGURACIÓN	177
10.2.1	Recurso application.properties	177
10.2.2	Recurso environment.properties	178
10.2.3	Recurso administrado application.properties	179
10.2.4	Recurso test-application.properties	180
10.2.5	Carga de las propiedades de configuración	181
10.3	CONFIGURACIÓN DE TRAZAS.....	182
10.3.1	Recurso logging.properties	182
10.3.2	Recurso administrado logging.properties	183
10.3.3	Recurso test-logging.properties	183
10.3.4	Carga de la configuración de trazas.....	184

Índice de ilustraciones

Ilustración 2-1 Modelo de sistema	22
Ilustración 2-2 Componentes lógicos	24
Ilustración 2-3 Descomposición del sistema en subsistemas.....	25
Ilustración 2-4 Interacciones entre sistemas	26
Ilustración 2-5 Ámbito limitado del entorno de producción	27
Ilustración 2-6 Estereotipo de Aplicación Web	29
Ilustración 2-7 Estereotipo de Fachada Web.....	30
Ilustración 2-8 Estereotipo de Aplicación de servicios Web	31
Ilustración 2-9 Estereotipo de Aplicación Híbrida	32
Ilustración 2-10 Arquitectura monolítica modular	33
Ilustración 2-11 Arquitectura de miniservicios	34
Ilustración 3-1 Elementos en primer nivel para subsistemas aplicativos con un interfaz Web accesible desde el puerto público	42
Ilustración 3-2 Elementos en primer nivel para subsistemas aplicativos con interfaces WS accesibles desde el puerto público	43
Ilustración 3-3 Elementos en primer nivel para subsistemas aplicativos con un interfaz Web accesible desde el puerto restringido o privado	43
Ilustración 3-4 Elementos en primer nivel para subsistemas aplicativos con interfaces WS accesibles desde el puerto restringido o privado.....	44
Ilustración 3-5 Elementos en el segundo nivel	45
Ilustración 3-6 Elementos en el tercer nivel	45
Ilustración 4-1 Arquitectura física del middleware externo	51
Ilustración 4-2 Arquitectura física del middleware interno	52
Ilustración 4-3 Arquitectura lógica de los servicios de intermediación	52
Ilustración 5-1 Patrón Layered Architecture con tres capas	61
Ilustración 5-2 Patrón Layered Architecture con descomposición en subcapas adyacentes.....	62
Ilustración 5-3 Capas lógicas en subsistemas aplicativos con estereotipo Aplicación Web	63
Ilustración 5-4 Capas lógicas en subsistemas aplicativos con estereotipo Fachada Web	63
Ilustración 5-5 Capas lógicas en subsistemas aplicativos con estereotipo Aplicación de Servicios..	64
Ilustración 5-6 Capas lógicas en subsistemas aplicativos con estereotipo Aplicación Híbrida	64
Ilustración 5-7 Patrón Business Facade	67
Ilustración 7-1 Representación de colecciones como consultas paginadas	102
Ilustración 7-2 Diagrama de esquema de navegación	105
Ilustración 8-1 Sistemas y subsistemas frontera	110

Índice de tablas

Tabla 3-1 Especificación de servidor frontal externo	38
Tabla 3-2 Especificación de servidor frontal interno	38
Tabla 3-3 Especificación de servidor de aplicaciones	39
Tabla 3-4 Especificación de servidor de base de datos	39
Tabla 3-5 Tiempos de indisponibilidad	48
Tabla 4-1 Especificación de servidor de integración	51
Tabla 4-2 Estándares de WS-I Basic Profile 1.1	55
Tabla 4-3 Estándares de WS-I Attachments Profile 1.0	55
Tabla 4-4 Estándares de WS-I Basic Security Profile 1.0	56
Tabla 7-1 Convenios de diseño RESTful de interfaces Web.....	101
Tabla 7-2 Parámetros de consulta y paginación	102
Tabla 7-3 Parámetro de tipo o subclase de recurso	104
Tabla 8-1 Determinación del mecanismo de autenticación y el tipo de credencial	122
Tabla 8-2 Determinación del LoA de Cl@ve	123
Tabla 8-3 Atributos presentes en la petición SAML de Cl@ve	123
Tabla 8-4 Proveedores de identificación disponibles en Cl@ve	124
Tabla 8-5 Atributos presentes en la respuesta SAML de Cl@ve	124
Tabla 8-6 Determinación de la clase de Principal en Cl@ve	125
Tabla 8-7 Autorizaciones a incluir en la autenticación con Cl@ve	125
Tabla 8-8 Estructura del Principal en la autenticación con Cl@ve	126
Tabla 8-9 Determinación del nivel de seguridad en Autentica	127
Tabla 8-10 Atributos presentes en la respuesta SAML de Autentica	127
Tabla 8-11 Correspondencia entre las autorizaciones de Autentica y Spring Security.....	128
Tabla 8-12 Estructura del Principal en la autenticación con Autentica.....	128
Tabla 8-13 Correspondencia entre los grupos de Active Directory y las autorizaciones de Spring Security	129
Tabla 8-14 Estructura del Principal en la autenticación con Active Directory	129
Tabla 8-15 Estructura del Principal en la autenticación con TLS	130
Tabla 8-16 Autorizaciones a incluir en la autenticación con TLS	131
Tabla 8-17 Estructura de la cabecera WS-S para la propagación de identidad.....	133
Tabla 8-18 Registros de actividad de los componentes lógicos.....	135
Tabla 8-19 Evento de traza.....	136
Tabla 8-20 Niveles de severidad en los eventos de traza	137
Tabla 8-21 Tipos de eventos aplicativos de seguridad.....	138
Tabla 8-22 Evento de inicio de sesión con autenticación correcta	138
Tabla 8-23 Evento de intento de inicio de sesión con autenticación fallida	139
Tabla 8-24 Evento de fin de sesión	139
Tabla 8-25 Evento de invocación de operativa de negocio	140
Tabla 8-26 Evento de intento de invocación con autorización fallida	140
Tabla 8-27 Tipos de eventos de intercambio de datos	141
Tabla 8-28 Evento de intercambio de datos general mediante servicios Web	142
Tabla 8-29 Evento de intercambio de datos a través de la Plataforma de intermediación de datos (síncrono)	143
Tabla 9-1 Estructura de proyectos de Sistema de información	153
Tabla 9-2 Estructura de proyectos de Aplicación Web	154
Tabla 9-3 Estructura de proyectos de Fachada Web	154
Tabla 9-4 Estructura de proyectos de Aplicación de servicios Web	155
Tabla 9-5 Estructura de proyectos de Aplicación híbrida	156
Tabla 9-6 Estructura de proyectos de Librería de componentes compartidos	156
Tabla 9-7 Estructura de proyectos de Librería de especificación de servicios Web	157
Tabla 9-8 Coordenadas Maven de artefactos POM	157
Tabla 9-9 Estructura interna de artefactos WAR	158
Tabla 9-10 Coordenadas Maven de artefactos WAR	158
Tabla 9-11 Estructura interna de artefactos JAR de librerías de componentes compartidos	159
Tabla 9-12 Estructura interna de artefactos JAR de librerías de especificación de servicios Web	159
Tabla 9-13 Coordenadas Maven de artefactos JAR.....	159

Tabla 9-14 Estructura interna de artefacto de scripts de base de datos	160
Tabla 9-15 Coordenadas Maven de artefactos de scripts de base de datos	160
Tabla 9-16 Estructura interna de artefacto de recursos estáticos Web.....	160
Tabla 9-17 Coordenadas Maven de artefactos de scripts de base de datos	161
Tabla 10-1 Especificación del recurso application.properties.....	178
Tabla 10-2 Especificación del recurso environment.properties	179
Tabla 10-3 Especificación del recurso administrado application.properties	180
Tabla 10-4 Especificación del recurso test-application.properties	181
Tabla 10-5 Especificación del recurso logging.properties	182
Tabla 10-6 Especificación del recurso administrado logging.properties	183
Tabla 10-7 Especificación del recurso test-logging.properties	184

Índice de normas

Norma AR-CFG-01	177
Norma AR-CFG-02	177
Norma AR-CFG-03	178
Norma AR-CFG-04	178
Norma AR-CFG-05	180
Norma AR-CFG-06	181
Norma AR-CFG-07	181
Norma AR-CFG-08	182
Norma AR-CFG-09	183
Norma AR-CFG-10	183
Norma AR-CFG-11	184
Norma AR-DAT-01	80
Norma AR-DAT-02	81
Norma AR-DAT-03	81
Norma AR-DAT-04	81
Norma AR-DAT-05	81
Norma AR-DAT-06	81
Norma AR-DAT-07	83
Norma AR-DAT-08	84
Norma AR-DAT-09	84
Norma AR-DAT-10	84
Norma AR-DAT-11	85
Norma AR-DAT-12	86
Norma AR-DAT-13	86
Norma AR-DAT-14	87
Norma AR-DAT-15	88
Norma AR-DAT-16	89
Norma AR-DAT-17	89
Norma AR-DAT-18	89
Norma AR-DAT-19	90
Norma AR-DAT-20	90
Norma AR-DAT-21	91
Norma AR-DAT-22	91
Norma AR-DAT-23	91
Norma AR-DAT-24	92
Norma AR-DAT-25	92
Norma AR-DAT-26	93
Norma AR-DAT-27	93
Norma AR-DAT-28	93
Norma AR-DAT-29	94
Norma AR-DAT-30	94
Norma AR-DAT-31	95
Norma AR-DAT-32	95
Norma AR-DAT-33	96
Norma AR-DAT-34	96
Norma AR-DAT-35	96
Norma AR-DAT-36	97
Norma AR-DAT-37	97
Norma AR-DAT-38	97
Norma AR-FIS-01	38
Norma AR-FIS-02	39
Norma AR-FIS-03	40
Norma AR-FIS-04	41
Norma AR-FIS-05	46
Norma AR-FIS-06	46
Norma AR-FIS-07	46

Norma AR-FIS-08	47
Norma AR-FIS-09	47
Norma AR-FIS-10	48
Norma AR-FIS-11	48
Norma AR-GUI-01	100
Norma AR-GUI-02	102
Norma AR-GUI-03	105
Norma AR-GUI-04	106
Norma AR-GUI-05	106
Norma AR-GUI-06	107
Norma AR-GUI-07	107
Norma AR-GUI-08	107
Norma AR-GUI-09	108
Norma AR-INT-01	50
Norma AR-INT-02	50
Norma AR-INT-03	53
Norma AR-INT-04	54
Norma AR-INT-05	54
Norma AR-INT-06	56
Norma AR-INT-07	57
Norma AR-INT-08	57
Norma AR-INT-09	58
Norma AR-INT-10	59
Norma AR-LOG-01	60
Norma AR-LOG-02	61
Norma AR-LOG-03	62
Norma AR-LOG-04	65
Norma AR-LOG-05	66
Norma AR-LOG-06	66
Norma AR-LOG-07	67
Norma AR-LOG-08	68
Norma AR-LOG-09	68
Norma AR-LOG-10	69
Norma AR-LOG-11	70
Norma AR-LOG-12	70
Norma AR-LOG-13	71
Norma AR-LOG-14	71
Norma AR-LOG-15	71
Norma AR-LOG-16	72
Norma AR-LOG-17	73
Norma AR-LOG-18	73
Norma AR-LOG-19	74
Norma AR-LOG-20	74
Norma AR-LOG-21	75
Norma AR-LOG-22	76
Norma AR-LOG-23	76
Norma AR-LOG-24	77
Norma AR-LOG-25	77
Norma AR-LOG-26	78
Norma AR-LOG-27	78
Norma AR-SEG-01	110
Norma AR-SEG-02	111
Norma AR-SEG-03	111
Norma AR-SEG-04	112
Norma AR-SEG-05	112
Norma AR-SEG-06	112
Norma AR-SEG-07	113
Norma AR-SEG-08	113

Norma AR-SEG-09	114
Norma AR-SEG-10	114
Norma AR-SEG-11	115
Norma AR-SEG-12	115
Norma AR-SEG-13	115
Norma AR-SEG-14	116
Norma AR-SEG-15	116
Norma AR-SEG-16	117
Norma AR-SEG-17	117
Norma AR-SEG-18	117
Norma AR-SEG-19	117
Norma AR-SEG-20	118
Norma AR-SEG-21	118
Norma AR-SEG-22	118
Norma AR-SEG-23	119
Norma AR-SEG-24	119
Norma AR-SEG-25	120
Norma AR-SEG-26	120
Norma AR-SEG-27	121
Norma AR-SEG-28	126
Norma AR-SEG-29	128
Norma AR-SEG-30	129
Norma AR-SEG-31	131
Norma AR-SEG-32	132
Norma AR-SEG-33	133
Norma AR-SEG-34	133
Norma AR-SEG-35	134
Norma AR-SEG-36	134
Norma AR-SEG-37	134
Norma AR-SEG-38	135
Norma AR-SEG-39	135
Norma AR-SEG-40	135
Norma AR-SEG-41	137
Norma AR-SEG-42	137
Norma AR-SEG-43	140
Norma AR-SEG-44	144
Norma AR-SEG-45	144
Norma AR-SEG-46	144
Norma AR-SEG-47	144
Norma AR-SEG-48	145
Norma AR-SEG-49	145
Norma AR-SEG-50	146
Norma AR-SEG-51	147
Norma AR-SIS-01	23
Norma AR-SIS-02	25
Norma AR-SIS-03	27
Norma AR-SIS-04	27
Norma AR-SIS-05	28
Norma AR-SIS-06	33
Norma AR-SIS-07	35
Norma AR-SIS-08	36
Norma AR-SRC-01	148
Norma AR-SRC-02	148
Norma AR-SRC-03	149
Norma AR-SRC-04	149
Norma AR-SRC-05	150
Norma AR-SRC-06	150
Norma AR-SRC-07	151

Norma AR-SRC-08	151
Norma AR-SRC-09	152
Norma AR-SRC-10	152
Norma AR-SRC-11	157
Norma AR-SRC-12	157
Norma AR-SRC-13	158
Norma AR-SRC-14	159
Norma AR-SRC-15	160
Norma AR-SRC-16	161
Norma AR-SRC-17	161
Norma AR-SRC-18	162
Norma AR-SRC-19	163
Norma AR-SRC-20	164
Norma AR-SRC-21	164
Norma AR-SRC-22	165
Norma AR-SRC-23	165
Norma AR-SRC-24	166
Norma AR-SRC-25	166
Norma AR-SRC-26	167
Norma AR-SRC-27	168
Norma AR-SRC-28	168
Norma AR-SRC-29	169
Norma AR-SRC-30	170
Norma AR-SRC-31	170
Norma AR-SRC-32	171
Norma AR-SRC-33	171
Norma AR-SRC-34	172
Norma AR-SRC-35	172
Norma AR-SRC-36	173
Norma AR-SRC-37	173
Norma AR-SRC-38	173
Norma AR-SRC-39	174
Norma AR-SRC-40	174
Norma AR-SRC-41	174
Norma AR-SRC-42	175

1 Introducción

La palabra **arquitectura** se emplea normalmente con dos sentidos diferentes: como diseño arquitectónico, o como estilo arquitectural.

El **diseño arquitectónico** de un sistema de información, también conocido como diseño de alto nivel, hace referencia al conjunto de decisiones de diseño, establecidas con carácter previo a su construcción, que tienen un mayor impacto en el resultado final.

Un **estilo arquitectural** es una abstracción que hace referencia a un conjunto de elementos comunes que se encuentran presentes en el diseño arquitectónico de varios sistemas de información. Son ejemplos de estilo arquitectural SOA, JEE, microservicios, etc.

Una Arquitectura de Referencia es el diseño arquitectónico de un sistema de información tipo, que sirve como modelo y solución de las problemáticas más frecuentes o relevantes en un determinado ámbito.

Este documento establece la Arquitectura de Referencia que habrán de seguir los sistemas de información basados en JEE que se construyan e implanten en la DSDD, así como su mantenimiento posterior.

Para su definición, se han establecido los siguientes objetivos:

- Debe estar adaptada a las características que comparten los sistemas de información más habituales que se diseñan, construyen y mantienen en la DSDD.
- Debe estar adaptada a las características de las instalaciones de que dispone la DSDD y sus políticas y mecanismos de explotación.
- Debe perseguir balancear los costes de infraestructura, los de diseño, construcción y mantenimiento del software, y los de explotación del sistema productivo, con el uso real previsto.
- Debe ser compatible con el resto de normativa técnica y metodológica existente en la DSDD.
- Debe tener una orientación a largo plazo, de manera que, a lo largo de su vida útil, no sufra modificaciones sustanciales. Cuando se considere que ésta ha finalizado, deberá ser sustituida por una nueva Arquitectura de Referencia, con un enfoque diferente, acorde al estado del arte en ese momento.

Esta Arquitectura de Referencia contempla el estilo arquitectónico **monolítico modular**, o bien el orientado a **miniservicios**, los cuales pueden comunicarse entre sí mediante servicios Web basados en SOAP.

1.1 Sistemas de información en el Ministerio de Justicia

La mayoría de los sistemas de información que se diseñan, construyen y mantienen en el Ministerio de Justicia comparten una serie de rasgos comunes desde el punto de vista del diseño, que se describen a continuación.

1.1.1 Diseño centrado en los datos

En la mayor parte de los sistemas de información de las Administraciones Públicas, el principal activo son los datos, por encima de las operativas de negocio.

Su diseño, por tanto, se centra en los datos, en la correcta y temprana definición de un modelo de datos adecuado y completo. Como resultado de esta definición, se obtiene un modelo de datos complejo, amplio y ramificado, que contrasta con la relativa sencillez de las operativas de negocio, y en ocasiones, incluso con su reducido número.

Por otro lado, al ser los datos el principal activo, se vuelven fundamentales los aspectos relacionados con su seguridad.

1.1.2 Integración con sistemas compartidos

Los sistemas de información de las Administraciones Públicas a menudo se integran con otros sistemas, de carácter compartido, los cuales proveen funcionalidades de uso frecuente, permitiendo así su reutilización.

En su diseño, por tanto, se refleja la participación de sistemas externos, con los que el sistema de información interactúa intercambiando información.

Debido a ello, se vuelven fundamentales los aspectos relacionados con la interoperabilidad.

1.1.3 Categorías funcionales típicas

De acuerdo a su funcionalidad, los sistemas de información más frecuentes en el Ministerio de Justicia son:

- **Sistemas de tramitación:** son sistemas que sustentan la tramitación electrónica de procedimientos o servicios administrativos o registrales. Sus usuarios son ciudadanos, así como empleados tramitadores o encargados del registro, en el ámbito del organismo encargado del procedimiento.
- **Sistemas de registro:** son sistemas donde se inscriben determinados hechos, circunstancias o derechos, acreditados por un título. El acceso a la información previamente inscrita se realiza mediante la expedición de certificaciones o manifestaciones.
- **Sistemas de gestión administrativa:** son sistemas que prestan funciones de gestión administrativa de índole general. No sustentan procedimientos

administrativos ni registrales, aunque podrían prestar servicio a estos sistemas si requieren de alguna de sus funciones. Sus usuarios suelen ser empleados en el ámbito de un organismo.

- **Sistemas de uso compartido:** son sistemas internos, que prestan funciones utilizadas de forma recurrente por sistemas de tramitación o de gestión administrativa, principalmente a través de servicios Web. Su objeto, por tanto, es la reutilización de funcionalidad.

1.2 Alcance

La Arquitectura de Referencia ofrece una visión integral del diseño arquitectónico de sistemas de información, el cual comprende:

- La **descomposición modular** del sistema en subsistemas.
- La determinación de la **arquitectura física y lógica** de éstos, así como la de los **datos** que explotan.
- Los estándares de **integración** que garantizarán su interoperabilidad con otros sistemas o subsistemas.
- La organización y requisitos de los **interfaces de usuario** Web que provean.
- Los requisitos mínimos de **seguridad lógica** que protegerán el ámbito del sistema.
- La organización del **código fuente** dentro de los proyectos aplicativos, que facilitará la construcción y el mantenimiento posterior.
- Los mecanismos de gestión de los recursos de **configuración**.

1.3 Ámbito de aplicación

Las normas de diseño arquitectónico que se describen en este documento tienen aplicación en los escenarios que se describen a continuación. No obstante, la DSDD podría requerir la aplicación individualizada de algunas de algunas de estas pautas en el diseño, construcción o mantenimiento de sistemas de información que se encuentren fuera de su ámbito de aplicación.

Si el sistema de información requiriese características especiales, tales que su diseño arquitectónico no pueda resolverse completamente aplicando las pautas definidas en este documento y que le sean de aplicación, deberán consensuarse con la Oficina de Arquitectura, y aprobarse por ésta, pautas de diseño adicionales, específicas y adecuadas a dichas características.

1.3.1 Sistemas de información construidos desde cero

Comprende los sistemas de información basados en la plataforma JEE, que vayan a ser diseñados y construidos por la DSDD a partir del análisis de las necesidades de

los usuarios, siguiendo la normativa técnica y metodológica establecida a tal efecto por la DSDD y que le sea de aplicación, así como su mantenimiento posterior.

En este ámbito, las pautas de diseño arquitectónico descritas en este documento son de aplicación en todo su alcance.

1.3.2 Re-ingeniería de sistemas de información ya implantados

Comprende los sistemas de información basados en la plataforma JEE, que vayan a ser diseñados y construidos por la DSDD a partir de un proceso de re-ingeniería sobre una versión anterior ya implantada, siguiendo la normativa técnica y metodológica establecida a tal efecto por la DSDD y que le sea de aplicación, así como su mantenimiento posterior.

En este ámbito, las pautas de diseño arquitectónico descritas en este documento son de aplicación en todo su alcance.

1.3.3 Adaptación de sistemas de información externos

Comprende los sistemas de información basados en la plataforma JEE, cuyo software haya sido diseñado y construido por otros organismos diferentes a la DSDD, y que vaya a ser adaptado y adecuado para poderse implantar sobre infraestructura proporcionada y operada por la DSDD, así como su mantenimiento posterior.

Este proceso de adaptación comprende las modificaciones que es necesario realizar sobre el software para que pueda funcionar sobre dicha infraestructura, siguiendo la normativa técnica y metodológica establecida a tal efecto por la DSDD y que le sea de aplicación.

Ocasionalmente, puede ser necesario realizar además una adecuación del software mediante la restricción o modificación no sustancial de alguna funcionalidad, siguiendo la normativa técnica y metodológica establecida a tal efecto por la DSDD y que le sea de aplicación.

En este ámbito, serán de aplicación únicamente las pautas de diseño arquitectónico relacionadas con la arquitectura física, la arquitectura de integración y la arquitectura de seguridad que hayan sido consensuadas con la DSDD y aprobadas por ésta.

1.3.4 Fuera de ámbito

Las pautas de diseño arquitectónico que se describen en este documento no serán de aplicación en los siguientes escenarios:

- El diseño, construcción o mantenimiento de sistemas de información que no se basen en la plataforma JEE.
- El mantenimiento de sistemas de información basados en la plataforma JEE y diseñados con anterioridad a la publicación de esta Arquitectura de

Referencia, salvo que la DSDD haya determinado expresamente su aplicabilidad.

- El diseño, construcción o mantenimiento de sistemas de información para los que, a pesar de encontrarse contenidos dentro del ámbito de aplicación de esta Arquitectura de Referencia, la DSDD haya determinado expresamente su no aplicabilidad.

1.4 Audiencia

Los destinatarios del presente documento son los equipos de desarrollo de la DSDD.

1.5 Confidencialidad

El presente documento es propiedad de la DSDD y está clasificado como de USO INTERNO.

El acceso a este documento está restringido al personal de la DSDD, tanto externo como interno. Cualquier acceso fuera de este ámbito, debe ser autorizado de forma expresa por la Dirección de la DSDD.

No podrá ser objeto de reproducción total o parcial, tratamiento informático ni transmisión de ninguna forma o por cualquier medio, ya sea electrónico, mecánico, por fotocopia, registro o cualquiera otro. Asimismo, tampoco podrá ser objeto de préstamo, o cualquier forma de cesión de uso sin el permiso previo y por escrito de la DSDD, titular de los derechos de propiedad intelectual. El incumplimiento de las limitaciones señaladas por cualquier persona que tenga acceso a la documentación será perseguido conforme dicte la ley.

1.6 Mantenimiento y revisiones

Este documento es realizado y mantenido por el equipo de Arquitectura de la DSDD.

1.7 Referencias

1.7.1 Normativa

- [Real Decreto 3/2010, de 8 de enero, por el que se regula el Esquema Nacional de Seguridad en el ámbito de la Administración Electrónica](#)
- [Real Decreto 951/2015, de 23 de octubre, de modificación del Real Decreto 3/2010, de 8 de enero, por el que se regula el Esquema Nacional de Seguridad en el ámbito de la Administración Electrónica](#)
- [Real Decreto 4/2010, de 8 de enero, por el que se regula el Esquema Nacional de Interoperabilidad en el ámbito de la Administración Electrónica](#)

- [Resolución de 3 de octubre de 2012, de la Secretaría de Estado de Administraciones Públicas, por la que se aprueba la Norma Técnica de Interoperabilidad de Catálogo de estándares](#)
- [Resolución de 28 de junio de 2012, de la Secretaría de Estado de Administraciones Públicas, por la que se aprueba la Norma Técnica de Interoperabilidad de Relación de modelos de datos](#)
- [Resolución de 19 de julio de 2011, de la Secretaría de Estado para la Función Pública, por la que se aprueba la Norma Técnica de Interoperabilidad de requisitos de conexión a la red de comunicaciones de las Administraciones Públicas españolas](#)
- [REGLAMENTO \(UE\) 2016/679 DEL PARLAMENTO EUROPEO Y DEL CONSEJO de 27 de abril de 2016 relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos](#)
- [Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales](#)
- [Ley 34/2002, de 11 de julio, de servicios de la sociedad de la información y de comercio electrónico.](#)
- [REGLAMENTO \(UE\) No 910/2014 DEL PARLAMENTO EUROPEO Y DEL CONSEJO de 23 de julio de 2014 relativo a la identificación electrónica y los servicios de confianza para las transacciones electrónicas en el mercado interior y por la que se deroga la Directiva 1999/93/CE](#)
- [Real Decreto 1065/2007, de 27 de julio, por el que se aprueba el Reglamento General de las actuaciones y los procedimientos de gestión e inspección tributaria y de desarrollo de las normas comunes de los procedimientos de aplicación de los tributos](#)
- [Orden EHA/451/2008, de 20 de febrero, por la que se regula la composición del número de identificación fiscal de las personas jurídicas y entidades sin personalidad jurídica](#)
- [Orden INT/2058/2008, de 14 de julio, por la que se modifica la Orden del Ministro del Interior de 7 de febrero de 1997, por la que se regula la Tarjeta de Extranjero, en lo concerniente al número de Identidad de Extranjero](#)

1.7.2 Guías de aplicación de normativa

- [Guía de Seguridad de las TIC CCN-STIC 808: Verificación del cumplimiento del ENS](#)
- [Guía de aplicación de la Norma Técnica de Interoperabilidad de Catálogo de estándares](#)
- [Guía de aplicación de la Norma Técnica de Interoperabilidad de Relación de modelos de datos](#)

- [Guía de aplicación de la Norma Técnica de Interoperabilidad de Requisitos de conexión a la red de comunicaciones de las Administraciones Públicas españolas](#)
- [Guía de auditoría de cumplimiento del Esquema Nacional de Interoperabilidad](#)
- [Guía de la AEPD sobre el uso de las cookies](#)
- [Guía de Perfiles de certificados electrónicos](#)

1.7.3 WS-I

- [WS-I Basic Security Profile 1.0](#)

1.7.4 OWASP Top 10

- [OWASP-Top 10 2021](#)

1.7.5 eIDAS

- [eIDAS SAML Attribute Profile 1.2](#)

2 Arquitectura del sistema

Los sistemas de información automatizados se suelen definir como conjuntos de componentes que colaboran entre sí para recolectar, crear, almacenar, procesar y distribuir información. Incluyen el hardware, el software y los propios datos, así como las personas que interactúan con el sistema.

Para facilitar el diseño de los sistemas de información, es conveniente establecer un modelo de sistema, de alto nivel de abstracción, del que pueda derivarse de forma sencilla el diseño arquitectónico.

2.1 Modelo de sistema

Desde un nivel de abstracción alto, los sistemas de información se entenderán compuestos por los siguientes elementos:

- Los **datos**, que son gestionados por el sistema.
- Las **operativas de negocio**, que extraen, procesan, almacenan y permiten el acceso a dichos datos, y que pueden interactuar, a su vez, con otros sistemas.
- Los **interfaces provistos**, a través de los cuales se disparan las operativas de negocio.
- Los **interfaces requeridos**, a través de los cuales las operativas de negocio interactúan con los interfaces provistos por otros sistemas, para llevar a cabo su cometido.
- Los **puertos**, a través de los cuales el sistema se comunica con el entorno, exponiendo sus interfaces provistos, o accediendo a los interfaces provistos por otros sistemas.

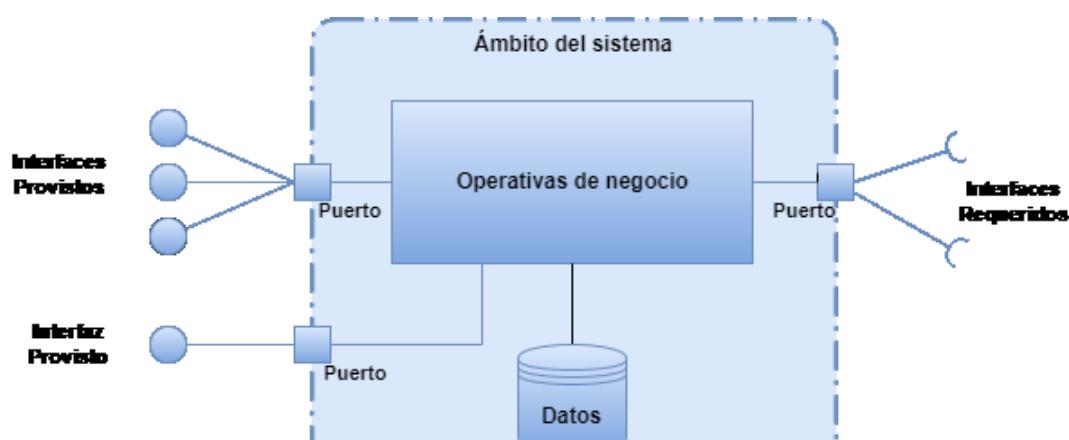


Ilustración 2-1 Modelo de sistema

Norma AR-SIS-01

Los **sistemas de información** se concebirán de acuerdo a un modelo de sistema que establezca un **ámbito limitado** y protegido, que garantice que la única forma de acceder a los **datos** que maneja el sistema sea a través de las **operativas de negocio**, las cuales son posibles de invocar únicamente desde los **interfaces provistos** por el sistema a tal efecto, y que éste expone a través de sus **puertos**, como se indica en la **Ilustración 2-1**.

A continuación, se describen cada uno de los elementos de este modelo.

2.1.1 Interfaces provistos

Los interfaces provistos son aquellos que permiten a los usuarios interactuar con el sistema. Según el tipo de usuario al que estén orientados, se distinguen los siguientes tipos de interfaz provisto:

- **Interfaz Web**, que permiten que usuarios humanos interactúen con el sistema mediante un navegador Web.
- **Interfaz de servicios Web**, o **interfaz WS**, que permiten que usuarios automáticos, esto es, otros sistemas, interactúen con el sistema.

2.1.2 Interfaces requeridos

Son aquellos interfaces provistos por otros sistemas, y con los que el sistema necesita interactuar para llevar a cabo determinadas operativas de negocio.

En este escenario, el sistema se comporta como un usuario automático, que consume el interfaz provisto por otro sistema. Por tanto, los interfaces requeridos únicamente pueden ser interfaces WS.

2.1.3 Puertos

Los puertos son los únicos puntos de acceso al ámbito del sistema, a través de los cuales es posible exponer interfaces provistos, o bien acceder a interfaces requeridos.

En este modelo, los puertos se entienden como *endpoints* ligados al protocolo empleado para interactuar con estos interfaces.

La especificación de los mecanismos de seguridad lógica que es necesario aplicar sobre los puertos y sobre la información intercambiada a través de ellos, forma parte de la **arquitectura de seguridad** del sistema.

2.1.4 Operativas de negocio

Las operativas de negocio codifican las reglas de negocio, que han sido definidas en los casos de uso del sistema, y que determinan cómo los datos que maneja el sistema pueden ser creados, almacenados, accedidos y modificados.

Las operativas de negocio se diseñan en un esquema basado en componentes, los cuales son unidades funcionales auto-contenidas y con un objetivo bien definido, que colaboran entre sí para realizar su trabajo.

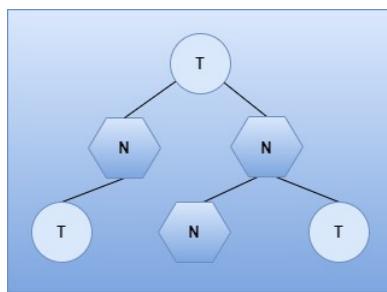


Ilustración 2-2 Componentes lógicos

A grandes rasgos, se pueden diferenciar dos tipos básicos de componente:

- **Componentes de negocio:** se especializan en la resolución de aspectos específicos del negocio, de forma que codifican y representan reglas de negocio.
- **Componentes técnicos:** no codifican reglas de negocio, sino que proporcionan funcionalidad meramente técnica, y son menos relevantes en el diseño que los componentes de negocio. Los componentes técnicos pueden ser implementados de forma aplicativa, o bien ser proporcionados por los distintos *frameworks* empleados. Tienen la característica de que pueden ser reemplazados en el futuro por nuevos componentes técnicos, de nuevas tecnologías, o de nuevos *frameworks*.

La especificación de los componentes necesarios para resolver las operativas de negocio, así como de las relaciones existentes entre ellos, determinan la **arquitectura lógica** del sistema.

Los componentes que conforman el sistema se ensamblan en forma de artefactos, que posteriormente serán desplegados sobre los nodos de un modelo físico, cuya especificación constituye la **arquitectura física** del sistema.

2.1.5 Datos

Los datos son el activo más importante del sistema, y su protección es el motivo fundamental que lleva a la definición de un ámbito limitado.

Físicamente, los datos se ubican en bases de datos u otros tipos de unidades o sistemas de almacenamiento, y siempre deben estar separados de los datos de otros sistemas, respetando el ámbito limitado de cada uno.

La especificación de la estructura y las políticas de gestión de los datos forma parte de la **arquitectura de datos**.

2.2 Descomposición de sistemas de información

En tiempo de diseño arquitectónico, un sistema de información se puede descomponer en varios **subsistemas** independientes, los cuales interactúan entre sí para proporcionar la funcionalidad requerida.

El sistema así concebido es, por tanto, de naturaleza **distribuida**, y no tiene entidad en sí mismo, tan solo es una abstracción compuesta por el conjunto de los subsistemas que lo conforman.

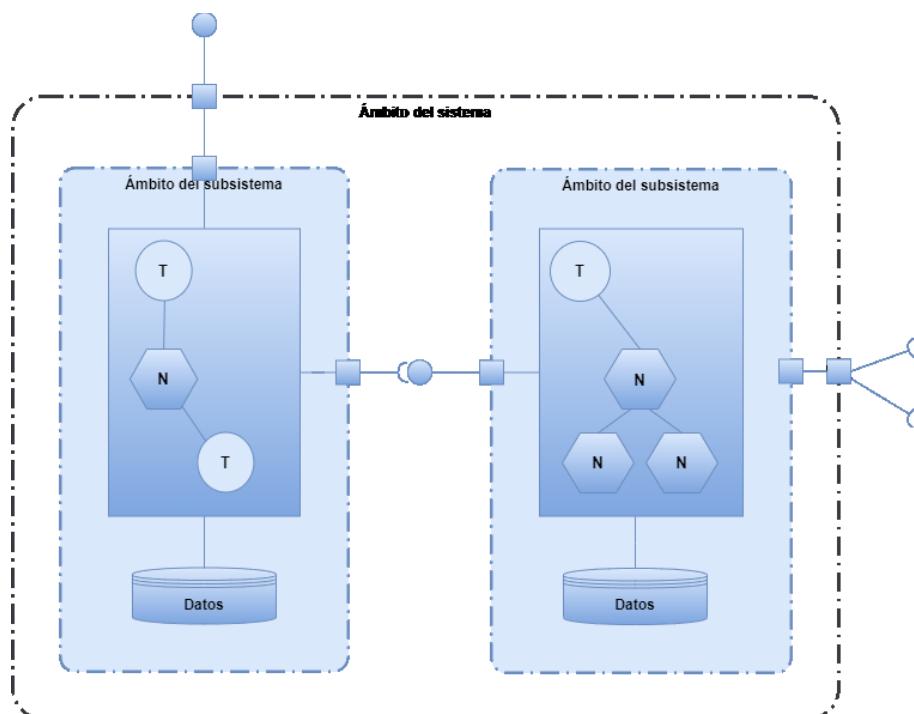


Ilustración 2-3 Descomposición del sistema en subsistemas

Norma AR-SIS-02

En tiempo de **diseño arquitectónico** de un sistema de información, éste puede concebirse como un sistema distribuido, compuesto a su vez por varios subsistemas independientes.

Esta descomposición del sistema en subsistemas debe realizarse de tal forma que se garantice una **alta cohesión** en la funcionalidad proporcionada por cada subsistema, y un **bajo acoplamiento** o grado de interdependencia entre ellos.

Cada uno de estos subsistemas debe seguir el mismo **modelo de sistema** especificado en la **Ilustración 2-1**. Por tanto, un subsistema posee un ámbito limitado, que protege los datos que maneja, permitiendo su acceso únicamente a través de las operativas de negocio reflejadas en sus interfaces provistos, los cuales son accesibles a través de puertos bien definidos.

Importante

El **diseño arquitectónico** tiene un carácter **especificativo**, es decir, establece requisitos técnicos que deberán ser tenidos en cuenta posteriormente, durante la construcción.

El **diseño detallado** completa el diseño arquitectónico, pero ya no tiene un carácter específico, sino **descriptivo**. Se confecciona en paralelo a la construcción, y constituye una referencia para facilitar el posterior mantenimiento.

2.3 Interacción entre sistemas

La interacción entre los distintos sistemas o subsistemas deberá realizarse de la forma más desacoplada posible. El objetivo que se persigue con ello es el de reducir las interdependencias entre ellos, que tanto dificultan su mantenimiento posterior, especialmente cuando los distintos sistemas o subsistemas evolucionan a velocidades diferentes.

El uso de servicios Web logra un desacoplamiento respecto a la tecnología empleada en la implementación de los sistemas o subsistemas que interactúan. Sin embargo, en este esquema de interacción sigue existiendo un relativo acoplamiento, determinado por el formato de los mensajes.

Para eliminarlo, se establece que las comunicaciones sean intermediadas por un middleware. Este middleware es capaz de procesar los mensajes y elaborar transformaciones de datos, antes de reenviarlos al subsistema destino, a la vez que soporta esquemas de interacción tanto síncronos como asíncronos.

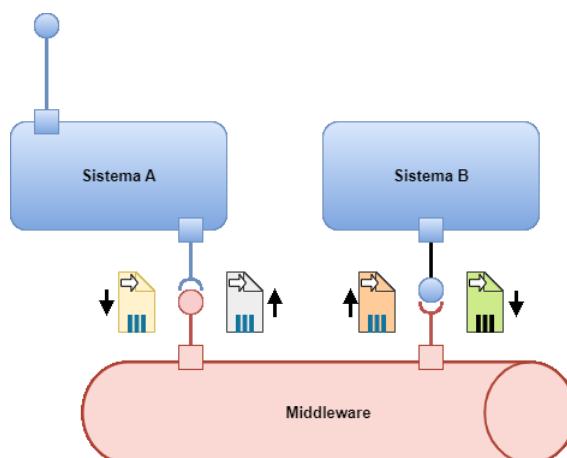


Ilustración 2-4 Interacciones entre sistemas

Norma AR-SIS-03

La **interacción** entre los distintos sistemas o subsistemas deberá realizarse de la forma más desacoplada posible, a través de **servicios Web** intermediados por un **middleware**.

2.4 Ámbito limitado del entorno de producción

Los distintos sistemas se encuentran (o se encontrarán) desplegados en el entorno de producción, al cual también define un **ámbito limitado**. Al fin y al cabo, el entorno de producción se puede entender como un gran sistema, compuesto a su vez por todos los sistemas desplegados dentro de éste.

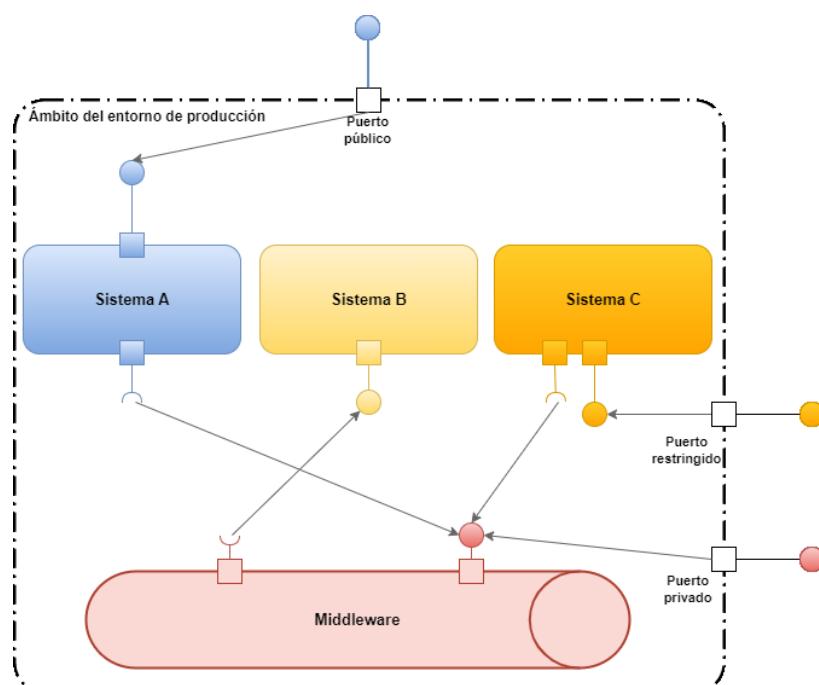


Ilustración 2-5 Ámbito limitado del entorno de producción

Norma AR-SIS-04

Los distintos sistemas o subsistemas desplegados en el entorno de producción podrán exponer sus interfaces provistas a través de los **puertos** existentes en dicho entorno:

- Puerto **público**: accesible al público desde Internet.
- Puerto **restringido**: accesible a determinadas Administraciones Públicas externas al Ministerio de Justicia, a través de la red SARA/TESTA.
- Puerto **privado**: accesible únicamente desde determinadas unidades del Ministerio de Justicia y/o sus organismos adscritos.

Para hacerlo, se establecen los **criterios** siguientes:

- Se expondrán por el puerto **público** todas aquellas interfaces Web que deban ser accesibles por usuarios humanos a través de Internet.
- Se expondrán por el puerto **restringido** todas aquellas interfaces Web que deban ser accesibles por usuarios humanos en el ámbito de otras Administraciones Públicas externas al Ministerio de Justicia, a través de la red SARA/TESTA.
- Se expondrán por el puerto **privado** todas aquellas interfaces Web que deban ser accesibles por usuarios humanos en el ámbito del Ministerio de Justicia y/o sus organismos adscritos.
- Se expondrán por el puerto **público** todas aquellas interfaces WS que deban ser accesibles por usuarios automáticos a través de Internet.
- Se expondrán por el puerto **restringido** todas aquellas interfaces WS que deban ser accesibles por usuarios automáticos en el ámbito de otras Administraciones Públicas externas al Ministerio de Justicia, a través de la red SARA/TESTA.
- Se expondrán por el puerto **privado** todas aquellas interfaces WS que deban ser accesibles por usuarios automáticos en el ámbito del Ministerio de Justicia y/o sus organismos adscritos.
- Un mismo interfaz no puede ser accesible a la vez a través de varios puertos.

2.5 Catálogo de estereotipos de subsistemas aplicativos

En el contexto de esta Arquitectura de Referencia, un sistema de información es una abstracción que representa la agrupación de uno o más **subsistemas aplicativos**, o aplicaciones.

Para facilitar y sistematizar el diseño arquitectónico, se determina el siguiente catálogo de estereotipos de subsistemas aplicativos.

Norma AR-SIS-05

Cada sistema de información deberá descomponerse en uno o más **subsistemas aplicativos**, que constituye la unidad básica y atómica de subsistema.

Cada subsistema aplicativo deberá disponer de un **estereotipo** específico, considerándose los siguientes:

- **Aplicación Web**, definido en la **Ilustración 2-6**
- **Fachada Web**, definido en la **Ilustración 2-7**
- **Aplicación de servicios Web**, definido en la **Ilustración 2-8**
- **Aplicación Híbrida**, definido en la **Ilustración 2-9**

2.5.1 Aplicación Web

Una aplicación Web se caracteriza por proveer un interfaz Web, siendo accesible a través de uno de los tres puertos del entorno de producción.

Dicho interfaz expone operativas de negocio, que implementan diferentes componentes lógicos, los cuales gestionan los datos que maneja el sistema.

Puede disponer de interfaces requeridos, pero en todo momento la interacción con los correspondientes interfaces provistos debe realizarse a través del middleware de integración.

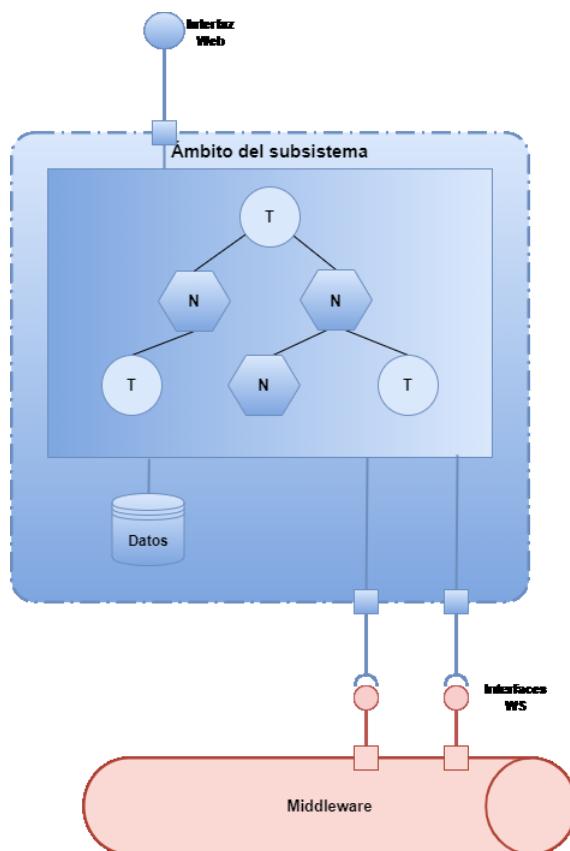


Ilustración 2-6 Estereotipo de Aplicación Web

2.5.2 Fachada Web

Una fachada Web se caracteriza por proveer un interfaz Web, accesible a través de uno de los tres puertos del entorno de producción.

Dicho interfaz expone operativas de negocio, que implementan diferentes componentes lógicos, los cuales las resuelven mediante la orquestación de llamadas a otros subsistemas del mismo sistema, a través de servicios Web, por lo que no maneja ni almacena datos propios.

Dispone de interfaces requeridos, que se corresponden con las operativas de negocio expuestas por otros sistemas o subsistemas.

En todo momento la interacción con los correspondientes interfaces provistos debe realizarse a través del middleware de integración.

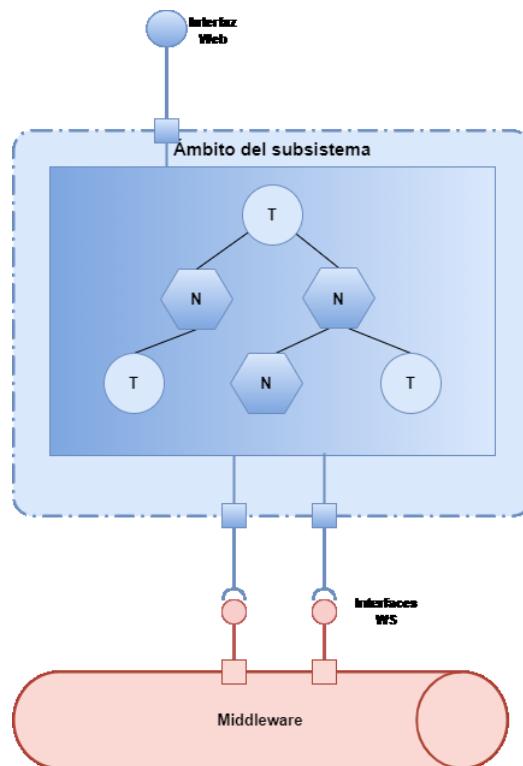


Ilustración 2-7 Estereotipo de Fachada Web

2.5.3 Aplicación de servicios Web

Una aplicación de servicios Web se caracteriza por proveer uno o más interfaces WS. Cada uno de ellos es accesible a través de uno de los tres puertos del entorno de producción, o bien permanece interno al ámbito del entorno de producción, esto es, no accesible desde el exterior. En todo caso, el acceso a dichos interfaces siempre ha de ser intermediado a través del middleware de integración.

Estos interfaces WS exponen operativas de negocio, que implementan diferentes componentes lógicos, los cuales gestionan los datos que maneja el sistema.

Puede disponer de interfaces requeridos, pero en todo momento la interacción con los correspondientes interfaces provistos debe realizarse a través del middleware de integración.

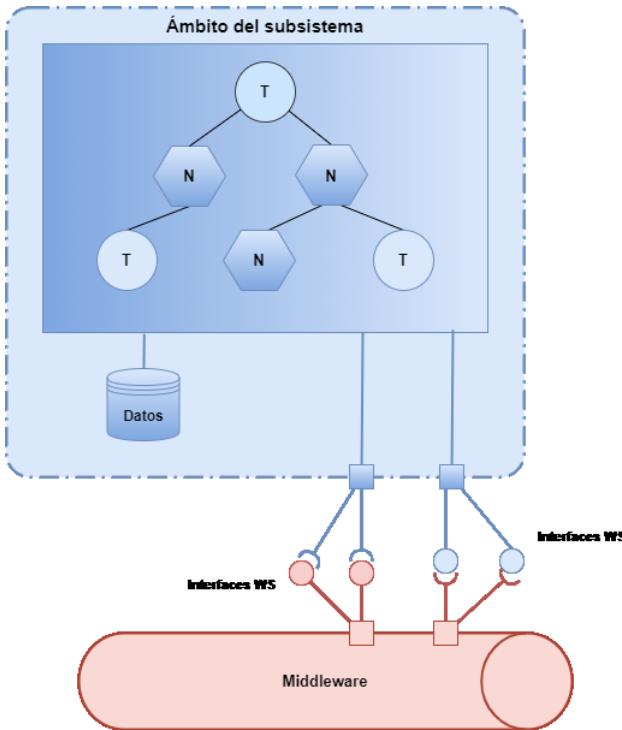


Ilustración 2-8 Estereotipo de Aplicación de servicios Web

2.5.4 Aplicación Híbrida

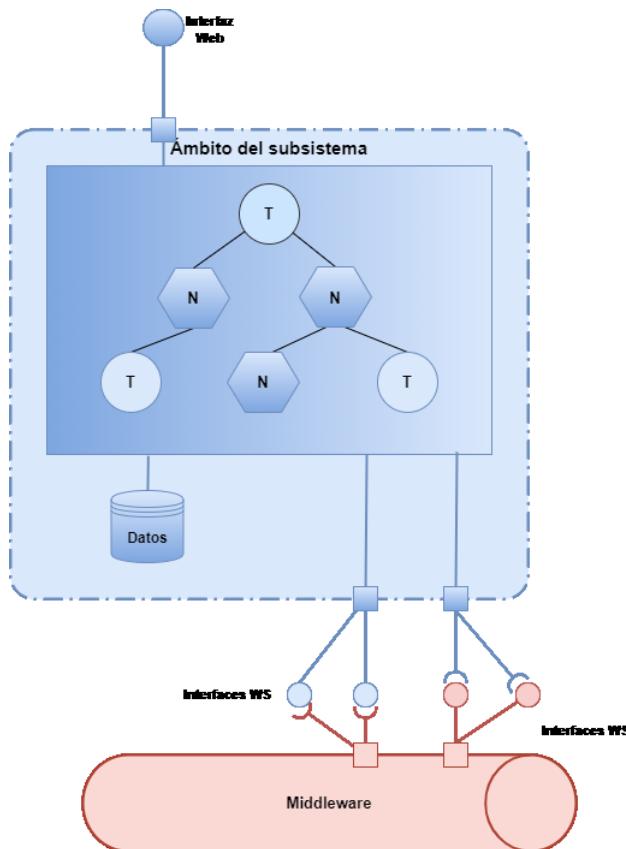
Una aplicación Híbrida es la combinación de una aplicación Web con una aplicación de servicios Web. Se caracteriza por proveer un interfaz Web, así como uno o más interfaces WS.

El interfaz Web es accesible a través de uno de los tres puertos del entorno de producción.

Cada uno de los interfaces WS es accesible a través de uno de los tres puertos del entorno de producción, o bien permanece interno al ámbito del entorno de producción, esto es, no accesible desde el exterior. En todo caso, el acceso a dichos interfaces WS siempre ha de ser intermediado, a través del middleware de integración.

Estos interfaces exponen operativas de negocio, que implementan diferentes componentes lógicos, los cuales gestionan los datos que maneja el sistema.

Puede disponer de interfaces requeridos, pero en todo momento la interacción con los correspondientes interfaces provistos debe realizarse a través del middleware de integración.


Ilustración 2-9 Estereotipo de Aplicación Híbrida

2.6 Estilos arquitecturales

La descomposición modular de un sistema de información en diferentes subsistemas reduce la complejidad funcional y técnica de cada uno de los subsistemas, lo cual facilita su mantenimiento, a la vez que el sistema en su conjunto puede soportar una mayor demanda haciendo un uso más razonable de la infraestructura.

Sin embargo, cuando un sistema de información se descompone en un elevado número de subsistemas, la problemática de la interacción entre ellos deriva en la aparición de una complejidad inesperada, ya que generalmente no se suele tener en cuenta en tiempo de diseño, debido a que aumentan los puntos de fallo, así como la complejidad del mantenimiento, de las pruebas integradas, del despliegue, del control de cambios, etcétera.

Esta complejidad se traduce en un incremento más que notable del coste de mantenimiento y operación del sistema en su conjunto, el cual podría asumirse siempre que se compense con algún beneficio.

Por estos motivos, se hace necesario establecer unos criterios para lograr un grado de descomposición modular razonable, adecuado y proporcionado al uso que va a tener el sistema.

2.6.1 Arquitectura monolítica modular

El estilo arquitectural monolítico modular se caracteriza por considerar el sistema como un monolito, esto es, como una única aplicación.

Sin embargo, el esfuerzo de descomposición se centra en su arquitectura lógica, a través de un proceso de diseño en el que se definen claramente diferentes componentes de negocio de alto nivel (módulos), con una alta cohesión funcional y un bajo acoplamiento, los cuales pueden apoyarse en otros componentes de carácter auxiliar.

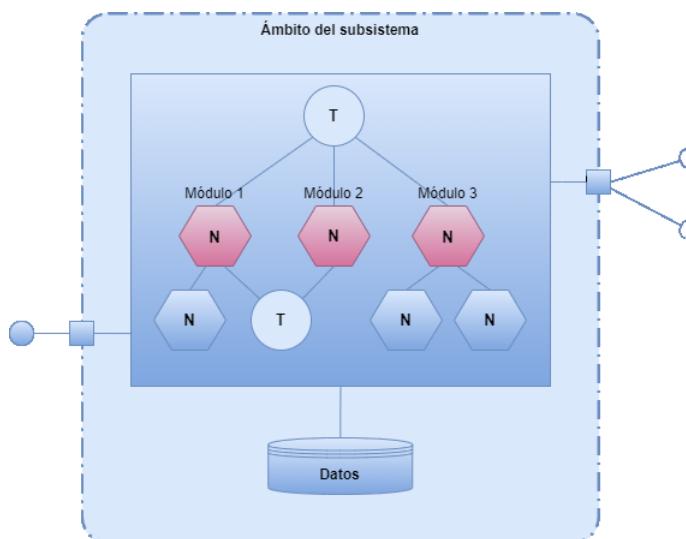


Ilustración 2-10 Arquitectura monolítica modular

Norma AR-SIS-06

Se recomienda el uso del estilo arquitectural **monolítico modular** en el diseño arquitectónico de sistemas de información:

- Que posean una **baja complejidad funcional**
- Que posean un **bajo valor de negocio**
- Que posean una **carga estimada notablemente baja**
- Aquellos para los que otros estilos arquitecturales, aun siéndoles de aplicación, resulten inabordables

El **diseño arquitectónico** del sistema, bajo este estilo arquitectural, deberá concebirse como una única aplicación Web, aplicación de servicios Web o aplicación Híbrida.

En su arquitectura lógica, deberán definirse los distintos módulos contemplados, y sus atribuciones funcionales.

2.6.2 Arquitectura de miniservicios

El estilo arquitectural de miniservicios se caracteriza por realizar una descomposición del sistema en varios subsistemas independientes, de tal manera que cada uno de ellos pueda manejar sus propios datos sin que sea necesario la realización de transacciones distribuidas para garantizar la integridad referencial de la información manejada por el conjunto.

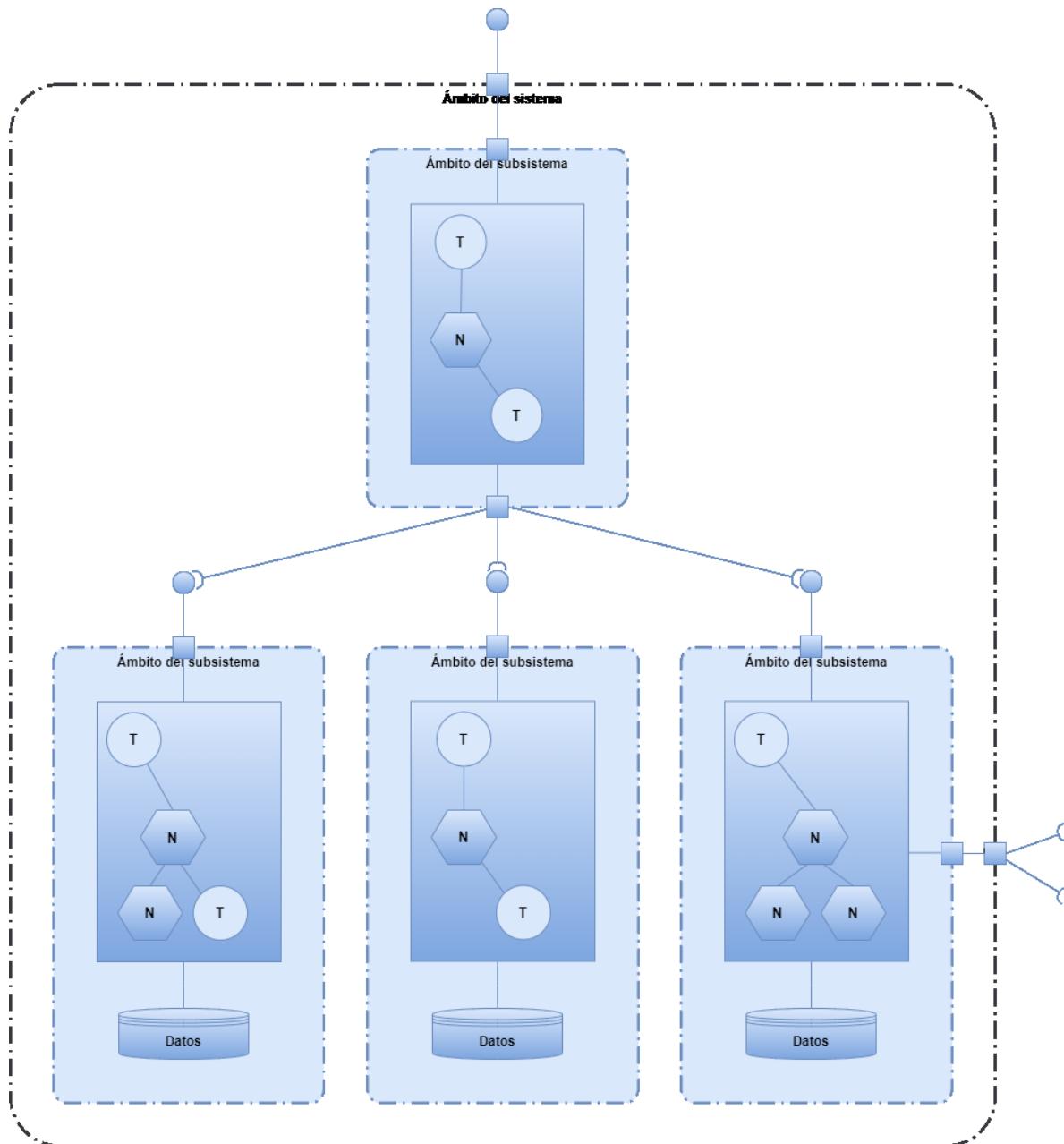


Ilustración 2-11 Arquitectura de miniservicios

El sistema, entendido como la unión de todos los subsistemas que lo conforman, maneja datos pertenecientes al mismo dominio, esto es, el dominio del sistema, el cual ha sido previamente definido en tiempo de análisis funcional.

El diseño arquitectónico del sistema, realizado a continuación del análisis funcional, está guiado completamente por dicho dominio, siguiendo un proceso mediante el cual éste es dividido en subdominios (agrupaciones de entidades) independientes, de tal forma que cada uno de ellos es manejado por un miniservicio.

A continuación, se establecen las atribuciones funcionales de cada uno de ellos en función de las entidades que manejan, y se publica su funcionalidad mediante servicios Web.

Cada miniservicio es susceptible de ser desplegado sobre su propia infraestructura, la cual posee una capacidad determinada en función de su demanda estimada.

Norma AR-SIS-07

Se recomienda el uso del estilo arquitectural de **miniservicios** en el diseño arquitectónico de sistemas de información:

- Que posean una **alta complejidad funcional**
- Que posean una **carga estimada notablemente alta**

El proceso de diseño arquitectónico deberá estar guiado por los datos, para lo cual se requiere disponer de un **diagrama entidad-relación extendido**, con ámbito del sistema completo, obtenido en tiempo de análisis funcional.

En primer lugar, sobre dicho diagrama se identificarán los distintos subdominios o **agrupaciones** de entidades, debiéndose tener en cuenta que:

- No debe existir ningún caso de uso que requiera la realización de consultas que crucen datos presentes en entidades de distintas agrupaciones.
- No debe existir ningún caso de uso que requiera la actualización de datos presentes en entidades de distintas agrupaciones en la misma transacción.

En caso de que no se cumplan las condiciones anteriores, la división en agrupaciones no será correcta, y deberá replantearse. Para ello, es útil tener en cuenta lo siguiente:

- Dos o más agrupaciones de entidades podrían unirse en una sola, bajo el mismo miniservicio.
- En ocasiones, varios miniservicios pueden tener una visión diferente de una misma entidad. Esto es, una entidad del dominio del sistema (un concepto) puede modelarse como entidades diferentes en distintos subdominios (distintas perspectivas o formas de ver el mismo concepto).

A continuación, se definirá un miniservicio para cada agrupación identificada, a la cual se le atribuirá la funcionalidad relacionada con el manejo de las entidades de su agrupación.

El **diseño arquitectónico** del sistema bajo este estilo arquitectural, deberá concebirse como una aplicación de servicios Web para cada uno de los miniservicios identificados, así como una fachada Web para cada interfaz Web provisto por el sistema.

2.6.3 Arquitectura de microservicios

El estilo arquitectural de microservicios se caracteriza por llevar al extremo la descomposición del sistema en subsistemas independientes, concibiendo tantos microservicios como sea necesario, de tal manera que cada uno de ellos maneje una única entidad (y sus entidades débiles, en caso de existir) o proceso del sistema.

De esta manera, las atribuciones funcionales de un microservicio son mínimas, pero la problemática de su interacción añade una extraordinaria complejidad, requiriéndose un *middleware* específico de publicación-suscripción para que los diferentes microservicios puedan publicar eventos, o bien suscribirse para su recepción, y de esta manera informar y ser informados de los cambios de estado del dominio que les puedan afectar.

Al igual que en el caso anterior, cada microservicio es desplegado sobre su propia infraestructura, la cual puede escalarse o desescalarse dinámicamente en función de su demanda.

Como puede observarse, el estilo arquitectural de microservicios es un modelo más complejo y con un coste de mantenimiento considerablemente más alto que el de otras alternativas, el cual únicamente podría compensarse en esquemas donde la infraestructura se proveyese en *cloud* y se facturase por su uso, no siendo este el caso de la DSDD.

Norma AR-SIS-08

Con carácter general, no se recomienda el uso del estilo arquitectural de **microservicios** en el diseño arquitectónico de los sistemas de información en el ámbito de la DSDD.

3 Arquitectura física

La arquitectura física de un subsistema aplicativo se articula en torno a una unidad básica de infraestructura, entendida como el conjunto mínimo de componentes físicos necesarios para dar servicio en alta disponibilidad, así como por el dimensionamiento de la capacidad requerida por los mismos.

Nótese que varios subsistemas aplicativos diferentes podrían compartir los mismos componentes físicos, si su capacidad estimada o medida así lo permite.

3.1 Componentes físicos

La arquitectura física describe los subsistemas aplicativos a partir de los componentes físicos que los conforman, y de las relaciones existentes entre ellos, utilizando un nivel de abstracción que resulte razonable para su propósito.

Se distinguen los componentes físicos que se describen a continuación.

3.1.1 Nodos

Los nodos representan entornos de ejecución, ya sean hardware o software, en los que es posible desplegar artefactos. Serán considerados nodos: las distintas máquinas, ya sean físicas o virtuales, firewalls, rúters y otros elementos de red, pero también los sistemas operativos, servidores Web, servidores de aplicaciones, sistemas gestores de bases de datos, etc.

Un nodo puede contener, a su vez, a otros nodos. Por ejemplo, una máquina física contiene a una máquina virtual, sobre la que se ejecuta el sistema operativo, que gestiona la ejecución de una máquina virtual Java, que ejecuta un servidor de aplicaciones JEE que, finalmente, gestiona la ejecución de una aplicación Web.

En este ejemplo, la aplicación Web es un artefacto, y el resto de los elementos conforman una estructura de nodos anidados.

En ocasiones es útil representar estas estructuras de nodos anidados con todo su nivel de detalle, especificando todos y cada uno de ellos. En otras, sin embargo, se pueden obviar los detalles, y trabajar a un nivel de abstracción más alto. En el ámbito de esta Arquitectura de Referencia, se establece un nivel de abstracción donde se obvian los detalles del hardware físico o virtual y del sistema operativo, prestando únicamente atención al nivel de plataforma.

En este nivel de abstracción, los nodos pueden clasificarse, a grandes rasgos, en tres bloques fundamentales: servidores, elementos de red, y sistemas o subsistemas externos.

3.1.1.1 Servidores

Se contemplan los tipos de nodo servidor que se indican a continuación.

Norma AR-FIS-01

Los tipos de nodo **servidor** que se contemplan como parte de la arquitectura física de un subsistema aplicativo quedan especificados por la **Tabla 3-1**, **Tabla 3-2**, **Tabla 3-3** y **Tabla 3-4**.

La versión vigente en cada momento del software base indicado para cada uno de ellos queda determinada por la **Política de actualización de la línea base de plataformas**.

Esta política tiene por objeto colaborar para evitar la obsolescencia tecnológica de los sistemas de información, liberando versiones frecuentes que mantengan actualizada dicha línea base.

Servidor	Servidor frontal externo
Software base	Apache HTTP Server
Nivel	Primer nivel
Subred	DMZ externa
Descripción	<p>Su objetivo es la recepción y tratamiento de peticiones Web provenientes de los usuarios a través de Internet.</p> <p>Este tratamiento queda limitado a servir recursos Web estáticos, o bien despachar peticiones dinámicas a los servidores de aplicaciones existentes en el segundo nivel, balanceando entre sus instancias.</p>

Tabla 3-1 Especificación de servidor frontal externo

Servidor	Servidor frontal interno
Software base	Apache HTTP Server
Nivel	Primer nivel
Subred	DMZ interna
Descripción	<p>Su objetivo es la recepción y tratamiento de peticiones Web provenientes de los usuarios a través de SARA/TESTA o de la Intranet.</p> <p>Este tratamiento queda limitado a servir recursos Web estáticos, o bien despachar peticiones dinámicas a los servidores de aplicaciones existentes en el segundo nivel, balanceando entre sus instancias.</p>

Tabla 3-2 Especificación de servidor frontal interno

Servidor	Servidor de aplicaciones
Software base	Red Hat JBoss Enterprise Application Platform Open JDK
Nivel	Segundo nivel
Subred	Subred de servidores
Descripción	Su objetivo es la recepción y tratamiento de peticiones dinámicas provenientes de sus correspondientes servidores frontales. Se encuentran configurados en clúster.

Tabla 3-3 Especificación de servidor de aplicaciones

Servidor	Servidor de base de datos
Software base	Oracle Database
Nivel	Tercer nivel
Subred	Subred de datos
Descripción	Su objetivo es permitir el almacenamiento, modificación y extracción de datos estructurados en el esquema del subsistema aplicativo, a través de peticiones provenientes de los servidores de aplicaciones.

Tabla 3-4 Especificación de servidor de base de datos

3.1.1.2 Elementos de red

Los elementos de red, tales como rúters, firewalls, balanceadores, proxis inversos, etc. permiten articular las comunicaciones de los nodos servidores entre sí, o con Internet, la Intranet o las redes SARA y TESTA. Las redes y subredes, habitualmente, suelen considerarse también nodos.

3.1.1.3 Sistemas externos

Los sistemas externos son aquellos componentes físicos con los que el subsistema aplicativo interactúa, sin formar parte estructural de él.

Norma AR-FIS-02

Se consideran **sistemas externos** los **servicios corporativos**, otros **sistemas de información**, de uso compartido o no, y ya sean provistos por la SGAD u otros organismos, o por la propia DSDD, los **terminales de usuario** y el **middleware** de integración.

Ejemplos de servicios corporativos son Active Directory, el servidor de correo, unidades de almacenamiento NFS, etc.

3.1.2 Conexiones

En el contexto de una arquitectura física, las conexiones representan los caminos de comunicación posibles entre nodos.

3.1.3 Artefactos

Los artefactos son los elementos físicos que son desplegados en los distintos nodos que conforman la arquitectura física del sistema o subsistema, y que se han generado a partir de la compilación y el ensamblado del código fuente.

Norma AR-FIS-03

Los **artefactos** son elementos físicos derivados de la compilación y ensamblado del código fuente.

Los **tipos** de artefacto que se contemplan como parte de la arquitectura física de un subsistema aplicativo son:

- **Archivo WAR:** formato estándar de archivo de Aplicación Web JEE, que ensambla los diferentes componentes de negocio, así como las interfaces Web o WS. Se despliega en cada uno de los servidores de aplicaciones.
- **Scripts de base de datos:** conjunto de scripts con sentencias de definición o modificación de base de datos, o de manipulación de datos, empaquetados en forma de archivo ZIP. Se despliega (ejecuta) en el servidor de base de datos.
- **Recursos estáticos Web aplicativos:** conjunto de recursos de presentación específicos de un determinado subsistema aplicativo, que son requeridos por sus interfaces Web, y que no son generados dinámicamente, empaquetados en forma de archivo ZIP. Se despliega en los servidores frontales, externos o internos, según corresponda.
- **Recursos estáticos Web compartidos:** conjunto de recursos de presentación compartidos por diferentes subsistemas aplicativos que basan la apariencia de sus interfaces Web en una misma maqueta, y que no son generados dinámicamente, empaquetados en forma de archivo ZIP. Se despliega en los servidores frontales, externos o internos, según corresponda.
- **Recursos de configuración compartida:** conjunto de recursos de configuración compartidos por diferentes subsistemas aplicativos, empaquetados en forma de archivo ZIP. Se despliega en cada uno de los servidores de aplicaciones.
- **Recursos de configuración administrada aplicativa:** conjunto de recursos de configuración específicos de un determinado subsistema aplicativo, empaquetados en forma de archivo ZIP. Se despliega en cada uno de los servidores de aplicaciones.

Importante

Las librerías son artefactos JAR que permiten la reutilización de código, pero no son directamente desplegables en el entorno, estando siempre contenidas en los artefactos WAR que las requieran. No forman parte, por tanto, de la arquitectura física del subsistema aplicativo.

3.2 Determinación del diagrama de red

La especificación de la arquitectura física de un subsistema aplicativo incluye la determinación de los componentes físicos requeridos por éste, en forma de diagrama de red, independientemente de que algunos de ellos sean compartidos por otros subsistemas aplicativos.

Para obtenerla, se deberá seguir un método sistemático, partiendo de los elementos presentes en el modelo de sistema de su estereotipo correspondiente, para completarla a continuación con los nodos faltantes.

Norma AR-FIS-04

Como parte de la **arquitectura física** de un **subsistema aplicativo**, se debe determinar su **diagrama de red**.

Este diagrama debe estar organizado en **tres niveles**, debiéndose componer a partir de la **unión de los fragmentos** de diagrama de red normalizados asociados a los distintos elementos presentes en el modelo de subsistema aplicativo del estereotipo correspondiente.

Estos fragmentos normalizados se encuentran especificados por la **Ilustración 3-1, Ilustración 3-2, Ilustración 3-3, Ilustración 3-4, Ilustración 3-5 e Ilustración 3-6**.

El diagrama resultante debe completarse con los posibles nodos faltantes.

En todo caso, habrá de especificarse en el diagrama:

- Servidores.
- Redes y subredes.
- Firewalls y balanceadores.
- Sistemas externos con los que el subsistema aplicativo deba integrarse.
- Middleware de integración.
- Terminales de usuario.

3.2.1 Primer nivel

El primer nivel del diagrama de red quedará determinado a partir de los interfaces provistos por el subsistema aplicativo, según el modelo de su estereotipo correspondiente.

3.2.1.1 Interfaz Web accesible desde el puerto público

Los subsistemas aplicativos que provean **un interfaz Web** accesible desde el **puerto público** deberán disponer de los siguientes elementos en el primer nivel de su arquitectura física.

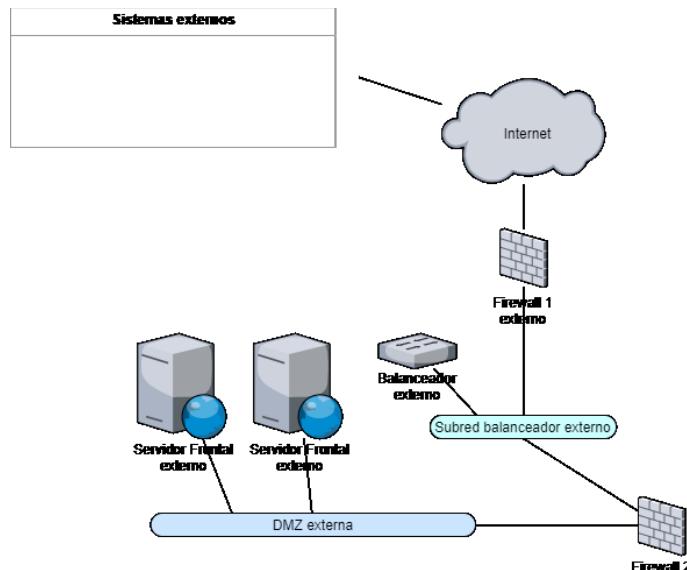


Ilustración 3-1 Elementos en primer nivel para subsistemas aplicativos con un interfaz Web accesible desde el puerto público

Las peticiones, provenientes de sistemas externos a través de Internet, son filtradas por un firewall y balanceadas entre dos servidores frontales externos, ubicados en la DMZ externa. Estos servidores sirven contenido Web estático, o bien despachan las peticiones al segundo nivel, que se encuentra separado del primero por un segundo firewall.

3.2.1.2 Interfaz WS accesible desde el puerto público

Los subsistemas aplicativos que provean **uno o más interfaces WS** accesibles desde el **puerto público** deberán disponer de los siguientes elementos en el primer nivel de su arquitectura física.

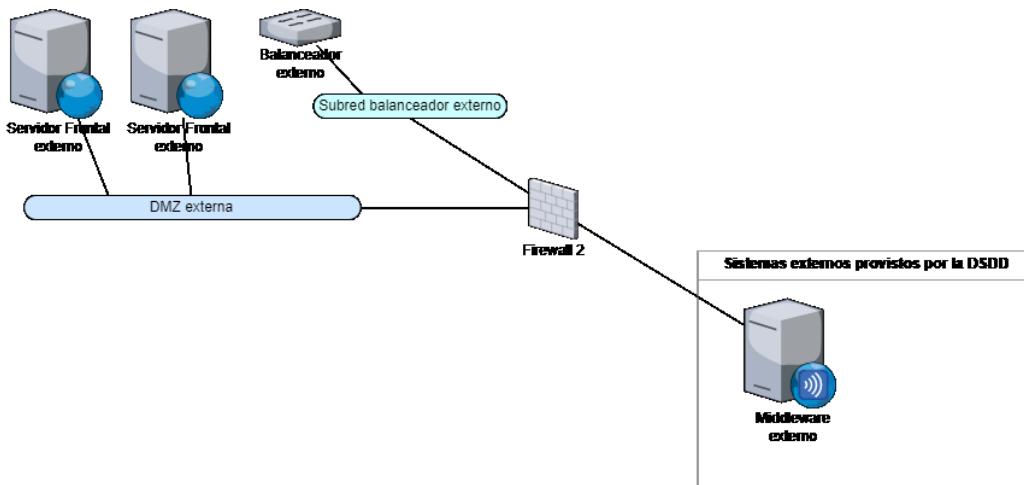


Ilustración 3-2 Elementos en primer nivel para subsistemas aplicativos con interfaces WS accesibles desde el puerto público

Las peticiones, provenientes del middleware externo, son balanceadas entre dos servidores frontales externos, ubicados en la DMZ externa. Estos servidores despachan las peticiones al segundo nivel, que se encuentra separado del primero por un segundo firewall.

3.2.1.3 Interfaz Web accesible desde el puerto restringido o privado

Los subsistemas aplicativos que provean **un interfaz Web accesible desde el puerto restringido o el puerto privado** deberán disponer de los siguientes elementos en el primer nivel de su arquitectura física.

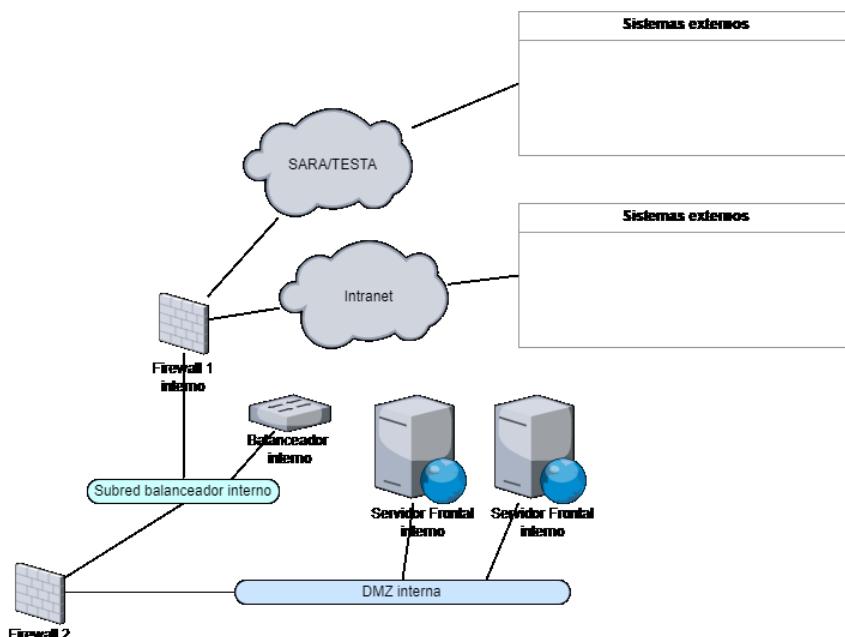


Ilustración 3-3 Elementos en primer nivel para subsistemas aplicativos con un interfaz Web accesible desde el puerto restringido o privado

Las peticiones, provenientes de la Intranet o desde las redes SARA/TESTA, son filtradas por un firewall y balanceadas entre dos servidores frontales internos, ubicados en la DMZ interna. Estos servidores sirven contenido Web estático, o bien despachan las peticiones al segundo nivel, que se encuentra separado del primero por un segundo firewall.

3.2.1.4 Interfaz WS accesible desde el puerto restringido o privado

Los subsistemas aplicativos que provean **uno o más interfaces WS** accesibles desde el **puerto restringido o el puerto privado**, deberán disponer de los siguientes elementos en el primer nivel de su arquitectura física.

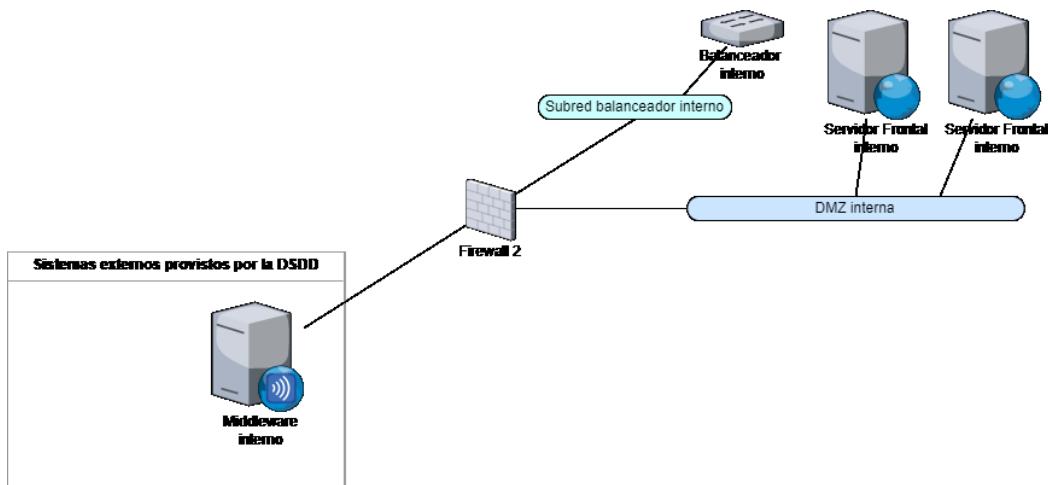
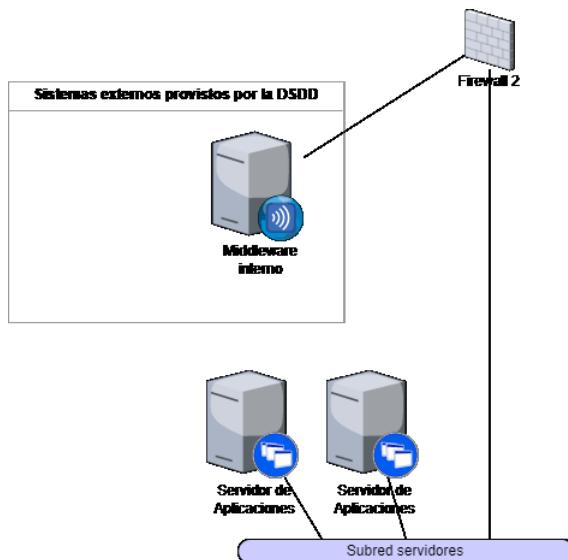


Ilustración 3-4 Elementos en primer nivel para subsistemas aplicativos con interfaces WS accesibles desde el puerto restringido o privado

Las peticiones, provenientes del middleware interno, son balanceadas entre dos servidores frontales internos, ubicados en la DMZ interna. Estos servidores despachan las peticiones al segundo nivel, que se encuentra separado del primero por un segundo firewall.

3.2.2 Segundo nivel

El segundo nivel del diagrama de red tiene carácter de general para cualquier estereotipo de subsistema aplicativo. No obstante, deberá completarse con los nodos faltantes en función de los sistemas o subsistemas externos con los que interactúe el subsistema aplicativo.

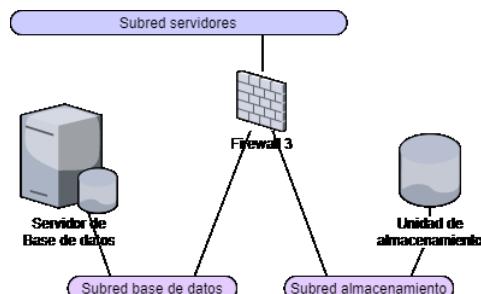

Ilustración 3-5 Elementos en el segundo nivel

Las peticiones, provenientes de los servidores frontales en la DMZ, son atendidas por dos servidores de aplicaciones en el segundo nivel. Desde ellos, y a través del middleware interno, es posible acceder a los interfaces WS requeridos por el subsistema aplicativo, y que serán provistos por otros sistemas o subsistemas externos.

Éstos deberán estar configurados en clúster siempre que el subsistema aplicativo provea un interfaz Web, no siendo necesario en caso contrario.

3.2.3 Tercer nivel

El tercer nivel del diagrama de red será de aplicación en los subsistemas aplicativos que manejen datos persistentes.


Ilustración 3-6 Elementos en el tercer nivel

Las peticiones, provenientes de los servidores de aplicaciones en el segundo nivel, serán atendidas por la base de datos del subsistema aplicativo, la cual se encuentra separada del segundo nivel por un tercer firewall.

También se ubicarán en la subred de datos, en caso de ser necesario, las unidades de almacenamiento compartidas.

3.3 Dimensionamiento de la capacidad

La especificación de la arquitectura física de un subsistema aplicativo debe incluir el dimensionamiento de la capacidad requerida por éste.

3.3.1 Capacidad de las unidades de almacenamiento

Deberá determinarse la capacidad de las unidades de almacenamiento compartido manejadas por el subsistema aplicativo como se indica a continuación.

Norma AR-FIS-05

Deberá determinarse el **volumen inicial** de los datos almacenados en cada una de las unidades de almacenamiento compartido que maneje el subsistema aplicativo, proveniente de la migración de datos inicial, así como el **incremento anual** de dicho volumen durante el funcionamiento ordinario del mismo.

Podrán establecerse **alarmas** de monitorización asociadas a las cuotas de almacenamiento obtenidas.

3.3.2 Capacidad de la base de datos

Deberá determinarse la capacidad del esquema de base de datos que maneje el subsistema aplicativo como se indica a continuación.

Norma AR-FIS-06

Deberá determinarse el **volumen inicial** de los datos almacenados en el esquema de base de datos que maneje el subsistema aplicativo, proveniente de la migración de datos inicial, así como el **incremento anual** de dicho volumen durante el funcionamiento ordinario del mismo.

Podrán establecerse **alarmas** de monitorización asociadas a las cuotas de almacenamiento obtenidas.

3.3.3 Tiempo máximo de respuesta

Deberá determinarse el tiempo máximo de respuesta en peticiones dirigidas a cada uno de los interfaces provistos por el subsistema aplicativo como se indica a continuación.

Norma AR-FIS-07

Deberá determinarse el **tiempo máximo** de respuesta en peticiones dirigidas a cada uno de los interfaces provistos por el subsistema aplicativo, a partir del cual

deberá suponerse que el mismo se encuentra sobrecargado. En todo caso, el tiempo máximo de respuesta deberá ser inferior a 30 segundos.

Podrán establecerse **alarmas** de monitorización asociadas a los tiempos obtenidos.

3.3.4 Distribución de carga

Deberá determinarse la distribución de carga soportada por cada uno de los interfaces provistos por el subsistema aplicativo como se indica a continuación.

Norma AR-FIS-08

Deberá determinarse la **distribución horaria** de la carga por cada uno de los interfaces provistos por el subsistema aplicativo, en peticiones por hora.

Asimismo, deberá determinarse la **carga-pico máxima**, entendida como el valor máximo de la carga agregada proveniente del conjunto de interfaces provistos por el subsistema aplicativo.

Deberá asegurarse, mediante la realización de **pruebas de carga** previas al paso a producción, que el subsistema aplicativo efectivamente puede asumir la carga-pico máxima determinada, durante la franja o franjas horarias consecutivas en que se prevea su ocurrencia.

3.3.5 Tamaño del clúster de servidores de aplicaciones

La unidad básica de infraestructura contempla un clúster compuesto por un mínimo de dos servidores de aplicaciones para un determinado subsistema aplicativo. Sin embargo, dependiendo de la carga determinada para el subsistema aplicativo, la capacidad ofrecida por dicho clúster puede considerarse excesiva, adecuada, o bien insuficiente.

Norma AR-FIS-09

Deberá determinarse el **tamaño del clúster** de servidores de aplicaciones como el redondeo del cociente entre la carga-pico máxima determinada para el subsistema aplicativo, y la carga que un nodo del clúster es capaz de asimilar en condiciones normales¹.

Si el tamaño obtenido fuese inferior a 2, el clúster deberá estar conformado por dos nodos servidor de aplicaciones, y podrá compartirse con otros subsistemas aplicativos hasta agotar su capacidad.

¹ Este valor puede estimarse en 50.000 peticiones/hora, suponiendo que una petición tarda entre 2 y 2,5 segundos, y que un nodo del clúster de servidores de aplicaciones dispone de un pool de 30 hilos para atender peticiones provenientes de los servidores frontales.

3.3.6 Disponibilidad

Deberá determinarse el tiempo máximo de indisponibilidad asumible del subsistema aplicativo, en función del nivel asociado a la dimensión Disponibilidad en el análisis de impacto que establece el Esquema Nacional de Seguridad, como se indica a continuación.

Nivel Disponibilidad ENS	Tiempo máximo de indisponibilidad
Alto	Inferior a 4 horas
Medio	Entre 4 horas y 1 día
Bajo	Entre 1 y 5 días
No aplica	Superior a 5 días

Tabla 3-5 Tiempos de indisponibilidad

Norma AR-FIS-10

Deberá determinarse **tiempo máximo de indisponibilidad** asumible por el subsistema aplicativo a partir del nivel asociado a la dimensión Disponibilidad, que se obtiene en la realización del análisis de impacto que impone el [Esquema Nacional de Seguridad](#), de acuerdo a la **Tabla 3-5**.

Adicionalmente, deberá determinarse el **volumen máximo de información** que sería asumible perder.

3.3.7 Revisión del dimensionamiento de la capacidad

Tras la puesta en producción de un subsistema aplicativo, deberá monitorizarse la capacidad real demandada por el mismo para mantener actualizado el dimensionamiento de su capacidad.

Norma AR-FIS-11

Tras la puesta en producción de un subsistema aplicativo, deberá **monitorizarse** la capacidad real demandada por el mismo, actualizando, si fuese necesario, el dimensionamiento de la capacidad realizado en su diseño arquitectónico, con medidas reales.

4 Arquitectura de integración

La arquitectura de integración define las normas a seguir, y los mecanismos a emplear, en la integración de sistemas o subsistemas que colaboren entre sí, de tal forma que, por un lado, se facilite la interoperabilidad y la reutilización de recursos, y, por otro, se reduzcan las interdependencias entre ellos.

Este último aspecto es especialmente relevante. Cuando se permite que dos o más sistemas o subsistemas se integren, se obtienen una serie de beneficios, pero, a cambio, se está introduciendo un acoplamiento entre ellos: cada uno depende de los demás en uno o más puntos.

Si un sistema o subsistema evoluciona a lo largo del tiempo, en su mantenimiento, es posible que produzca un impacto en los sistemas o subsistemas que dependan de él. Este impacto es a veces difícil de estimar o incluso de predecir.

Uno de los objetivos fundamentales de la arquitectura de integración es reducir este acoplamiento al mínimo. El uso de servicios Web logra un desacoplamiento respecto a las tecnologías empleadas, pero aún sigue quedando cierto grado de acoplamiento, determinado por el formato de los mensajes intercambiados.

Para eliminarlo, se establece que las comunicaciones entre servicios Web sean intermediadas por un middleware. Este middleware es capaz de procesar los mensajes y elaborar transformaciones de datos, antes de reenviarlos al subsistema destino.

4.1 Middleware de integración

A nivel arquitectural, se define la utilización de un middleware de integración, que hará uso del patrón de arquitectura *Enterprise Service Bus*, en el contexto de una arquitectura orientada a servicios, o SOA.

Este middleware permitirá que los sistemas o subsistemas que necesitan integrarse, puedan abstraerse de la lógica de integración necesaria, al proveer una infraestructura que actúa como intermediario y que permite, entre otras funciones:

- Procesamiento de mensajes intercambiados entre sistemas
- Transformaciones de datos o del formato de los mensajes
- Direcciónamiento de mensajes
- Manejo de diferentes protocolos para el envío de mensajes

El middleware de integración constituye un sistema en sí mismo. Por tanto, dispone de una arquitectura física y una arquitectura lógica propias, que se representan a continuación.

Existirán, en realidad, dos instancias diferenciadas y separadas del middleware de integración: la externa y la interna. Esta separación obedece a motivos de seguridad. De este modo, ante un posible ataque desde Internet, podría eventualmente

detenerse el tráfico hacia el middleware externo, afectando por tanto a los sistemas de información accesibles desde Internet a través de servicios Web. Sin embargo, el resto de sistemas de información no se verían afectados.

Norma AR-INT-01

Por motivos de seguridad, se establece la existencia de dos instancias diferenciadas y separadas del middleware de integración.

El middleware **externo**, será accesible únicamente a través del puerto **público** del entorno, mientras que el **middleware interno** únicamente lo será desde los puertos **restringido y privado**.

El middleware **externo** intermediará el **acceso** a todos los **interfaces WS** que deban estar disponibles a través del **puerto público**.

El middleware **interno** intermediará el **acceso** a todos los **interfaces WS** que deban estar disponibles a través del **puerto restringido** o del **puerto privado**, así como el acceso a todos los interfaces WS **internos**, esto es, no accesibles desde ninguno de los puertos del entorno.

4.1.1 Arquitectura física

Tanto en el caso del middleware externo como del interno, las peticiones son filtradas por un firewall y balanceadas entre dos servidores frontales, ubicados en la DMZ.

Estos servidores despachan las peticiones a dos servidores de integración, ubicados en la subred de servidores y separados por un segundo nivel de firewall.

Norma AR-INT-02

La arquitectura física del middleware externo se especifica en la **Ilustración 4-1**, mientras que la del middleware interno se especifica en la **Ilustración 4-2**.

Los **servidores de integración** que se contemplan en ambos casos quedan especificados por la **Tabla 4-1**.

La versión vigente en cada momento del software base indicado queda determinada por la **Política de actualización de la línea base de plataformas**.

Esta política de actualización tiene por objeto colaborar para evitar la obsolescencia tecnológica de los sistemas de información, liberando versiones frecuentes que mantengan actualizada dicha línea base.

Los nodos de tipo Servidor de integración quedan especificados de la siguiente forma:

Servidor	Servidor de integración
Software base	Oracle Service Bus
Nivel	Segundo nivel

Subred	Subred de servidores
Descripción	Facilitan la integración mediante el procesamiento, transformación y enrutamiento de los mensajes intercambiados entre los diferentes sistemas o subsistemas, así como el uso de distintos protocolos de comunicaciones. De esta manera, se reduce el acoplamiento entre los sistemas que interactúan

Tabla 4-1 Especificación de servidor de integración

La arquitectura física del middleware externo es la siguiente:

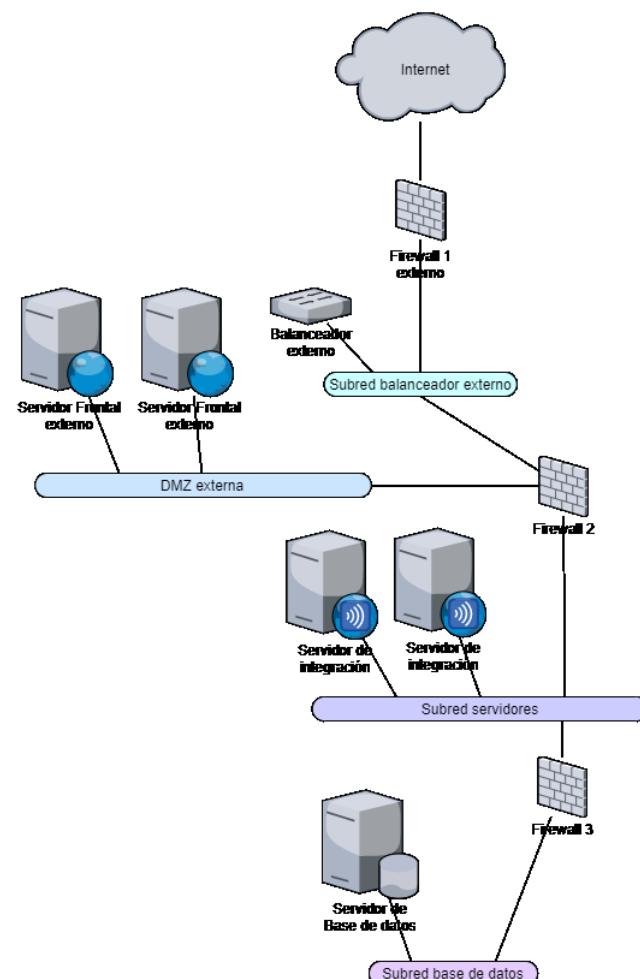


Ilustración 4-1 Arquitectura física del middleware externo

La arquitectura física del middleware interno es la siguiente:

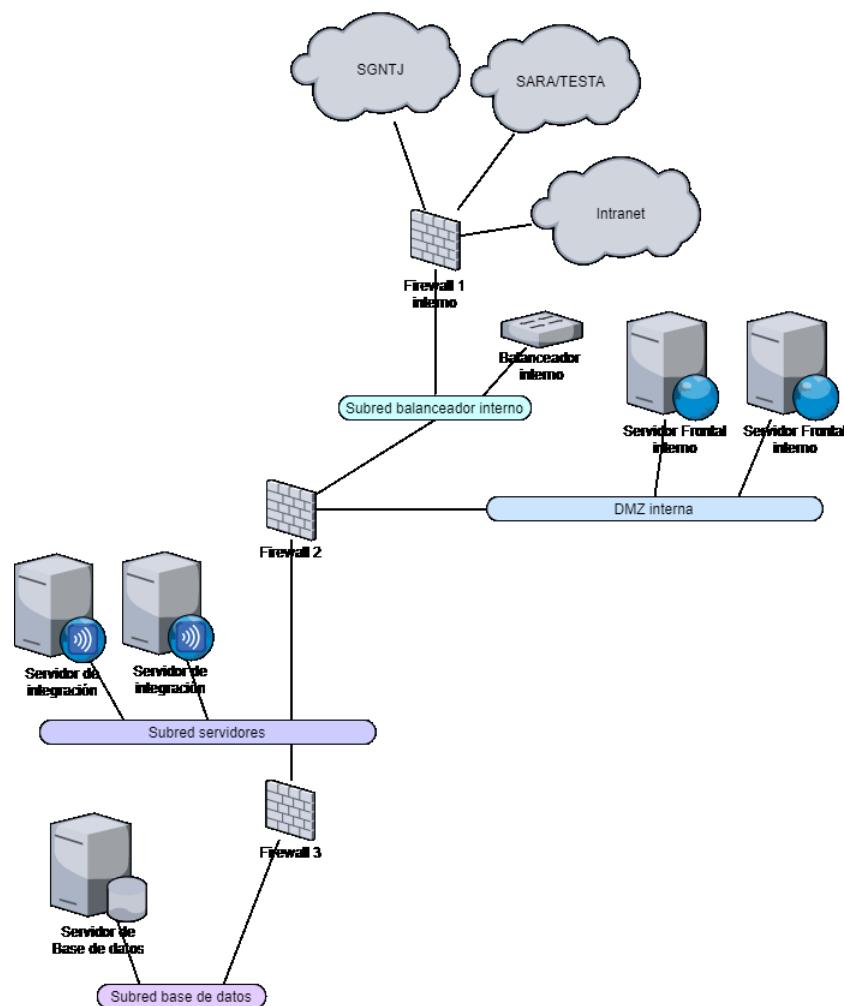


Ilustración 4-2 Arquitectura física del middleware interno

4.1.2 Arquitectura lógica

En los servidores de integración se despliegan los servicios de intermediación, que contienen la lógica de integración entre los sistemas o subsistemas llamantes, y el sistema o subsistema destino.

La arquitectura lógica de estos servicios de intermediación define los elementos que se detallan a continuación.

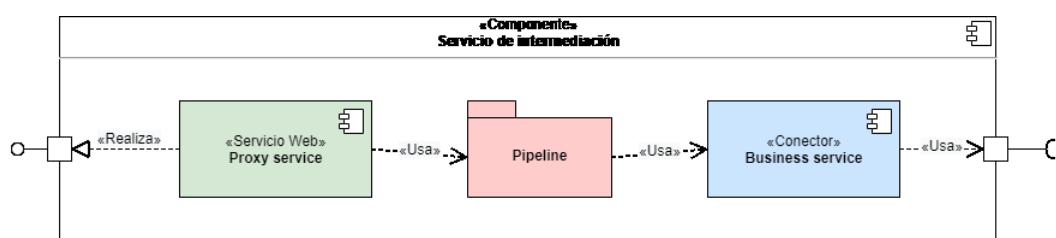


Ilustración 4-3 Arquitectura lógica de los servicios de intermediación

Los **Proxy Service** modelan el interfaz expuesto a los sistemas o subsistemas llamantes, así como la configuración de transporte y seguridad necesarias para acceder a él.

Los **Business Service**, por su lado, modelan el cliente del servicio Web destino, y la configuración de transporte y seguridad necesaria para acceder a él.

Las peticiones entrantes que se reciban por un *proxy service* son encaminadas a un **Pipeline** o flujo de procesamiento, que contiene la lógica de integración. Ésta lógica se articula en un flujo de pasos, que permiten el procesamiento, transformación y enrutamiento de los mensajes de petición al *business service* destino, así como de los mensajes de respuesta al *proxy service* origen.

4.2 Diseño de servicios Web interoperables

Los diferentes sistemas o subsistemas que expongan funcionalidad que pueda ser reutilizada por otros, deberán hacerlo mediante la definición de servicios Web. Para ello, se habrán de seguir las pautas que se desarrollan a continuación.

Norma AR-INT-03

Los diferentes **sistemas o subsistemas** que **expongan funcionalidad** que pueda ser reutilizada por otros, deberán hacerlo mediante la definición de **servicios Web**.

4.2.1 WSDL-first

Tradicionalmente, se han considerado dos estrategias para la definición de servicios Web: *code-first* y *WSDL-first*.

La estrategia *code-first* se basa en realizar primero la implementación del servicio, delegando a herramientas la generación del documento WSDL que especifica el interfaz del servicio. Estas herramientas aplican patrones y reglas que no siempre son bidireccionales, pudiendo derivar en documentos WSDL que podrían dar lugar a problemas de interoperabilidad entre los dos extremos de la integración.

La estrategia *WSDL-first*, por su parte, se basa en diseñar primeramente los XML Schema que modelan los tipos de datos que intercambiará el servicio Web, así como el documento WSDL que especifica los mensajes intercambiados y las operaciones existentes. Posteriormente se usan herramientas para generar el código correspondiente.

Importante

La estrategia *WSDL-first* es la que garantiza la interoperabilidad de los artefactos de código que se construyan mediante herramientas. Además, esta estrategia permite desacoplar la construcción de los elementos que se integrarán, que de esta forma puede realizarse en paralelo una vez definido el WSDL.

Norma AR-INT-04

Debe usarse la estrategia **WSDL-first** para la **especificación** de servicios Web.

4.2.2 Conformidad con WS-I

La *Web Services Interoperability Organization*, dependiente de OASIS, tiene por objeto fomentar y promover la interoperabilidad de servicios Web (*Web Services Interoperability*, o WS-I) sobre cualquier plataforma, aplicación o lenguaje de programación.

En ocasiones, los estándares involucrados en la especificación de servicios Web no están redactados de una forma suficientemente precisa, lo cual da lugar a distintas interpretaciones que, finalmente, redundan en problemas de interoperabilidad.

La WS-I organiza los estándares que afectan a la interoperabilidad de los servicios Web en **perfíles**. Un perfil es un conjunto de estándares o especificaciones comúnmente aceptadas por la industria, junto con una serie de indicaciones que establecen cómo deben usarse o interpretarse dichas especificaciones para desarrollar servicios Web interoperables entre sí.

Norma AR-INT-05

Los servicios Web que se diseñen deberán ser conformes al perfil [WS-I Basic Security Profile 1.0](#) de acuerdo a la **Tabla 4-2**, **Tabla 4-3** y **Tabla 4-4**.

El perfil *Basic Security Profile 1.0* incluye los perfiles *Basic Profile 1.1* y *Attachment Profile 1.0*, quedando determinado el conjunto de estándares, sus versiones y sus pautas de uso de la siguiente forma. Todos los elementos incluyen enlaces a las correspondientes especificaciones.

Basic Profile 1.1

[Namespaces in XML](#)

[RFC2246: The TLS Protocol Version 1.0](#)

[RFC2459: Internet X.509 Public Key Infrastructure Certificate and CRL Profile](#)

[RFC2616: Hypertext Transfer Protocol -- HTTP/1.1](#)

[RFC2818: HTTP over TLS](#)

[RFC2965: HTTP State Management Mechanism](#)

[SOAP 1.1](#)

[The SSL Protocol Version 3.0](#)

[UDDI Version 2 XML Schema](#)

[UDDI Version 2.03 Data Structure Reference, Dated 19 July 2002](#)

[UDDI Version 2.03 Data Structure Reference, Section 7](#)

[UDDI Version 2.04 API Specification, Dated 19 July 2002](#)

[WSDL 1.1](#)

[XML 1.0 \(Second Edition\)](#)

[XML Schema Part 1: Structures](#)

[XML Schema Part 2: Datatypes](#)

Tabla 4-2 Estándares de WS-I Basic Profile 1.1

Attachments Profile 1.0

[Namespaces in XML](#)

[RFC2045: Multipurpose Internet Mail Extensions \(MIME\) Part One: Format of Internet Message Bodies](#)

[RFC2046: Multipurpose Internet Mail Extensions \(MIME\) Part Two: Media Types](#)

[RFC2392: Content-ID and Message-ID Uniform Resource Locators](#)

[SOAP Messages with Attachments](#)

[WSDL 1.1, Section 5.0](#)

[XML 1.0 \(Second Edition\)](#)

Tabla 4-3 Estándares de WS-I Attachments Profile 1.0

Basic Security Profile 1.0

[RFC2246: The TLS Protocol Version 1.0](#)

[RFC2459: Internet X.509 Public Key Infrastructure Certificate and CRL Profile](#)

[RFC2818: HTTP over TLS](#)

[Simple SOAP Binding Profile Version 1.0](#)

[The SSL Protocol Version 3.0](#)

[WS-Security: Kerberos Token Profile 1.1 OASIS Standard Specification, 1 February 2006](#)

[WS-Security: Rights Expression Language \(REL\) Token Profile 1.0, OASIS Standard: 19 December 2004](#)

[WS-Security: SAML Token Profile 1.0, OASIS Standard, 01 Dec. 2004](#)

[WS-Security: SOAP Message Security 1.0 \(WS-Security 2004\), Errata 1.0 Committee Draft 200512, December 2005](#)

[WS-Security: SOAP Message Security 1.0 \(WS-Security 2004\), OASIS Standard 200401, March 2004](#)

<u>WS-Security: SOAP Messages with Attachments (SwA) Profile 1.1, OASIS Standard, 1 February 2006</u>
<u>WS-Security: UsernameToken Profile 1.0, Errata 1.0 Committee Draft 200401, September 2004</u>
<u>WS-Security: UsernameToken Profile 1.0, OASIS Standard 200401, March 2004</u>
<u>WS-Security: X.509 Certificate Token Profile, OASIS Standard 200401, March 2004</u>
<u>WS-Security: X.509 Token Profile 1.0, Errata 1.0 Committee Draft 200512, December 2005</u>
<u>XML Encryption Syntax and Processing</u>
<u>XML-Signature Syntax and Processing</u>
<u>XPointer W3C Recommendation, 25 March 2003</u>

Tabla 4-4 Estándares de WS-I Basic Security Profile 1.0

WS-I distribuye herramientas que facilitan evaluar la conformidad de los documentos WSDL.

Importante

La herramienta [SOAPUI](#), de uso muy extendido, integra estas herramientas, de modo que permite evaluar la conformidad de un documento WSDL contra WS-I Security Profile 1.0 y otros perfiles WS-I.

4.2.3 MTOM

SOAP Message Transmission Optimization Mechanism, mecanismo de optimización de la transmisión de mensajes SOAP, o MTOM, es un mecanismo que optimiza el envío de información binaria anexa a los mensajes SOAP.

Norma AR-INT-06

Los archivos adjuntos a peticiones o respuestas en interfaces WS deberán emplear el mecanismo [MTOM](#) para optimizar su transmisión.

4.2.4 Activos semánticos

El [Centro de Interoperabilidad Semántica](#) o CISE permite publicar y descargar modelos de datos normalizados e interoperables, en forma de esquemas XML, así como sus codificaciones y documentación asociada.

De esta manera, se facilita que la información intercambiada entre distintos sistemas pueda ser interpretable de forma automática.

Norma AR-INT-07

Los servicios Web que se diseñen deberán hacer uso de los modelos de datos que les sean de aplicación, de entre aquellos disponibles en el [Centro de Interoperabilidad Semántica](#).

4.3 Plataforma de servicios compartidos

La Plataforma de servicios compartidos es el conjunto de sistemas internos disponibles en la DSDD, que prestan funciones utilizadas de forma recurrente por otros sistemas o subsistemas, a través de servicios Web. El objetivo de estos sistemas, por tanto, es la reutilización de funcionalidad.

También forman parte de la plataforma de servicios compartidos diferentes pasarelas y brókeres, que facilitan el acceso, la operación y la extracción de métricas en el acceso a servicios compartidos proporcionados por la SGAD u otros organismos.

Norma AR-INT-08

Los subsistemas aplicativos que requieran funcionalidades proporcionadas por sistemas de la **Plataforma de servicios compartidos**, deben resolverlas mediante la integración con dichos sistemas.

Los subsistemas aplicativos que requieran de funcionalidades proporcionadas por sistemas externos a la DSDD para los cuales exista un **sistema intermediario** en la Plataforma de servicios compartidos, deberán acceder a dicho sistema externo a través de la integración con el correspondiente sistema intermediario.

La especificación de los sistemas que conforman en cada momento la Plataforma de servicios compartidos queda determinada por la **Política de actualización del catálogo de servicios compartidos**.

4.3.1 Oracle APEX

Oracle Application Express (APEX) es un entorno de desarrollo y ejecución que sigue el paradigma *low code*, poco código o poca programación, para crear aplicaciones Web sencillas basadas en bases de datos Oracle.

Oracle APEX se considera parte de la Plataforma de servicios compartidos, enfocándose a la construcción sencilla, rápida y barata de aplicaciones Web con funcionalidades propias de una consola de administración para un subsistema aplicativo concreto.

Norma AR-INT-09

Los subsistemas aplicativos que requieran **funcionalidades básicas** propias de una **consola de administración**, tales como la obtención de **estadísticas** predefinidas, la comprobación del **estado** del subsistema aplicativo, o la administración de **tablas maestras**, deberán basar su construcción en [Oracle APEX](#).

La versión vigente en cada momento del software base anterior queda determinada por la **Política de actualización de la línea base de plataformas**.

Debe tenerse en cuenta que el uso de Oracle APEX **viola la restricción de ámbito limitado** del subsistema aplicativo, siendo la **única excepción** contemplada al respecto en esta Arquitectura de Referencia, y el motivo por el que se restringe su aplicabilidad al de las consolas de administración.

El motivo para esta excepción es que el coste de desarrollo de dicha consola de administración como un interfaz Web en el ámbito del subsistema aplicativo no suele estar justificado por su uso, que generalmente será muy reducido.

4.4 Integración con el sistema de Business Intelligence

El sistema de *Business Intelligence*, inteligencia empresarial o BI se considera, a efectos de esta Arquitectura de Referencia, como un sistema externo, con el que, eventualmente, se habrán de integrar algunos subsistemas aplicativos.

Los sistemas de BI se alimentan de datos operacionales, esto es, datos provenientes de los distintos subsistemas aplicativos.

Mediante procesos de *Extract-Transform-Load*, extracción-transformación-carga o ETL, los datos recogidos son limpiados, normalizados, procesados y adaptados para su almacenamiento en un *Data Warehouse* o almacén de datos.

En ocasiones, se contemplan *Operational Data Stores*, almacenes de datos operacionales u ODS, que son almacenes intermedios que facilitan el procesamiento de los datos hasta su ubicación en el almacén de datos.

La integración de los subsistemas aplicativos con el sistema de BI corporativo suele realizarse tradicionalmente a través de vistas en la base de datos del subsistema aplicativo, desde la cual los procesos ETL acceden a los datos.

Sin embargo, esto no se considera una buena práctica, por distintos motivos. En primer lugar, se está produciendo un acoplamiento entre ambos sistemas, a través de la base de datos, del cual se deriva un mantenimiento mucho más costoso. Por otro lado, el acceso a los datos puede introducir una carga adicional, no contemplada en el diseño del sistema de información, que puede perjudicar su rendimiento.

Debido al alto volumen de datos intercambiados, esta integración no puede resolverse de la forma convencional establecida con carácter general en esta

Arquitectura de Referencia, esto es, a través de servicios Web. Son necesarios, por tanto, mecanismos específicos.

Norma AR-INT-10

Se establece que la integración de los subsistemas aplicativos con el sistema de BI, cuando sea requerida, se realice a través de una ***Staging Area***, área de ensayo o área temporal.

Esta área temporal se suele implementar en forma de base de datos relacional, aunque también podría considerarse el uso de archivos de texto, en formatos tales como CSV, XML, JSON, etc. generados por una tarea de mantenimiento del subsistema aplicativo y almacenados en una unidad compartida entre ambos sistemas.

En el caso de usarse bases de datos relacionales en la *Staging Area*, se emplearía un esquema *similar* al del subsistema aplicativo, aunque no idéntico:

- Debe contener únicamente los datos necesarios para las cargas, es decir, no tiene por qué tener todas las tablas existentes en el esquema origen.
- Es conveniente añadir algún campo adicional, que permita almacenar la fecha de la carga.
- No se deben aplicar restricciones de integridad, ni utilizar claves. De esta manera, se minimiza la afectación al subsistema aplicativo origen.

Un script, en el servidor donde reside la base de datos del subsistema aplicativo, se encargaría de la exportación de datos. La ejecución de este script debe planificarse para que siempre ocurra en un valle de carga, ya que su ejecución impactará al rendimiento del subsistema aplicativo. Los datos deberán exportarse a fichero en una unidad de almacenamiento compartida.

Otro script, en el servidor donde reside la base de datos de la *Staging Area*, se encargaría de la importación de los datos. La ejecución de este script debe planificarse para que siempre ocurra de forma posterior a la exportación correspondiente, pero antes de la siguiente exportación planificada.

La planificación de la ejecución de estos scripts deberá de contar con márgenes temporales suficientes para cubrir eventualidades.

Una vez trasladados los datos desde el subsistema aplicativo a la *Staging Area*, se ejecutarán los procesos ETL correspondientes, que procesarán los datos y los ubicarán en el sistema de BI.

Tras su procesamiento, los datos importados en la *Staging Area* deberán ser eliminados.

Importante

La *Staging Area*, como mecanismo de intercambio de información entre sistemas, puede requerir de medidas de protección específicas que garanticen la seguridad de la información.

5 Arquitectura lógica

La arquitectura lógica describe los subsistemas aplicativos a partir de los componentes lógicos que los conforman y de las relaciones existentes entre ellos, utilizando un nivel de abstracción que resulte razonable para su propósito.

5.1 Patrón Layered Architecture

El patrón *Layered Architecture*, o arquitectura en capas, es, probablemente, uno de los patrones de arquitectura de software más conocidos. Este patrón arquitectural se basa en la definición de una serie de capas lógicas, atribuyendo a cada una de ellas una determinada responsabilidad. A continuación, cada componente lógico se ubica en una capa u otra, en base a la correspondencia entre su función y la responsabilidad de la capa.

Las capas lógicas se organizan en forma de pila, existiendo una relación de delegación entre ellas: los componentes que se ubican en una determinada capa únicamente pueden interactuar con componentes ubicados en la capa inmediatamente inferior, o bien en la misma capa. De la misma manera, los componentes que se ubican en una determinada capa exponen su funcionalidad a componentes ubicados en la capa inmediatamente superior, o bien en la misma capa.

El patrón no exige un número concreto de capas. Por ejemplo, el modelo OSI, que se basa en este patrón para especificar una forma de diseñar pilas de protocolos, define siete capas lógicas.

La definición de estas capas constituye otra forma de descomposición modular, por lo que, de nuevo, prevalecen los principios de máxima cohesión y mínimo acoplamiento. La cohesión funcional entre los componentes que se ubican en una determinada capa viene dada por la responsabilidad atribuida a dicha capa. Por otro lado, el bajo acoplamiento se consigue mediante las restricciones que el patrón impone a la interacción de componentes, y que consiguen que cada capa únicamente esté acoplada con la capa inmediatamente inferior.

Norma AR-LOG-01

Se establece el uso del patrón **Layered Architecture**, estableciendo tres capas lógicas de acuerdo a la **Ilustración 5-1**, cuya responsabilidad se determina a continuación.

- Capa de **presentación**: su responsabilidad es facilitar la interacción de los usuarios con el subsistema aplicativo, traduciendo sus peticiones en llamadas a la capa de negocio, y formateando la información intercambiada de modo que pueda ser inteligible por ellos.
- Capa de **negocio**: su responsabilidad es ejecutar las operativas de negocio del subsistema aplicativo, a petición de la capa de presentación.

- Capa de **acceso a datos**: su responsabilidad es facilitar el acceso a los datos, que se ubican en bases de datos, unidades de almacenamiento u otros sistemas de información.

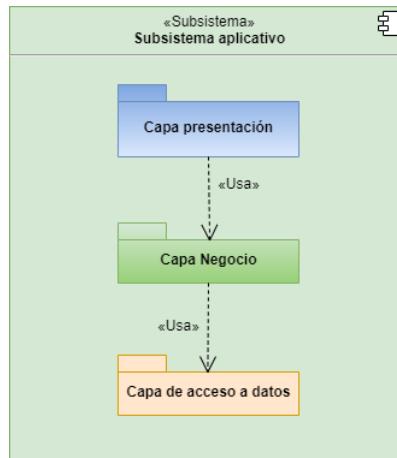


Ilustración 5-1 Patrón Layered Architecture con tres capas

En ocasiones, es posible establecer un segundo nivel de descomposición modular, descomponiendo las capas lógicas anteriores en sub-capas, con un mayor nivel de especialización. Por ejemplo, la capa de negocio podría descomponerse, a su vez, en las sub-capas de control transaccional, de validación de datos, de reglas de negocio, etc.

También es habitual descomponer una capa lógica en forma de sub-capas adyacentes, en lugar de apiladas, que se caracterizan por estar completamente desacopladas entre sí, esto es, sin que exista ninguna interacción entre ellas.

Norma AR-LOG-02

Se establece la descomposición de capas en **sub-capas adyacentes** de acuerdo a la **Ilustración 5-2**, según se determina a continuación.

La capa de **presentación** queda descompuesta en las siguientes sub-capas adyacentes:

- Capa **Web**: su responsabilidad es facilitar la interacción de usuarios humanos con el subsistema aplicativo, empleando para ello tecnologías Web.
- Capa de **servicios Web**: su responsabilidad es facilitar la interacción de usuarios automáticos con el subsistema aplicativo, empleando para ello servicios Web.

La capa de **acceso a datos** queda descompuesta, a su vez, en las siguientes sub-capas adyacentes:

- Capa de **persistencia**: su responsabilidad es facilitar el acceso y manipulación de los datos persistentes, que se ubican en bases de datos u otras unidades de almacenamiento.
- Capa de **integración**: su responsabilidad es facilitar el acceso y manipulación de los datos que se ubican en otros sistemas de información externos.

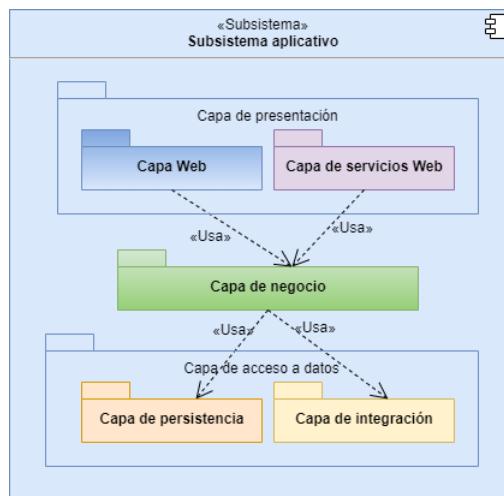


Ilustración 5-2 Patrón Layered Architecture con descomposición en subcapas adyacentes

5.2 Capas lógicas en estereotipos aplicativos

La especificación de las capas existentes en la arquitectura lógica de un subsistema aplicativo dependerá del estereotipo de éste, como se indica a continuación.

Norma AR-LOG-03

La especificación de las capas existentes en la arquitectura lógica de un subsistema aplicativo dependerá del estereotipo de éste, de acuerdo con la **Ilustración 5-3, Ilustración 5-4, Ilustración 5-5 e Ilustración 5-6**.

5.2.1 Aplicación Web

Las aplicaciones Web dispondrán de una capa Web, una capa de negocio, una capa de persistencia y, eventualmente, una capa de integración.

Los componentes ubicados en la capa Web *realizan* (materializan o implementan) el interfaz Web que provee el subsistema aplicativo.

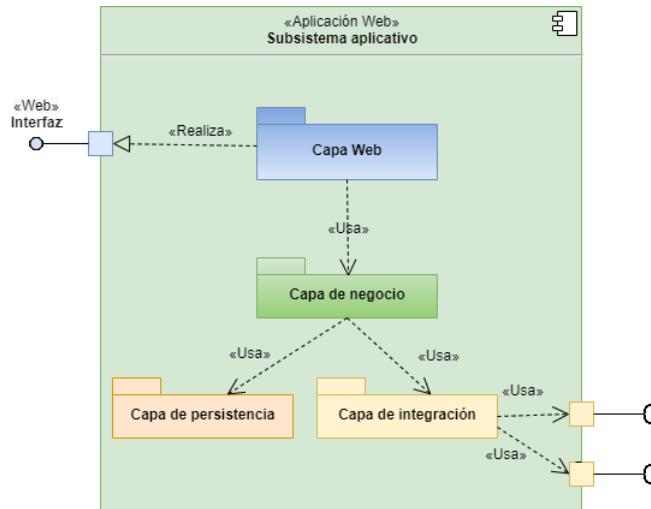


Ilustración 5-3 Capas lógicas en subsistemas aplicativos con estereotipo Aplicación Web

5.2.2 Fachada Web

Las fachadas Web dispondrán de una capa Web, una delgada capa de negocio, y una capa de integración, careciendo de capa de persistencia.

Los componentes ubicados en la capa Web *realizan* (materializan o implementan) el interfaz que provee el subsistema aplicativo.

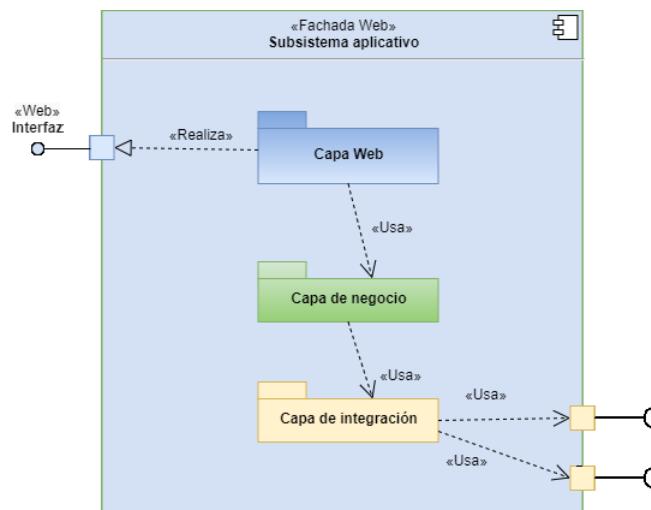


Ilustración 5-4 Capas lógicas en subsistemas aplicativos con estereotipo Fachada Web

5.2.3 Aplicación de servicios Web

Las aplicaciones de servicios Web dispondrán de una capa de servicios Web, una capa de negocio, una capa de persistencia y, eventualmente, una capa de integración.

Los componentes ubicados en la capa de servicios Web *realizan* (materializan o implementan) los interfaces WS que provee el subsistema aplicativo.

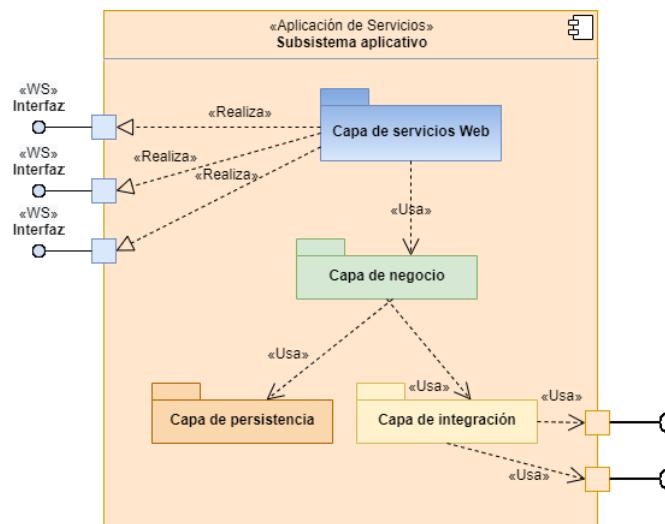


Ilustración 5-5 Capas lógicas en subsistemas aplicativos con estereotipo Aplicación de Servicios

5.2.4 Aplicación Híbrida

Las aplicaciones Híbridas dispondrán de una capa Web, una capa de servicios Web, una capa de negocio, una capa de persistencia y, eventualmente, una capa de integración.

Los componentes ubicados en la capa Web *realizan* (materializan o implementan) el interfaz o interfaces Web que provee el subsistema aplicativo, mientras que los componentes ubicados en la capa de servicios Web *realizan* los interfaces WS.

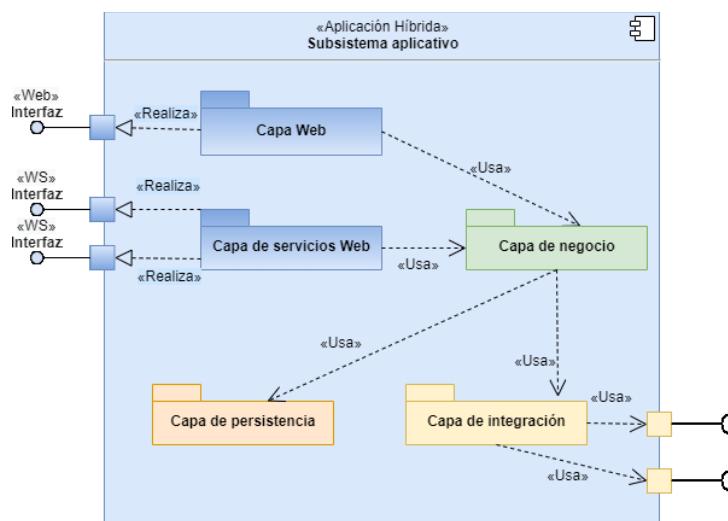


Ilustración 5-6 Capas lógicas en subsistemas aplicativos con estereotipo Aplicación Híbrida

5.3 Componentes lógicos

Los componentes lógicos son unidades modulares de software que encapsulan un conjunto de funcionalidades. Por tanto, de nuevo aplican los principios de máxima cohesión en la funcionalidad que ofrecen, y mínimo acoplamiento en la interacción con otros componentes con los que colaboren.

Los componentes son abstracciones de mayor nivel que las clases, siendo muy útiles en el ámbito del diseño arquitectural. Posteriormente, en tiempo de diseño detallado, cada componente que se haya definido se expresará mediante un conjunto de clases, estando formado por:

- Una clase principal que lo dota de comportamiento, junto con los interfaces que implementa y, eventualmente, la clase que extienda.
- Otras clases auxiliares que puedan colaborar con la clase principal, como por ejemplo clases de objetos internos, aspectos AOP, etc.
- Todas las clases que representan los datos que recibe como parámetros, los datos que retorna como resultado, o las excepciones que pueda lanzar en cada uno de sus métodos.

Los componentes colaboran entre sí para resolver problemáticas complejas. Esta interacción se lleva a cabo mediante interfaces, que cada componente provee para que otros puedan utilizar las funcionalidades que expone.

5.3.1 Definidos por un interfaz

Cada componente se define mediante un interfaz, que especifica los métodos que determinan su funcionalidad.

Cuando un componente interactúa con otro, siempre debe hacerlo a través del interfaz que lo define, ya que esta es la forma más desacoplada posible.

De esta manera, se abre la posibilidad a que existan diferentes implementaciones de un mismo componente, esto es, de las atribuciones funcionales que éste tiene, y que quedan especificadas por el interfaz que lo define. Esto permite entender los componentes como piezas intercambiables: una implementación puede ser sustituida por otra, sin que el resto de componentes quede afectado.

Otra ventaja derivada de ello es la mayor facilidad de realización de pruebas unitarias: los componentes de los que dependa el componente que se desea probar pueden ser fácilmente sustituidos por *mocks*, que facilitan notablemente la creación de escenarios específicos de prueba.

Norma AR-LOG-04

Se establece que los componentes se definen mediante un **interfaz**, el cual especifique los métodos que determinen sus atribuciones funcionales.

5.3.2 Patrón Singleton

En la gran mayoría de las ocasiones, los componentes siguen un patrón *Singleton*, que determina que únicamente exista una instancia de la clase principal que implementa al componente.

Puesto que los componentes colaboran unos con otros, lo que en realidad existirá será una estructura de objetos, interconectados entre sí. Esta estructura será única y compartida por todas las hebras que, concurrentemente, atiendan las peticiones provenientes de los usuarios.

Norma AR-LOG-05

Se establece el uso del patrón **Singleton** en el diseño de componentes lógicos.

5.3.3 Patrón Stateless

El patrón *Stateless* establece que los componentes no tengan estado, esto es, que no almacenen en sus variables internas información relacionada con la petición o con la sesión. Esta información pertenece al ámbito de la hebra que está atendiendo la petición, y no al ámbito del componente.

Norma AR-LOG-06

Se establece el uso del patrón **Stateless** en el diseño de componentes lógicos.

Importante

Esto se traduce en que las únicas variables que puede declarar una clase que implementa un componente son:

- Constantes
- Propiedades de configuración
- Referencias a otros componentes de los que dependa

Toda otra información que requiera dicha clase, debe ser recibida desde los argumentos de sus métodos. De igual manera, toda variable que precise dicha clase para facilitar la implementación de una operativa deberá ser definida en el método correspondiente.

5.3.4 Patrón Business Facade

La colaboración entre componentes, por lo general, suele seguir un esquema jerárquico, distinguiéndose componentes principales y componentes secundarios, de carácter auxiliar o de apoyo.

El patrón *Business Facade*, o fachada de negocio, se refiere a la existencia de componentes que declaran operativas con un nivel de abstracción mayor que las expuestas por el resto de componentes.

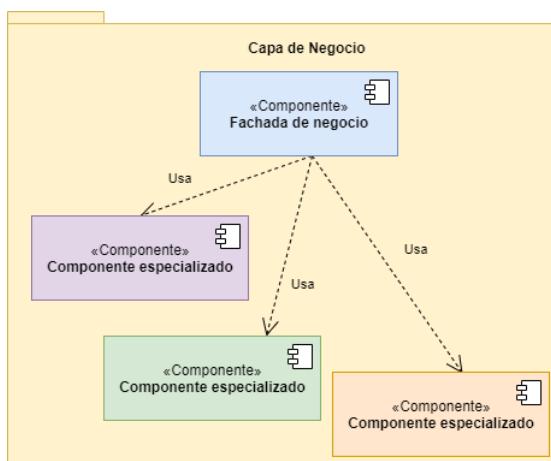


Ilustración 5-7 Patrón Business Facade

La implementación de estas operativas se resuelve mediante la orquestación de llamadas a otros componentes de la capa de negocio, jerárquicamente inferiores, así como a los componentes ubicados en la capa de acceso a datos.

Norma AR-LOG-07

Se establece que se defina en la capa de negocio una o más **fachadas de negocio** para definir y modelar las operativas de negocio que conforman la funcionalidad provista por un subsistema aplicativo.

En caso de emplear un diseño arquitectónico **monolítico modular**, el subsistema aplicativo deberá disponer de una fachada de negocio para definir y modelar la funcionalidad provista por cada uno de los módulos que lo conforman.

En caso de emplear un diseño arquitectónico basado en **miniservicios**, cada subsistema aplicativo deberá disponer de una fachada de negocio para definir y modelar la funcionalidad provista a través del interfaz correspondiente, ya sea Web (fachadas de negocio) o de servicios Web (aplicaciones de servicios Web).

5.3.5 Manifestación de componentes por artefactos

Los componentes que se definen en la arquitectura lógica de un determinado subsistema aplicativo son *manifestados* por uno o más artefactos. Esto es, las clases que implementan dichos componentes son compiladas y ensambladas en forma de artefactos.

El código fuente de estas clases se organiza en proyectos aplicativos, los cuales facilitan su compilación y ensamblado en forma de artefactos. En función de los artefactos que se hayan concebido, pueden ser necesarios uno o más proyectos

aplicativos para manifestar todos los componentes que se hayan especificado en la arquitectura lógica de un subsistema aplicativo.

Norma AR-LOG-08

Los componentes que vayan a ser compartidos por distintos subsistemas aplicativos, o que puedan serlo, deben ser manifestados por **librerías**. De esta manera se facilita su reutilización.

Norma AR-LOG-09

Cada servicio Web identificado en la arquitectura lógica del sistema llevará asociado, como artefacto propio, una **librería de especificación de servicio Web**.

Este artefacto contendrá los interfaces y clases que se habrán generado automáticamente mediante herramientas, a partir del WSDL que define el servicio Web y los esquemas XML que éste incluye o referencia.

Estos elementos facilitarán la implementación tanto de la parte servidora, como de la parte cliente.

5.4 Contenedor de componentes

El uso de un contenedor de componentes facilita notablemente el trabajo con éstos, como se indica a continuación.

5.4.1 Principio Inversion of Control

El principio *Inversion of Control*, IoC o inversión de control, se basa en invertir el flujo de ejecución tradicional de un programa.

En los métodos de programación tradicionales, la interacción se expresa de forma imperativa, haciendo llamadas a métodos. La inversión de control, sin embargo, se basa en especificar las respuestas deseadas ante la ocurrencia de determinados eventos, dejando que un elemento externo al programa se encargue de realizar las acciones de control que sean necesarias para generar dichas respuestas. Este elemento externo tiene sentido de contenedor, o *framework*, en cuyo interior se despliegan los componentes lógicos.

Por ejemplo, un componente puede especificar, mediante metadatos, que cuando llegue una petición cuya URL cumpla determinado patrón, el contenedor debe invocar uno de sus métodos, proporcionándole los datos de la petición que éste requiera a través de sus argumentos, y finalmente plasmar en la respuesta el objeto retornado por el método como resultado.

De esta manera, el componente queda completamente liberado de implementar esta lógica de control, centrándose únicamente en resolver la funcionalidad para la que fue concebido.

5.4.2 Patrón Dependency Injection

El patrón *Dependency Injection*, DI, o inyección de dependencias, es una de las posibles formas de implementar el principio de inversión de control, a través del establecimiento de las dependencias entre los distintos componentes.

De esta manera, el contenedor o *framework* será el encargado de instanciar los diferentes componentes, conectar unos con otros, y gestionar su ciclo de vida. Para todo ello, se basará en metadatos proporcionados por los propios componentes.

5.4.3 Framework Spring

El *framework* Spring proporciona un contenedor de inversión de control mediante inyección de dependencias basado en Java, ampliamente empleado y con gran madurez. Adicionalmente, proporciona numerosos componentes técnicos que permiten adaptarlo a diferentes ámbitos: acceso a datos, interfaces Web, servicios Web, integración con sistemas externos, pruebas, etcétera.

Norma AR-LOG-10

Se establece el uso [Spring](#) como *framework* base para la implementación de componentes lógicos y la gestión de su ciclo de vida.

5.5 Modelo de dominio

El dominio es el conjunto de elementos del mundo real, o conceptos, que participan en la problemática que el subsistema aplicativo resuelve.

El modelo de dominio es una representación de dichos elementos o conceptos mediante objetos. Esta representación es, como en todo modelo, simplificada, pero ha de ser lo suficientemente sofisticada como para resultar útil.

Los modelos de dominio se suelen implementar mediante objetos planos, que definen, mediante atributos, los datos más relevantes de cada elemento o concepto.

5.5.1 Patrón Value Object

El patrón *Value Object*, u objeto valor, se refiere a la representación mediante objetos del valor de alguno de dichos atributos, esto es, definiendo clases para representar tipos de datos que aparecen de forma recurrente en los atributos de los objetos del modelo de dominio.

Un ejemplo de aplicación de este patrón sería disponer de clases que modelen tipos de datos como, por ejemplo, un NIF.

Los objetos valor contienen métodos que facilitan el tratamiento de los valores que representan.

Norma AR-LOG-11

Se establece el uso del patrón **Value Object** para la definición de tipos de datos de uso frecuente en los modelos de dominio.

5.5.2 Desacoplamiento entre el modelo de dominio y la capa de presentación

El modelo de dominio es utilizado de forma compartida por los diferentes componentes lógicos de la capa de negocio, puesto que representa los elementos de información con los que trabajar para resolver las operativas de negocio.

Por otro lado, los elementos del dominio también son visibles a través de los interfaces que las fachadas de negocio exponen a la capa de presentación, cuyo cometido es realizar una representación de estas estructuras de modo que el usuario las pueda comprender.

Sin embargo, en muchas ocasiones es conveniente restringir la información que se presenta a los usuarios, escondiendo determinada información por motivos de seguridad, limitándola por motivos de rendimiento, simplificándola para lograr una mayor usabilidad, o modelándola de una manera alternativa.

Norma AR-LOG-12

Se establece desdoblar el modelo de dominio en dos modelos.

El modelo de dominio interno o **privado** será el que emplearán internamente los componentes ubicados por debajo de las fachadas de negocio. Este modelo queda reflejado en los métodos que definen estos componentes, a través de los tipos de datos de entrada o salida.

Sin embargo, las fachadas de negocio deben ofrecer a la capa de presentación un modelo de dominio alternativo, externo o **público** y desacoplado del modelo privado, que contenga exclusivamente la información que se debe mostrar a los usuarios, y que se estuture de forma que se facilite la interacción con ellos.

5.5.3 Patrón Domain Entity

El patrón *Domain Entity* o entidad de dominio, o también simplemente *Entity* o entidad, se aplica en los objetos del modelo de dominio que se encuentren definidos por su identidad, y que tengan la característica de ser persistentes en el tiempo.

Las entidades son, por tanto, objetos del modelo de dominio que tienen un largo tiempo de vida, y que están dotados de una identidad, esto es, una referencia que las identifica únicamente y que nunca cambia a lo largo del tiempo.

Debido a su largo tiempo de vida, estos objetos (mejor dicho, la información que contienen) deben guardarse en bases de datos o unidades de almacenamiento, recuperándose y almacenándose según sea necesario para resolver las operativas

de negocio. Su identidad, por otro lado, permite referirse a ellos en cualquier momento del tiempo.

En un caso general, el modelo de dominio privado estará compuesto por entidades, pero también por elementos no persistentes.

Norma AR-LOG-13

Únicamente podrán ser considerados entidades persistentes algunos de los elementos del modelo de dominio **privado**. El modelo de dominio público no deberá disponer en ningún caso de entidades persistentes.

Norma AR-LOG-14

Se establece el uso de la [Java Persistence API](#) como especificación para la definición de modelos de dominio persistentes.

5.6 Interacción entre componentes

La interacción entre componentes se lleva a cabo mediante la invocación de los métodos que los componentes exponen en sus interfaces.

Norma AR-LOG-15

La interacción de un componente con otros se llevará a cabo únicamente a través de los **métodos** definidos en los **interfaces** de éstos.

5.6.1 Patrón Data Transfer Object

El patrón *Data Transfer Object* o DTO, objeto de transferencia de datos, se refiere al uso de objetos para agregar y encapsular los datos que se han de intercambiar en una interacción, ya sea entre subsistemas, mediante el uso de protocolos de comunicaciones, o incluso entre componentes, de forma local. Son objetos, por tanto, usados exclusivamente para transmitir información, y carecen de cualquier lógica.

Los DTO tienen un tiempo de vida muy corto, de apenas unos milisegundos: se crean, se pueblan de datos, se utilizan para transmitir la información requerida por la interacción, y finalmente se desechan.

Existe una relación estrecha entre el modelo de dominio y los DTO empleados en las interacciones entre componentes: los objetos del modelo de dominio pueden estar contenidos dentro de los DTO, o incluso ser directamente usados como DTO.

Los componentes de la capa de integración usan DTO para facilitar el intercambio de información entre los componentes de la capa de negocio y los sistemas externos. Estos DTO manifiestan el dominio de dichos sistemas externos. Aunque en ocasiones el dominio de dichos sistemas externos pueda coincidir parcialmente con

el dominio del subsistema aplicativo, los elementos comunes no tienen por qué estar modelados de la misma forma en ambos sistemas.

Los componentes de la capa de persistencia, así como los componentes internos de la capa de negocio usan DTO que reflejan el modelo de dominio interno o privado del subsistema aplicativo.

Las fachadas de negocio emplean DTO para intercambiar información con la capa de presentación. Estos DTO reflejan el modelo de dominio externo o público del subsistema aplicativo, debiendo encargarse la fachada de negocio de la necesaria conversión de formato.

Los componentes de la capa de servicios Web emplean DTO para intercambiar información con los sistemas externos, debiendo usar para ello estructuras de datos adecuadas al protocolo empleado.

Los componentes de la capa Web pueden utilizar DTO para estructurar la información intercambiada con los usuarios, o bien utilizar directamente los DTO que ofrecen las fachadas de negocio.

5.6.2 Excepciones

Las excepciones son formas anómalas en las que termina la ejecución de un método, y son susceptibles también del patrón DTO, agregando los datos que modelan el error ocurrido.

Norma AR-LOG-16

En función de la forma en la que se han de tratar, se pueden distinguir los siguientes **tipos** de excepción:

- **Excepciones técnicas:** modelan errores que no tienen solución en tiempo de ejecución, generalmente porque el software está indebidamente o incompletamente programado o probado. No hay nada que pueda hacer el usuario, o el operador del sistema, para solucionar el error, que únicamente puede subsanarse en una versión correctiva del software.
- **Excepciones transitorias:** modelan errores debidos a que el entorno se encuentra en una situación anómala. Por ejemplo, un sistema externo, del cual depende una operativa de negocio, se encuentra indisponible. En cuanto el operador de dicho sistema solucione su indisponibilidad, el error dejará de ocurrir.
- **Excepciones de negocio:** modelan errores relacionados con las reglas de negocio, generalmente porque el usuario ha introducido datos o ha solicitado acciones que incumplen determinadas reglas de negocio. Por tanto, son excepciones que se detectan y originan en los componentes de

la capa de negocio. Si se asiste al usuario adecuadamente, en un reintento posterior el error ya no se dará.

Norma AR-LOG-17

Las excepciones deben **mostrarse al usuario** en la capa de **presentación**, de una forma conveniente. En función del tipo de excepción, el tratamiento habrá de ser el siguiente:

- Excepciones **técnicas**: indicación, de forma textual o gráfica (usuarios humanos), o mediante un código de error específico (usuarios automáticos), de que se ha producido un error interno, y que el usuario tiene la posibilidad de ponerse en contacto con el soporte del sistema para abrir una incidencia.
- Excepciones **transitorias**: indicación, de forma textual o gráfica (usuarios humanos), o mediante un código de error específico (usuarios automáticos), de que se ha producido una indisponibilidad transitoria, y que el usuario tiene la posibilidad de reintentarlo más adelante, o bien ponerse en contacto con el soporte del sistema para abrir una incidencia.
- Excepciones **de negocio**: indicación, de forma textual o gráfica (usuarios humanos), o mediante un código de error específico (usuarios automáticos), de que se ha producido un error, acompañada de la información necesaria para especificar, de la forma más detallada posible, dónde se produjo el error y por qué, de modo que el usuario pueda reintentarlo de nuevo.

5.7 Catálogo de estereotipos de componente

En función de la capa lógica donde se ubiquen, se establece el siguiente catálogo de estereotipos de componente, que determina su comportamiento.

5.7.1 DAO

Los componentes con estereotipo **DAO** se ubican en la **capa de persistencia**, y poseen las características que se indican a continuación.

Norma AR-LOG-18

Los componentes con estereotipo **DAO** se ubican en la **capa de persistencia**, y se caracterizan por aplicar el patrón *Data Access Object*, DAO u objeto de acceso a datos, que determina su función como facilitador del acceso a datos persistentes por parte de sus componentes usuarios, ocultando toda la complejidad subyacente.

Estos datos persistentes son modelados mediante los objetos persistentes del modelo de dominio interno, que emplean el patrón *Data Entity*.

Los componentes **DAO** se organizan en torno a las entidades más relevantes para la lógica de negocio, cuyo acceso gestionan, de modo que existirá un componente **DAO** para cada una de ellas.

Estos componentes ofrecerán funciones básicas de acceso y almacenamiento individual de entidades, así como las búsquedas que puedan requerir las operativas de negocio, y son susceptibles de emitir excepciones técnicas o transitorias.

Estos componentes no deben encargarse del control transaccional del acceso a datos persistentes, ya que éste corresponde al ámbito de la capa de negocio, en concreto, a fachada de negocio correspondiente. No obstante, sí deben verificar que efectivamente se ha iniciado una transacción, si es que ésta es necesaria, en el momento en que son invocados.

5.7.2 Conector

Los componentes con estereotipo **Conector** se ubican en la **capa de integración**, y poseen las características que se indican a continuación.

Norma AR-LOG-19

Los componentes con estereotipo **Conector** se ubican en la **capa de integración**, y se caracterizan por facilitar la interacción de los componentes de negocio con los diferentes sistemas externos, usando el patrón *Data Transfer Object* o **DTO** para agregar y encapsular la información intercambiada.

Los componentes **Conector** se organizan en torno a los sistemas externos, de modo que existirá uno o más conectores para cada uno de ellos.

Estos componentes son susceptibles de emitir excepciones técnicas o transitorias, o bien excepciones que tienen sentido de negocio en el sistema externo correspondiente.

5.7.3 Componente

Los componentes con estereotipo **Componente** se ubican en la **capa de negocio**, y poseen las características que se indican a continuación.

Norma AR-LOG-20

El estereotipo **Componente** refleja componentes genéricos. Los componentes de la **capa de negocio** utilizan este estereotipo, independientemente de su relevancia o rango en la jerarquía.

Los componentes de la capa de negocio son susceptibles de emitir excepciones técnicas o de negocio. Deben gestionar las posibles excepciones de negocio de la capa de integración, así como propagar las excepciones técnicas o transitorias que hayan podido emitir los componentes subyacentes.

Si el tratamiento de una excepción de negocio lanzada por un conector motiva el lanzamiento de una excepción de negocio en un componente de negocio, la primera debe establecerse como **causa** de la segunda. De esta manera, se garantiza la trazabilidad de la cadena de errores.

Un tipo especial de componente en la capa de negocio son las *Business Facade* o fachadas de negocio, que poseen las características que se indican a continuación.

Norma AR-LOG-21

Las **fachadas de negocio** definirán métodos reflejando operativas de negocio, y emplearán el patrón *Data Transfer Object* o **DTO** para agregar y encapsular la información que intercambian con la capa de presentación a través de dichos métodos. Estos DTO reflejan el modelo de **dominio externo** del subsistema aplicativo.

Las fachadas de negocio deben encargarse del control transaccional del acceso a datos, y la implementación de sus métodos se limitará básicamente a **orquestar** a otros componentes de la capa de negocio, de menor rango y granularidad más fina.

Las fachadas de negocio son agnósticas respecto al tipo de componente que las invoca, y sus métodos deben resolver la lógica de negocio de la operativa correspondiente, independientemente de la forma en que se presente la información al usuario.

5.7.4 Controlador

Los componentes con estereotipo **Controlador** se ubican en la **capa Web**, donde se aplica el patrón *Model-View-Controller* o Modelo-Vista-Controlador. Este patrón descompone el problema de la presentación en tres partes diferenciadas:

- **Modelo:** engloba tanto al modelo de dominio como a las operativas de negocio del subsistema aplicativo. Se corresponde, por tanto, con las fachadas de negocio que se ubican en la capa de negocio, así como los DTO que participan en sus métodos, y que reflejan el modelo de dominio público.
- **Vista:** permite la visualización de los datos resultantes de la invocación a una operativa de negocio, e incluye los enlaces, formularios y otros controles que pueda utilizar el usuario para solicitar la siguiente interacción con el sistema.
- **Controlador:** se encarga de controlar las interacciones del usuario con el sistema, traduciendo los eventos generados por el usuario en la ejecución de métodos expuestos por las fachadas de negocio, y determinando la vista encargada de la representación de su resultado.

Los componentes **Controlador** se ubican en este último bloque. Su única función, por tanto, es la de orquestación. La lógica necesaria para la presentación de la información, incluidas las excepciones técnicas, transitorias o de negocio, se encuentra en las vistas.

Norma AR-LOG-22

Los componentes con estereotipo **Controlador** se ubican en la **capa de presentación**, y se caracterizan por actuar como controladores en el contexto del patrón *Model-View-Controller* o *Modelo-Vista-Controlador*.

Los controladores se encargan de gestionar las interacciones del usuario con el sistema, traduciendo los eventos generados por el usuario al actuar sobre una determinada vista en la ejecución de métodos expuestos por la correspondiente fachada de negocio, y determinando la siguiente vista que habrá de mostrarse, la cual se encargará de la representación del resultado obtenido.

5.7.5 Servicio Web

Los componentes con estereotipo **Servicio Web** se ubican en la **capa de servicios Web**, donde, al igual que en la capa Web, se aplica el patrón *Model-View-Controller* o *Modelo-Vista-Controlador*.

Si en la capa Web las vistas se implementan mediante páginas que incluyen la lógica de presentación necesaria, en el caso de los servicios Web es un *framework* quien se encarga de presentar la información, a partir de la meta-information existente en DTO específicos de la capa de servicios Web.

Estos DTO se generan automáticamente mediante herramientas, a partir del WSDL que define el servicio Web y los esquemas XML que éste incluye o referencia, y no tienen por qué corresponderse con los DTO propios de las fachadas de negocio.

Norma AR-LOG-23

Los componentes con estereotipo **Servicio Web** se ubican en la **capa de servicios Web**, y se caracterizan por actuar como **controladores** en el contexto del patrón *Model-View-Controller* o *Modelo-Vista-Controlador*.

Estos componentes deben encargarse de los necesarios mapeos de datos entre los DTO de las fachadas de negocio, incluidas las excepciones técnicas, transitorias y de negocio, y sus propios DTO, especificados en el WSDL y los correspondientes esquemas XML.

Se definirá un componente **Servicio Web** específico para cada **fachada de negocio** cuya funcionalidad se desee publicar, aunque no se expongan al exterior todos los métodos que se han definido en ésta.

5.7.6 Filtro Web

Los componentes con estereotipo **Filtro Web** se ubican en la **capa Web**, o bien en la **capa de servicios Web**, y poseen las características que se indican a continuación.

Norma AR-LOG-24

Los componentes con estereotipo **Filtro Web** se ubican en la **capa Web** o en la **capa de servicios Web**, y se caracterizan por aplicar el patrón *Interceptor* al tratamiento de peticiones Web, que determina su función como pre-procesador y/o post-procesador de peticiones Web.

Estos componentes suelen organizarse en cadenas ordenadas, para lograr así comportamientos complejos, como por ejemplo los de control de autenticación y autorización.

Estos componentes son susceptibles de emitir excepciones técnicas.

5.8 Tareas de mantenimiento

Muchos sistemas de información requieren de la realización de tareas de mantenimiento automáticas, de forma periódica, siendo las más habituales las siguientes:

- Tareas de **purgado**: eliminación de datos antiguos u obsoletos.
- Tareas de **consolidación**: consolidación de información en la base de datos o unidades de almacenamiento compartido.
- Tareas de **importación**: importación de datos recibidos desde otros sistemas o subsistemas.
- Tareas de **exportación**: exportación de datos para su envío a otros sistemas o subsistemas, o bien para la elaboración de informes.

Las tareas de mantenimiento deberán diseñarse según las pautas que se indican a continuación.

Norma AR-LOG-25

Las **tareas de mantenimiento** deberán diseñarse según las siguientes pautas:

- Deben estar funcionalmente definidas, entendiéndose como **casos de uso** durante la etapa de análisis funcional del sistema de información, disparados por el **actor Tiempo**.
- Todas las tareas de mantenimiento de un subsistema aplicativo deberán diseñarse como operativas de negocio, ubicadas por tanto en una **fachada de negocio**.
- La capacidad del sistema requerida por una tarea de mantenimiento debe ser limitada. Esto implica la **división en lotes** del trabajo realizado por la tarea de mantenimiento, de tamaño tal que su procesamiento no se apodere de todos los recursos, sino de una fracción limitada.
- El **tamaño** de estos **lotes** debe ser **configurable**, para una mayor flexibilidad.

- Las tareas de mantenimiento deben ejecutarse en los periodos de tiempo en los que existe un valle en la distribución de carga, para que su impacto en el servicio sea mínimo.
- El **patrón de planificación** de las tareas de mantenimiento debe ser **configurable**, para una mayor flexibilidad.

Norma AR-LOG-26

Se establece el uso de [Quartz](#) como framework base para la planificación de tareas.

5.9 Solicitudes de trabajo por lotes

Algunos sistemas de información requieren el soporte de solicitudes de trabajo por lotes, enviadas por usuarios específicos a través de interfaces Web o WS.

El tratamiento de solicitudes de trabajo por lotes supone un consumo de recursos que podría afectar al servicio del sistema de información. Puesto que la capacidad del sistema es limitada, la fracción de ésta que se dedique al procesamiento del lote, no puede dedicarse a absorber la carga inyectada por los usuarios.

Para abordar esta problemática sin comprometer al servicio, se determinan una serie de normas fundamentales, que se enumeran a continuación.

Norma AR-LOG-27

Las **solicitudes de trabajo por lotes** deberán diseñarse según las siguientes pautas:

- Las solicitudes de trabajo por lotes se modelarán en forma de **fichero**, con un formato normalizado que facilite su tratamiento, tal como CSV, XML, JSON, etc.
- El fichero será recibido por el sistema de información a través de un interfaz Web, o bien como archivo adjunto a una petición WS. Alternativamente, sería posible que el sistema de información generase dicho fichero a partir de información introducida por el usuario.
- El lote deberá tener un **tamaño máximo** establecido, que el sistema de información deberá validar.
- Este tamaño máximo de lote deberá ser **configurable**, para mayor flexibilidad.
- El sistema de información validará el formato del fichero recibido, y lo almacenará en una **unidad de almacenamiento**, para su procesamiento posterior.
- Si es necesario, el sistema de información generará un recibo o respuesta equivalente que acredite la **correcta recepción del fichero y su validez**.

- El procesamiento en sí del fichero recibido se realizará por una **tarea de mantenimiento**, y nunca en tiempo de procesamiento de la petición que incluía el fichero.

6 Arquitectura de datos

El activo principal de la mayoría de los sistemas de información de las Administraciones Públicas son los datos, por encima de las operativas de negocio. Incluso, por encima del propio sistema de información que los maneja, que desaparecerá, y aparecerá otro nuevo que lo sustituya, y seguirá manejando los datos de su predecesor.

Se trata, por tanto, de sistemas de información con un diseño centrado en los datos, donde cobran una especial relevancia los aspectos del diseño relacionados con éstos.

Por lo general, los datos estarán estructurados, de acuerdo a algún modelo que determine la estructura lógica de éstos y permita su gestión. No obstante, también habrá ocasiones en las que se deban gestionar datos no estructurados, esto es, datos que no siguen ningún modelo, y que únicamente pueden ser considerados objetos o archivos binarios.

La arquitectura de datos establece normas relativas a los diferentes aspectos del diseño relacionados con los datos, tales como la determinación de su estructura, el mantenimiento de su integridad, y la especificación de los mecanismos que permiten su acceso y modificación de manera segura y eficiente.

6.1 Gestión de datos estructurados

Los datos estructurados se almacenarán en **bases de datos**. Las bases de datos relacionales organizan los datos de acuerdo a un modelo relacional, esto es, en forma de tablas relacionadas entre sí. La estructura concreta de estas tablas queda predefinida en lo que se denomina **esquema**.

No obstante, es posible que, para determinadas problemáticas, sean más convenientes bases de datos que no requieran un esquema fijo, como por ejemplo bases de datos documentales o bases de datos orientadas a grafos.

Norma AR-DAT-01

Se establece el uso de **bases de datos relacionales** como caso general para la gestión de datos estructurados.

6.1.1 Modelo lógico

La gestión de los datos estructurados comienza por la definición de un modelo lógico, en forma de **diagrama entidad-relación extendido**, el cual debe confeccionarse en tiempo de análisis funcional del sistema de información.

Norma AR-DAT-02

Cada sistema de información que maneje datos estructurados deberá definir un modelo lógico de éstos en forma de **diagrama entidad-relación extendido**, en tiempo de **análisis funcional**.

6.1.2 Modelo físico

El modelo físico se determinará a partir del modelo lógico, en tiempo de diseño detallado de cada proyecto aplicativo.

Norma AR-DAT-03

Cada subsistema aplicativo que maneje datos estructurados, deberá documentar el **modelo físico** de éstos, en tiempo de **diseño detallado**.

En el proceso de obtención del modelo físico a partir del modelo lógico, se deberá reducir la redundancia y la dependencia de la información, mediante su normalización.

La teoría de la normalización tiene por objetivo la eliminación de dependencias entre atributos que originen anomalías en la actualización de los datos, y proporcionar una estructura más regular para la representación de las tablas, constituyendo el soporte para el diseño de bases de datos relacionales.

Norma AR-DAT-04

El modelo físico deberá estar normalizado en **tercera forma normal**.

6.1.3 Esquema

El esquema de base de datos es la especificación del modelo físico de la base de datos mediante un *Data Definition Language*, lenguaje de definición de datos o DDL.

Norma AR-DAT-05

Cada subsistema aplicativo que maneje datos estructurados deberá disponer de **un único esquema**.

El esquema deberá incorporar las *constraints* o restricciones necesarias para garantizar la integridad de los datos. Se trata de reglas que restringen los posibles valores que puede tener un dato.

Norma AR-DAT-06

El esquema deberá incorporar las siguientes restricciones:

- Cada tabla deberá disponer de una **clave primaria**, que siempre será única y no nula.
- En función de cada caso, la clave primaria podrá ser una clave **natural** (formada a partir de datos de negocio), o una clave **subrogada** (que no tiene significado de negocio).
- Para las claves subrogadas se utilizarán valores auto-generados a partir de una **secuencia**, excepto en el caso de las tablas maestras, donde se usarán valores predefinidos.
- En la definición de las **columnas**, deberán usarse tipos de datos lo más específicos posible para representar el atributo correspondiente, y de un tamaño lo más restringido posible.
- Deberán especificarse **restricciones** específicas para las columnas que:
 - Constituyan una clave foránea.
 - Deban tener obligatoriamente valor y, en su caso, especificar un valor por defecto.
 - Su valor deba ser único, no pudiéndose dar dos veces el mismo valor en la columna.
 - Su valor pueda ser nulo.
 - Su valor deba cumplir reglas de negocio para que los datos tengan sentido, como por ejemplo ser positivo, ser posterior a cierta fecha, etcétera.

El esquema deberá incorporar los índices que se consideren necesarios para asegurar un buen rendimiento. Aunque algunos índices pueden crearse inicialmente, en la mayoría de las ocasiones los índices se crearán después de la implantación, para ajustar el rendimiento del sistema.

Importante

Deben tenerse en cuenta los siguientes consejos a la hora de crear índices:

- En caso de requerirse una migración de datos, es más eficiente crear los índices después de ésta. De lo contrario, la base de datos deberá actualizar cada índice según se inserta cada fila, y la migración requerirá mucho más tiempo.
- Crear índices solo cuando sea necesario:
 - Si frecuentemente se van a obtener menos del 15% de las filas en una tabla grande.
 - Crear índices en las columnas que constituyan claves foráneas y que estén involucradas en *joins* o búsquedas cruzadas.
 - Las tablas pequeñas no requieren índices.
- El orden de las columnas en la especificación del índice puede afectar al rendimiento. El orden ideal debería ser comenzando por la columna usada con más frecuencia.

- Cuantos más índices haya en una tabla, mayor sobrecarga existirá en escritura, ya que los índices deberán actualizarse cada vez que una fila se actualice o se elimine, llegándose a una situación de compromiso.

6.1.4 Control de acceso a la base de datos

Las bases de datos soportan mecanismos de control de acceso, basados en usuarios y contraseñas para el control de autenticación, y roles y permisos para la autorización.

Norma AR-DAT-07

Para proteger el ámbito del subsistema aplicativo, los únicos usuarios de la base de datos serán:

- Los **servidores de aplicaciones**, que deben contar con un rol específico y los permisos necesarios para poder ejecutar las operativas de negocio que proporcionan.
- Los **usuarios administradores** del entorno, que deben contar con un rol específico y los permisos necesarios para poder ejecutar las diferentes tareas de administración que sean necesarias, en el contexto de una intervención planificada.
- Los **usuarios automáticos de administración** que sean necesarios, para tareas automatizadas, tales como monitorización de negocio, exportaciones de datos planificadas, etc.

6.1.5 JDBC data source

Un *JDBC data source* u origen de datos, es una abstracción perteneciente a la especificación JDBC, que representa un *pool* o agrupación de conexiones hacia una fuente de datos física, tal como una base de datos.

Un origen de datos puede configurarse para que mantenga abiertas y disponibles cierto número de conexiones. De este modo, cuando una petición esté siendo tratada en el servidor de aplicaciones, no será necesario invertir tiempo en su establecimiento. Este tiempo no es despreciable, ya que incluye la apertura de la conexión y la autenticación.

Por otro lado, en una situación de concurrencia, donde varias peticiones están siendo tratadas a la vez en el servidor de aplicaciones, cada una de ellas puede utilizar su propia conexión. Según la implementación del origen de datos y su configuración, si no hay conexiones libres en el momento de requerirse una, puede abrirse una nueva, o bien esperar a que alguna quede disponible, o incluso lanzarse una excepción para protegerse del incremento de carga.

Norma AR-DAT-08

Los subsistemas aplicativos que manejen datos estructurados deberán disponer de un **origen de datos** específico en cada uno de sus servidores de aplicaciones.

La configuración del origen de datos contendrá, entre otros parámetros, la URL de acceso a la base de datos propia del subsistema aplicativo, el nombre de usuario y la contraseña, así como el driver de base de datos que se deberá emplear.

En todo caso, en la configuración del origen de datos, la contraseña deberá estar cifrada.

6.1.6 Entidades

Las entidades son objetos del modelo de dominio privado, que se caracterizan por tener una identidad, esto es, una referencia que las identifica únicamente, y porque su valor se puede almacenar o recuperar de una base de datos.

La especificación [Java Persistence API](#) define un conjunto de anotaciones con las que decorar clases Java, plasmando en ellas información acerca de la correspondencia entre un modelo de objetos y un modelo relacional.

Norma AR-DAT-09

La especificación de las entidades deberá realizarse y documentarse en tiempo de diseño detallado, y siempre de forma posterior a la especificación del esquema, del cual deberá derivarse.

Existen herramientas de ingeniería inversa, que permiten la generación automática del código Java de las entidades a partir de un esquema determinado.

Importante

La suite de herramientas [Hibernate Tools](#) proporciona herramientas de ingeniería inversa que permiten la generación automática del código Java de las entidades asociadas a una base de datos ya existente, desde el IDE de desarrollo.

Estas herramientas se configuran para acceder a la base de datos y leer sus metadatos, a partir de los cuales obtienen la información necesaria.

Norma AR-DAT-10

En caso de usarse herramientas de ingeniería inversa para generar el código fuente de las entidades, deben configurarse adecuadamente para customizar el código generado de modo que resulte usable.

Esta configuración se suele llevar a cabo mediante ficheros, en los que se debe especificar detalladamente:

- Las tablas que deberán incluirse en la ingeniería inversa.
- Mapeos específicos de tipos de datos entre SQL y Java.

- Mapeos específicos de tablas, incluyendo los nombres de las clases que se generarán.
- Mapeos específicos de columnas, incluyendo los nombres de los atributos que se generarán, su visibilidad, la visibilidad de sus métodos *get* y *set*, etc.
- Secciones Javadoc sobre clases y atributos.
- Generación automática de métodos *hashcode* y *equals* en base a los atributos que representan la clave primaria.

6.1.7 Unidad de persistencia

Una unidad de persistencia representa una agrupación lógica de clases persistentes, esto es, clases decoradas con las anotaciones anteriores, así como la especificación de los mecanismos de acceso a la base de datos correspondiente.

Norma AR-DAT-11

Los subsistemas aplicativos que manejen datos estructurados deberán definir una única unidad de persistencia, usando los mecanismos que proporciona Spring para ello.

Esta unidad de persistencia debe configurarse para acceder a la base de datos del subsistema aplicativo a través del origen de datos correspondiente en el servidor de aplicaciones.

6.1.8 Contexto de persistencia

Siguiendo el principio de inversión de control, los objetos de una unidad de persistencia necesitan de un contenedor que los administre, encargándose por tanto de gestionar la correspondencia entre el modelo de objetos y el modelo relacional.

El contexto de persistencia es una abstracción propia de la especificación [Java Persistence API](#), y que constituye este contenedor de inversión de control, especializado en la gestión de los objetos persistentes asociados a una determinada unidad de persistencia.

La gestión de estos objetos se basa en la instrumentación o manipulación de sus correspondientes clases, añadiendo código adicional al que fue programado, generalmente en tiempo de carga de clases.

Mediante esta instrumentación, los objetos quedan dotados de los mecanismos necesarios para controlar la correspondencia entre su estado, esto es, el valor de sus variables internas, y la entidad correspondiente en la base de datos.

Estos mecanismos permiten que los objetos estén conectados (*managed* o *attached*) a la base de datos durante el tiempo que dura la transacción. En ese tiempo, el contexto de persistencia actúa como un caché, indexando los objetos por clase y clave primaria.

Si se realiza un cambio sobre el estado de un objeto, dicho cambio automáticamente se trasladará a la base de datos, ya sea en ese mismo momento, o bien posteriormente, justo al finalizar la transacción.

Un contexto de persistencia tiene asociado un ámbito temporal, que puede ser de una transacción por petición, o bien un ámbito extendido, que permite que una misma transacción abarque varias peticiones.

Norma AR-DAT-12

El uso de contextos de persistencia con ámbito extendido está asociado al mantenimiento de un estado conversacional en la capa de persistencia.

Este tipo de diseños son muy propensos a errores y, además, son complicados de mantener. Por estos motivos, no se permite el uso de contextos de persistencia con ámbito extendido.

Cuando la transacción finaliza, todos los cambios pendientes sobre los objetos conectados se trasladan a la base de datos (*flush*), el caché se vacía (*close*), y los objetos que contenía dejan de estar administrados por el contexto de persistencia (*detached*). Aunque estos objetos aún contienen datos y puede accederse a ellos, ya no están conectados a la base de datos, por lo que cualquier intento de modificación de éstos, o bien de acceso a datos faltantes, terminará con un error.

6.1.9 Entity manager

El *entity manager*, o administrador de entidades, es un componente técnico cuyo interfaz queda normalizado por la especificación [Java Persistence API](#), y que permite al resto de componentes lógicos la realización de búsquedas relacionadas con una determinada unidad de persistencia, así como la manipulación de entidades a través de un contexto de persistencia, que gestiona internamente.

El administrador de entidades dispone de un ciclo de vida normalizado, el cual está relacionado con la hebra que atiende la petición, con la transacción vinculada a dicha hebra, y con el ámbito del contexto de persistencia subyacente.

Este ciclo de vida puede ser gestionado por parte del servidor de aplicaciones (*container-managed*), o bien por parte de la aplicación (*application-managed*).

Norma AR-DAT-13

Los subsistemas aplicativos que manejen datos estructurados deberán utilizar administradores de entidades gestionados por la aplicación, utilizando los mecanismos que proporciona Spring para ello.

6.1.10 Transaction manager

Una transacción representa una unidad de trabajo, compuesta por una secuencia de acciones, que se tratará de forma atómica, confiable, e independientemente de otras transacciones.

La delimitación de la transacción, esto es, la especificación de cuándo comienza y cuándo termina una transacción, se realiza de forma declarativa, asociándola al comienzo y fin de un método en un componente lógico, marcando dicho método como transaccional mediante una anotación específica.

De esta manera, con esta anotación se está indicando que todas las interacciones que se lleven a cabo desde dicho método contra uno o más recursos transaccionales, formarán parte de la misma transacción. Esta transacción comienza justo antes de la invocación del método, y termina en cuanto éste termine. Si el método terminase con una excepción no chequeada, la transacción se deshará automáticamente.

Siguiendo el principio de inversión de control, la ejecución de los métodos marcados como transaccionales necesita de un contenedor que se encargue de gestionar el ciclo de vida de cada transacción. El *transaction manager* o gestor de transacciones es el componente lógico encargado de dicha gestión.

Puesto que varias peticiones concurrentes pueden estar llamando al mismo método transaccional, cada petición debe poder trabajar con su propia transacción, independiente de las demás.

Cuando comienza la ejecución de un método marcado como transaccional, el gestor de transacciones crea un objeto que representa la transacción, y lo asocia a la hebra que está atendiendo la petición.

Cuando el método interactúa con un recurso transaccional, éste accede a la transacción asociada a la hebra actual, y se registra en ella para ser notificado acerca de su ciclo de vida.

El origen de datos en que se apoya el administrador de entidades anterior, constituye un recurso transaccional.

Norma AR-DAT-14

Los subsistemas aplicativos que manejen datos estructurados deberán utilizar los mecanismos que proporciona Spring para definir un gestor de transacciones que se encargue de la gestión de su ciclo de vida en las interacciones con recursos transaccionales.

En la unidad de persistencia, el tipo de transacción que deberá configurarse habrá de ser **RESOURCE_LOCAL**, ya que la base de datos será el único recurso transaccional con el que interactuará el subsistema aplicativo.

Importante

El gestor de transacciones de Spring no considera que la ocurrencia de una excepción chequeada deba motivar que la transacción se deshaga de manera automática.

Para provocar que la transacción se deshaga en caso de ocurrencia de una excepción chequeada, debe indicarse explícitamente.

6.1.11 Delimitación de transacciones

Los subsistemas aplicativos son responsables de mantener la integridad de los datos que manejan. En el caso de los datos estructurados, su integridad se mantiene mediante el **control transaccional**, el cual forma parte de la **lógica de negocio**, y no de la lógica de acceso a datos.

Por este motivo, las **operativas de negocio** deberán considerarse **transaccionales**.

Norma AR-DAT-15

El **control transaccional** de las operativas de negocio recae sobre los correspondientes métodos de las **fachadas de negocio**.

Los métodos de las fachadas de negocio que interactúen, directa o indirectamente, con recursos transaccionales, deberán marcarse como transaccionales. Para ello se empleará la anotación proporcionada a tal efecto por Spring.

De esta manera, si la operativa de negocio finaliza satisfactoriamente, los cambios son efectivamente comprometidos. Pero si finaliza con el lanzamiento de una excepción, dichos cambios deben ser efectivamente deshechos, para no poner en peligro la integridad de los datos.

Las transacciones así delimitadas deberán parametrizarse de la siguiente manera:

- **Aislamiento:** DEFAULT, indicando que se usará el nivel de aislamiento que se haya configurado por defecto en la base de datos.
- **Propagación:** REQUIRED, indicando que el método requiere que una transacción se haya iniciado y vinculado a la hebra de la petición antes de su ejecución. Si ya existiese una transacción abierta, la ejecución del método se realizará en el ámbito de dicha transacción. Si no, se abrirá una transacción justo antes de su ejecución.

Los componentes **DAO** que se ejecuten desde una operativa de negocio, deberán realizar sus acciones en el ámbito de la transacción asociada a dicha operativa de negocio.

6.1.12 Componentes DAO basados en Spring Data

Spring Data es un *framework* de la familia de Spring, que facilita enormemente la creación de componentes DAO (denominados *repository* o repositorio en Spring) que gestionen el acceso a una determinada entidad del modelo persistente.

Para ello, basta con definir un interfaz para el componente DAO, el cual deberá extender alguno de los que proporciona Spring Data y que ya proporcionan métodos básicos de acceso a datos, y añadir la declaración de los métodos de búsqueda que se requieran por parte de las distintas operativas de negocio para dicha entidad.

En tiempo de arranque, Spring Data proporcionará una implementación perfectamente funcional del componente DAO definido a partir de dicho interfaz, sin necesidad de escribir ninguna línea de código.

Aunque el uso de Spring Data con JPA es el caso más habitual, Spring Data también soporta el uso de otros mecanismos de persistencia.

Norma AR-DAT-16

Los componentes **DAO** de acceso a datos estructurados se definirán como *repository* mediante el *framework* **Spring Data**.

6.1.13 JPQL

Java Persistence Query Language, lenguaje de consultas de Java Persistence API o JPQL, es el lenguaje de consultas que proporciona la especificación [Java Persistence API](#) para la realización de búsquedas, a partir de los métodos que proporciona para ello el administrador de entidades.

Norma AR-DAT-17

Las consultas o modificaciones de datos realizadas por los componentes **DAO** de acceso a datos estructurados relacionales se especificarán mediante el lenguaje **JQPL**, con las excepciones contempladas en esta Arquitectura de Referencia.

6.1.14 Consultas

Cuando se utiliza Spring Data, las consultas pueden indicarse de dos formas: aplicando un convenio de nombres en los métodos de consulta que se definen en el interfaz que especifica al **DAO**, del cual se derivará automáticamente la consulta correspondiente, o bien indicando explícitamente la consulta, mediante una anotación específica, sobre el método.

El primer mecanismo es adecuado en casos muy simples, prefiriéndose el segundo para la mayoría de las ocasiones.

Norma AR-DAT-18

Las operaciones de consulta de datos en un **DAO** se definirán, preferentemente, mediante la anotación correspondiente de **Spring Data**.

En Spring Data, las operaciones de modificación en bloque se tratan de igual forma que las consultas, aunque es necesario indicar su naturaleza de operación de modificación con una anotación específica.

Norma AR-DAT-19

Las operaciones de modificación de datos en bloque no se deberán realizar modificando las diferentes entidades una a una, sino utilizando los mecanismos de modificación en bloque que proporciona Spring Data, y que deberán aplicarse sobre una operación específica en el **DAO** correspondiente.

6.1.15 Filtros de búsqueda

En un caso general, no todas las consultas pueden definirse en tiempo de diseño, existiendo casuísticas en las que únicamente es posible definir la consulta en tiempo de ejecución. Este suele ser el caso de los filtros de búsqueda, donde el usuario establece libremente los criterios por los que desea buscar, de entre un abanico de opciones predefinido.

Spring Data proporciona un interfaz tal que, si el interfaz que define el componente **DAO** lo extiende, proporciona mecanismos para definir filtros de búsqueda mediante una combinación lógica de *Specifications* o especificaciones.

Una especificación es un criterio de búsqueda predefinido, el cual puede incluir parámetros proporcionados por el usuario.

El filtro, esto es, la combinación lógica de especificaciones por la que finalmente se hará la búsqueda, se genera en tiempo de ejecución en función de los criterios de búsqueda seleccionados por el usuario.

Norma AR-DAT-20

Las consultas realizadas por componentes **DAO** de acceso a datos relacionales que deban soportar filtros de búsqueda se deberán implementar mediante una combinación lógica de especificaciones, compuesta en tiempo de ejecución mediante los mecanismos que proporciona Spring Data, a partir de los datos introducidos por el usuario.

Cada criterio de búsqueda disponible para el usuario deberá disponer de su especificación asociada.

6.1.16 Consultas exactas y consultas laxas

Una consulta exacta es aquella en la que los resultados obtenidos presentan una coincidencia exacta con respecto a los criterios de búsqueda empleados.

Por otro lado, una búsqueda laxa es aquella en la que se permite que los resultados obtenidos presenten una coincidencia aproximada con respecto a los criterios de

búsqueda empleados, con cierto grado de parecido con los valores de referencia dados.

Norma AR-DAT-21

Por defecto, todas las consultas serán exactas.

Cuando, por requisitos del subsistema aplicativo, se requiera que éste deba realizar **consultas laxas**, éstas deberán definirse mediante **SQL nativo**, empleando cualquiera de las funciones de distancia existentes en el paquete **UTL_MATCH** de Oracle.

Estas funciones permiten obtener resultados en base a la distancia o medida del parecido de un valor literal con una determinada columna alfanumérica, comparando dicha distancia o medida del parecido con determinado umbral. Este umbral habrá de ser predefinido por el subsistema aplicativo, y configurable.

Norma AR-DAT-22

Debido a su mal rendimiento, especialmente cuando la base de datos tiene cierto tamaño, no se permite el uso del operador **LIKE** en consultas.

6.1.17 Limitación de campos obtenidos

Por defecto, las consultas obtendrán todos los campos de cada entidad obtenida con un componente **DAO**, así como todas sus entidades relacionadas y sus correspondientes campos.

El volumen de datos obtenidos de esta forma suele exceder de las necesidades reales de la operativa de negocio, que generalmente no requiere más que un subconjunto específico y pequeño de datos, teniendo además un impacto negativo en el rendimiento, y consumiendo más recursos de los necesarios.

Norma AR-DAT-23

Por motivos de rendimiento, las consultas realizadas por componentes **DAO** de acceso a datos relacionales deberán obtener únicamente los datos requeridos por la operativa de negocio correspondiente.

Para especificar el conjunto de campos requerido como resultado de una consulta, se deberá definir un *entity graph* o grafo de entidades específico, indicando los campos que efectivamente se poblarán en el grafo conformado por la entidad y sus entidades relacionadas, y utilizando para ello los mecanismos que proporciona Spring Data.

Los grafos de entidades constituyen un mecanismo de especificación del modo de *fetch* de entidades relacionadas mucho más potente y versátil que el mecanismo tradicional en JPA, basado en indicarlo directamente en las anotaciones que especifican las relaciones en el modelo. Si con el mecanismo tradicional el modo

de *fetch* era estático, con los grafos de entidad el modo de *fetch* es específico para cada consulta.

6.1.18 Consultas con resultados mixtos

En ocasiones, existirán consultas tales que los datos que retornan no pueden modelarse en forma de una o varias entidades del mismo tipo.

Por ejemplo, presentarán resultados mixtos las consultas que retornan una combinación de columnas de diferentes entidades relacionadas, las consultas que contengan resultados agregados, etcétera.

Norma AR-DAT-24

Las consultas con resultados mixtos se resolverán mediante el mecanismo de *projections* o proyecciones que proporciona Spring Data.

Este mecanismo permite mapear los resultados obtenidos por una consulta sobre un objeto ligero, definido únicamente mediante un interfaz que disponga de los métodos *get* correspondientes a las columnas del resultado, según el nombre que se les haya dado a éstas en la consulta.

6.1.19 Paginación

Las operativas de negocio que impliquen la presentación al usuario, humano o automático, de un conjunto de resultados potencialmente grande, deberán diseñarse de tal forma que los datos retornados se encuentren paginados.

Norma AR-DAT-25

Las operativas de negocio que deban retornar un conjunto de resultados potencialmente grande, el cual vaya a ser presentado al usuario en la capa de presentación, deberán diseñarse de tal forma que proporcionen un **acceso paginado** a dichos resultados.

Por tanto, los correspondientes **métodos** de las **fachadas de negocio** deberán ser igualmente **paginados**, con un diseño tal que:

- Reciba, a través de sus parámetros, la indicación del número de página y del tamaño de página.
- Retorne un modelo de página de resultados, el cual contenga el subconjunto correspondiente del total de resultados, así como el número total de elementos y el número total de páginas.

El tamaño de página puede ser un valor predeterminado, o bien establecido por el usuario. En todo caso, el tamaño de página deberá estar limitado a un valor máximo y configurable en el subsistema aplicativo.

Por tanto, los correspondientes métodos de consulta en los componentes **DAO** implicados también habrán de ser paginados. Spring Data soporta la realización de búsquedas paginadas con opciones de ordenación.

Norma AR-DAT-26

Los métodos de consulta de los componentes **DAO** involucrados en operativas de negocio con acceso paginado, también deberán soportar paginación, usando los mecanismos que proporciona Spring Data para ello.

6.1.20 Depuración de consultas

En tiempo de construcción, es útil visualizar el código SQL generado por el administrador de entidades en tiempo de ejecución de las diferentes operativas de negocio, a través de las trazas que éste genera.

De esta manera se puede verificar que cada consulta obtiene únicamente las columnas que efectivamente debe obtener, a la vez que se facilita la determinación de las columnas sobre las que se deberán crear índices para mejorar el rendimiento.

Norma AR-DAT-27

La escritura de trazas conteniendo el código SQL generado internamente por el administrador de entidades se recomienda en tiempo de construcción, si bien deberá estar deshabilitada en los entornos de pre-producción y producción.

6.1.21 Mantenimiento de datos

En ocasiones, es preciso realizar acciones de mantenimiento de datos, tales como la consolidación de información entre diferentes sistemas o subsistemas, el purgado de datos obsoletos, el archivado de datos antiguos o la importación o exportación masivas de datos.

Norma AR-DAT-28

Las acciones de mantenimiento de datos estructurados deben realizarse en el contexto de una **tarea de mantenimiento** desde la capa de negocio, la cual deberá estar planificada para su ejecución en un **valle de carga**.

Los componentes **DAO** deberán proporcionar los métodos necesarios para facilitar la implementación de dicha tarea de mantenimiento.

El purgado y el archivado de datos se consideran especialmente relevantes, en tanto que permiten descargar volumen de las tablas que almacenan datos transaccionales, que de otro modo crecerían indefinidamente y provocarían una paulatina **degradación del rendimiento** del sistema a lo largo del tiempo.

El purgado de datos se entiende como la eliminación definitiva de datos no esenciales y que se consideran obsoletos. Esto es, datos que, pasado cierto

tiempo, ya no son útiles, y que, al considerarse no esenciales, deben eliminarse del operacional para no perjudicar el rendimiento.

El **archivado de datos** se entiende como la copia, ya sea exacta o sintética, de datos esenciales y que se consideran obsoletos. Esto es, datos que, pasado cierto tiempo, ya no son útiles, y que, al considerarse esenciales, deben moverse a **tablas específicas de archivo** para no perjudicar el rendimiento. Estas tablas de archivo pueden contener toda la información que tenían las entidades en las tablas transaccionales, o bien un mero resumen, y deberán estar configuradas para almacenar sus datos en un espacio físico separado de aquél o aquellos en los que se almacenan los datos transaccionales.

6.2 Gestión de datos no estructurados

Los datos no estructurados son aquellos cuya organización interna se desconoce, no sigue ningún modelo predefinido que permita un tratamiento específico de los datos que contiene o, simplemente, no es relevante.

6.2.1 Objetos binarios

Un bloque de datos no estructurados se gestionará como un todo, considerándose un objeto binario.

Norma AR-DAT-29

Los bloques de datos no estructurados se considerarán como objetos binarios.

Un objeto binario se considerará compuesto por:

- Un nombre.
- Un tipo de datos MIME, el cual podría deducirse a partir del nombre.
- Un bloque de bytes, el cual será accesible para lectura a través de flujos de bytes de entrada, y accesible para escritura a través de flujos de bytes de salida.

6.2.2 Persistencia de objetos binarios

La persistencia de objetos binarios se gestionará a través del sistema de archivos del servidor de aplicaciones.

Norma AR-DAT-30

La persistencia de objetos binarios se gestionará a través del sistema de ficheros del servidor de aplicaciones.

Cada objeto binario se asociará con un fichero independiente en el sistema de ficheros.

6.2.3 Procesamiento de objetos binarios

El procesamiento de un objeto binario, en el contexto de la ejecución de una operativa de negocio, suele abarcar diferentes etapas.

En cada etapa del procesamiento, se tendrá como resultado uno o más nuevos objetos binarios, cada uno de ellos almacenado en su correspondiente fichero.

Al finalizar el procesamiento, se habrán conseguido uno o más ficheros. Algunos de ellos serán finales, esto es, los que se pretendía obtener como resultado del procesamiento, mientras que otros serán intermedios o temporales.

6.2.4 Ficheros temporales

Los ficheros intermedios o temporales ya no serán útiles una vez finalizado el procesamiento, de modo que deberán eliminarse.

Norma AR-DAT-31

Se utilizarán ficheros temporales para almacenar los resultados de las etapas intermedias de procesamiento de las correspondientes operativas de negocio.

Como parte de la última etapa del procesamiento, todos los ficheros temporales que se hayan generado deberán eliminarse.

6.2.5 Unidad de almacenamiento para ficheros finales

Los ficheros finales, obtenidos como resultado del procesamiento, deberán almacenarse en una unidad de almacenamiento compartida, que será considerada el almacén de datos operacionales para los datos no estructurados que maneje el subsistema aplicativo.

Norma AR-DAT-32

Los ficheros finales generados como resultado del procesamiento de objetos binarios realizado por las correspondientes operativas de negocio serán persistidos en una unidad de almacenamiento.

Esta unidad de almacenamiento será considerada el almacén de datos operacionales para los datos no estructurados que maneje el subsistema aplicativo, y será compartida por los servidores de aplicaciones del subsistema aplicativo, y accesible por todos ellos desde la misma ruta en el sistema de ficheros.

6.2.6 Control de acceso a la unidad de almacenamiento

El sistema de ficheros soporta mecanismos de control de acceso, basados en los usuarios y grupos del sistema, y los permisos de lectura, escritura o ejecución sobre archivos y directorios.

Norma AR-DAT-33

Para proteger el ámbito del subsistema aplicativo, los únicos usuarios de la unidad de almacenamiento donde éste almacene objetos binarios serán:

- Los **servidores de aplicaciones**, que deben contar con un usuario específico y los permisos necesarios para poder ejecutar las operativas de negocio que proporcionan.
- Los **usuarios administradores del entorno**, que deben contar con un usuario o grupo específico y los permisos necesarios para poder ejecutar las diferentes tareas de administración que sean necesarias, en el contexto de una intervención planificada.
- Los **usuarios automáticos de administración** que sean necesarios, para tareas como monitorización de negocio, exportaciones de datos automatizadas, etc.

6.2.7 Organización en carpetas

Los ficheros manejados por un subsistema aplicativo, ya sean temporales o finales, pueden organizarse en estructuras de carpetas para facilitar su clasificación.

Norma AR-DAT-34

Los subsistemas aplicativos que manejen datos no estructurados, podrán organizar sus correspondientes ficheros finales en estructuras de carpetas, para facilitar su clasificación.

Norma AR-DAT-35

Cuando sea necesario disponer de carpetas temporales de trabajo separadas por petición, éstas se deberán denominar mediante un UUID.

Como parte de la última etapa del procesamiento, la carpeta temporal de trabajo, con todo su contenido, deberá eliminarse.

6.2.8 Componentes DAO de acceso a datos no estructurados

Los componentes **DAO** de acceso a datos no estructurados deberán proporcionar funcionalidades básicas de creación, acceso, modificación y eliminación de objetos binarios como se indica a continuación.

Norma AR-DAT-36

Los componentes **DAO** de acceso a datos no estructurados estarán vinculados de forma lógica a un determinado **directorio**.

Proporcionarán, como mínimo, las funcionalidades básicas de creación, acceso, modificación y eliminación de objetos binarios, los cuales se considerarán como ficheros en dicho directorio.

En función de la complejidad de la estructura de carpetas con que se organicen los ficheros, puede ser necesario disponer de una factoría de componentes **DAO**, encargada de resolver la carpeta a partir de los parámetros que se precisen, y proporcionar el componente **DAO** correspondiente.

6.2.9 Mantenimiento de datos

En ocasiones, es preciso realizar acciones de mantenimiento de datos, tales como la consolidación de información entre diferentes sistemas o subsistemas, el purgado de datos obsoletos, el archivado de datos antiguos o la importación o exportación masiva de datos.

Norma AR-DAT-37

Las acciones de mantenimiento de datos no estructurados deben realizarse en el contexto de una **tarea de mantenimiento** desde la capa de negocio, la cual deberá estar planificada para su ejecución en un valle de carga.

Los componentes **DAO** de acceso a datos no estructurados deberán proporcionar los métodos necesarios para facilitar la implementación de dicha tarea de mantenimiento.

6.2.10 Almacenamiento de objetos binarios en la base de datos

De manera excepcional, pueden almacenarse objetos binarios en la base de datos, cuando se cumplan los requisitos siguientes.

Norma AR-DAT-38

Los objetos binarios podrán almacenarse excepcionalmente en base de datos únicamente cuando se cumplan las siguientes condiciones:

- Que el subsistema aplicativo también vaya a manejar datos estructurados y, por tanto, se disponga ya de una base de datos asociada.
- Que el número total de objetos binarios manejados por el subsistema aplicativo vaya a ser reducido.
- Que el tamaño de cada objeto binario sea pequeño, inferior a 512 KB.

En este escenario, los objetos binarios se deberán modelar como una **entidad**, que en todo caso deberá contemplar:

- Un nombre.

- Un tipo de datos MIME, el cual podría deducirse a partir del nombre.
- Un bloque de bytes, en forma de **BLOB** o **CLOB**, el cual será accesible para lectura a través de flujos de bytes de entrada, y accesible para escritura a través de flujos de bytes de salida.

En caso de no cumplirse las restricciones anteriores, los objetos binarios deberán tratarse de la forma habitual, como ficheros.

7 Arquitectura del interfaz de usuario

La arquitectura del interfaz de usuario establece normas relativas a los diferentes aspectos del diseño relacionados con interfaces de usuario Web provistos por los subsistemas aplicativos.

7.1 Control de la visualización desde el servidor

Existen dos alternativas fundamentales para controlar la visualización de un interfaz de usuario Web: llevarlo a cabo desde el lado del cliente, o bien desde el lado del servidor.

El control de la visualización desde el lado del cliente se basa en la captura de eventos generados por la interacción del usuario con la página Web, el manejo de éstos en el cliente, mediante llamadas a diferentes APIs y, como resultado, la modificación dinámica del código HTML, todo ello controlado desde un script, que se ejecuta en el navegador del usuario.

En todo momento la página HTML es la misma, siguiendo un modelo de *Single Page Application*, aplicación de una única página o SPA.

Este esquema de control es ideal para crear interfaces de usuario Web altamente dinámicas y visualmente muy ricas. Sin embargo, como contrapartida, se consumirán más recursos en el terminal del cliente, podrían presentarse más problemas de compatibilidad, al apoyarse fuertemente en scripts, y podrían proporcionar una mala experiencia de usuario o baja accesibilidad, especialmente cuando existan conexiones más lentas.

El control de la visualización desde el lado del servidor se basa en la captura de eventos generados por la interacción del usuario con la página Web y el manejo de éstos en el servidor, provocando como resultado la creación de una nueva página Web, que se visualizará en el navegador del usuario, y con la que éste podrá interactuar.

Se trata, por tanto, del esquema tradicional de control de la visualización, y es ideal para crear interfaces de usuario sencillas, que no requieren de un gran dinamismo, o que tienen requisitos especiales de accesibilidad, de posicionamiento SEO, de la variedad de terminales con los que ser compatible o de rendimiento de ejecución sobre ellos. Sin embargo, como contrapartida, y aparte de las limitaciones en dinamismo y riqueza visual, ha de tenerse en cuenta que se consumirán más recursos en el servidor.

La gran mayoría de interfaces de usuario Web en el Ministerio de Justicia no requieren de un gran dinamismo ni de una gran riqueza visual. Se basan fundamentalmente en completar y enviar formularios o realizar búsquedas y visualizar tablas de resultados. Además, aplican sobre ellos requisitos especiales de accesibilidad.

Por estos motivos, se opta por imponer un control de la visualización desde el servidor, mediante el uso de componentes lógicos con estereotipo Controlador, ubicados en la capa Web.

7.2 Diseño RESTful de interfaces de usuario Web

Los interfaces de usuario Web, al igual que las APIs REST, pueden seguir un diseño RESTful, aunque adaptado a sus particularidades, empleando HTML como formato de representación.

Este enfoque de diseño se caracteriza por plantear las operativas provistas por el subsistema aplicativo, y que estén expuestas a través del interfaz Web, mediante un conjunto de **operaciones básicas** (de creación, lectura, sobre-escritura y eliminación) que es posible aplicar sobre determinado conjunto de **recursos**.

Estos recursos se corresponden con objetos del **modelo de dominio público** o externo del subsistema aplicativo, y se ubican, de manera lógica, bajo un determinado path en el esquema de navegación del interfaz Web.

Las ventajas de utilizar un diseño RESTful en interfaces Web son, entre otras, facilitar el diseño sistemático, no necesitar el uso de la sesión para almacenar el estado conversacional, mejorar la navegabilidad general del interfaz, facilitar la configuración de seguridad, y disponer de una mucho mejor trazabilidad de los accesos.

Norma AR-GUI-01

Los **interfaces Web** provistos por cualquier subsistema aplicativo deberán seguir un **diseño RESTful**, según los convenios establecidos en la **Tabla 7-1** y **Tabla 7-3**, donde los recursos contemplados se corresponderán con objetos del **modelo de dominio público** o externo.

7.2.1 Tipos de recurso

Se consideran dos tipos de recurso:

- **Colecciones**, que deberán tener asociado en el path un nombre en **plural**. Por ejemplo, **/pedidos**.
- **Elementos** concretos en el ámbito de una colección, que deberán tener asociado un identificador único, el cual deberá reflejarse en el path. Por ejemplo, **/pedidos/{id}**.

Adicionalmente, y para adaptar este modelo a la representación de los recursos en HTML, es necesario contemplar adicionalmente dos **recursos ficticios**:

- Formulario de creación de un nuevo elemento en una colección. Por ejemplo, **/pedidos/form**.
- Formulario de modificación de un elemento concreto en el ámbito de una colección. Por ejemplo, **/pedidos/{id}/form**.

7.2.2 Primitivas

Sobre los recursos que exponga el interfaz Web será posible realizar cuatro posibles operaciones primitivas, que se corresponden con los verbos del protocolo HTTP GET, POST, PUT y DELETE.

Path	Verbo	Descripción
/[colección]	GET	Consulta y muestra los elementos de la colección, representados en forma de tabla
	POST	Crea un nuevo elemento en la colección, y re-direcciona a /[colección]/{id}
/[colección]/form	GET	Muestra un formulario para solicitar la creación de un nuevo elemento de la colección, pudiendo recibir en la query el valor de algunos de sus campos
/[colección]/{id}	GET	Muestra un elemento de la colección
	PUT	Modifica un elemento de la colección, y re-direcciona a /[colección]/{id}
	DELETE	Elimina un elemento de la colección, y re-direcciona /[colección]
/[colección]/{id}/form	GET	Muestra un formulario para solicitar la modificación de un elemento de la colección

Tabla 7-1 Convenios de diseño RESTful de interfaces Web

Nótese que existe un paralelismo entre el diseño RESTful y el diseño tradicional de modelos de datos relacionales, de manera que /[colección] se corresponde con una tabla, y /[colección]/{id} se corresponde con una fila concreta en dicha tabla, identificada por su clave primaria.

7.2.3 Recursos anidados

En ocasiones, un elemento de una colección puede incluir, a su vez, otras colecciones de elementos. Por ejemplo, un pedido /pedidos/{id} puede contener una serie de ítems /pedidos/{id}/ítems.

Como puede observarse, dichas colecciones anidadas se ubican lógicamente bajo el path del elemento que las contiene. En la representación HTML del detalle de un pedido puede optarse por incluir una tabla reflejando sus ítems asociados.

Volviendo al paralelismo existente entre el diseño RESTful y el diseño de modelos de datos relacionales, se trataría en este caso de una entidad fuerte (pedidos), relacionada con una entidad débil (ítems), con una cardinalidad 1:N.

Otros modelos de relación entre recursos deberán sustentarse en la definición de campos de relación o referencia, a modo de claves foráneas, cuyo valor sea el path del elemento relacionado.

7.2.4 Consultas y paginación

La representación de colecciones deberá entenderse como una consulta, la cual deberá ser paginada cuando el conjunto de resultados sea potencialmente grande, como se muestra a continuación.

Campo 1	Campo 2	Campo 3	Campo 4
opción 1 <input type="button" value="▼"/>	100 <input type="button" value="▼"/>	Valor <input type="text"/>	Valor <input type="text"/>
		<input type="button" value="Limpiar"/>	<input style="background-color: #0070C0; color: white; border-radius: 5px; padding: 2px 10px; font-weight: bold; border: none;" type="button" value="Buscar"/>

Columna 1	Columna 2	Columna 3	Columna 4
Elemento 1	Valor	Valor	Valor
Elemento 2	Valor	Valor	Valor
Elemento 3	Valor	Valor	Valor
Elemento 4	Valor	Valor	Valor
Elemento 5	Valor	Valor	Valor
Elemento 6	Valor	Valor	Valor
Elemento 7	Valor	Valor	Valor
Elemento 8	Valor	Valor	Valor
Elemento 9	Valor	Valor	Valor
Elemento 10	Valor	Valor	Valor

[**<< Anterior**](#) [**1 2 ... Página 3 ... Siguiente >>**](#)

Ilustración 7-1 Representación de colecciones como consultas paginadas

Para ello se habrán de emplear en la **query** los parámetros normalizados que se indican a continuación.

Parámetro y formato	Descripción
sort=[columna],[ASC DESC]	Criterio de ordenación de los resultados en base a una columna, con sentido ascendente (por defecto) o descendente
page=[página]	Número de página, comenzando en 0
size=[tamaño]	Tamaño de página

Tabla 7-2 Parámetros de consulta y paginación

Norma AR-GUI-02

La representación HTML de una **colección** debe entenderse como una **consulta** sobre los elementos que la conforman, la cual deberá ser **paginada** cuando el conjunto de resultados sea potencialmente grande, en base a los parámetros de

query normalizados definidos en la **Tabla 7-2**, junto con los correspondientes a cada campo de búsqueda presente en el formulario.

Dicha representación HTML se considerará compuesta por un formulario de consulta, una tabla de resultados y un pie de paginación, de acuerdo con la **Ilustración 7-1**. Eventualmente, la representación podrá incluir un enlace al formulario de creación de nuevos elementos en la colección.

El formulario de consulta podrá disponer de uno o más campos de búsqueda, los cuales deberán reflejar el valor indicado en el parámetro de query correspondiente. El formulario de consulta podrá no visualizarse si los criterios de búsqueda están predefinidos y no requieren, por tanto, de la introducción de datos por el usuario.

La tabla de resultados deberá disponer de una fila de encabezado, y podrá disponer de una o más columnas. La tabla de resultados podrá no visualizarse si no se indica ningún criterio de búsqueda mediante los correspondientes parámetros de query.

Si la consulta soporta la ordenación de resultados, la fila de encabezado deberá mostrar los nombres de las columnas como enlaces a la misma consulta y página, añadiendo además mediante el parámetro **sort** el criterio de ordenación por la columna seleccionada y el sentido por defecto, o bien el contrario al actual, si el criterio de ordenación fuese la misma columna.

La tabla de resultados deberá listar aquéllos correspondientes al número de página y tamaño de página indicado en los parámetros **page** y **size**. El tamaño de página puede ser un valor predeterminado, o bien establecido por el usuario. En todo caso, el tamaño de página deberá estar limitado a un valor máximo y configurable en el subsistema aplicativo.

La tabla de resultados puede incluir en cada fila controles de acción sobre el elemento correspondiente.

El pie de paginación debe incluir el número de página actual, un enlace a las páginas siguiente y anterior, así como a la primera y a la última página.

7.2.5 Jerarquías de clases de recurso

En ocasiones podría surgir la necesidad de distinguir jerarquías de clases de recurso.

Por ejemplo, se puede plantear disponer de una colección de formularios, en el contexto de un interfaz Web orientado a ciudadanos y provisto por un sistema de información que realiza la tramitación de un procedimiento administrativo.

Dicho interfaz deberá soportar la creación de varias clases de formularios:

- Formularios de **solicitud de iniciación**: para iniciar el procedimiento.
- Formularios de **subsanación** de solicitud: para aportar datos faltantes o corregir incorrectos en una solicitud de iniciación.
- Formularios de **alegación**: para contestar a requerimientos y aportar documentación adicional durante la tramitación o en el trámite de audiencia.
- Formularios de **comunicación de cambio de datos** de notificación: para informar de cambios en los datos de notificación o en el medio de notificación.

Para modelar este escenario, se recomienda soportar la jerarquía en base a un atributo, que refleje el **tipo** o subclase de recurso.

Volviendo a la comparación con el diseño de modelos de datos relacionales, se correspondería con el modelado de una jerarquía de entidades en una única tabla, añadiendo una columna para discriminar el tipo.

En el ejemplo anterior, los posibles valores de este atributo podrían ser **solicitud-inicio**, **subsanación-solicitud**, **alegación** y **comunicación-cambio-datos-notificación**.

Finalmente, es necesario indicar el tipo de recurso que se desea crear, en tiempo de solicitar el correspondiente formulario de creación mediante **/[colección]/form**. Para ello se habrá de emplear en la **query** el parámetro normalizado que se indican a continuación.

Parámetro y formato	Descripción
type=[tipo]	Tipo o subclase de recurso para el que se solicita el formulario de creación

Tabla 7-3 Parámetro de tipo o subclase de recurso

El resultado de dicha petición deberá ser una vista representando el formulario de creación del elemento, especializado en el tipo de recurso correspondiente. Dicho formulario deberá reflejar este dato mediante un campo específico, para que pueda ser tenido en cuenta en tiempo de la posterior creación del elemento.

7.2.6 Diagrama de esquema de navegación

El esquema de navegación puede representarse mediante un diagrama de clases, donde cada clase represente un recurso, y las relaciones entre ellas representen un segmento de path que, eventualmente, puede disponer de variables. Las clases disponen de métodos, asociados a los distintos verbos HTTP.

La siguiente ilustración especifica el esquema de navegación del interfaz Web del ejemplo anterior.

En color azul se modelan los formularios con los que es posible interactuar con el procedimiento, distinguiendo las subclases que se mencionaban anteriormente.

En color gris se modela el recurso ficticio para la creación de formularios, el cual podrá recibir a través de la query el tipo de formulario solicitado, así como el identificador de expediente asociado. Los formularios distintos de las solicitudes de iniciación siempre deben estar vinculados a un expediente creado previamente.

En color verde se modelan los registros, abstracción cuya creación representa el acto de registrar en el registro electrónico de la AGE un formulario, que previamente ha sido creado y firmado.

El formulario de creación de un registro se encuentra embebido en la vista de detalle del formulario, no requiriéndose en este caso de recursos ficticios para su creación.

En color naranja se modelan los expedientes, así como los documentos contenidos en ellos. Se creará un nuevo expediente de manera implícita con el registro de un formulario de solicitud de iniciación, de modo que no se requieren de recursos ficticios para su creación.

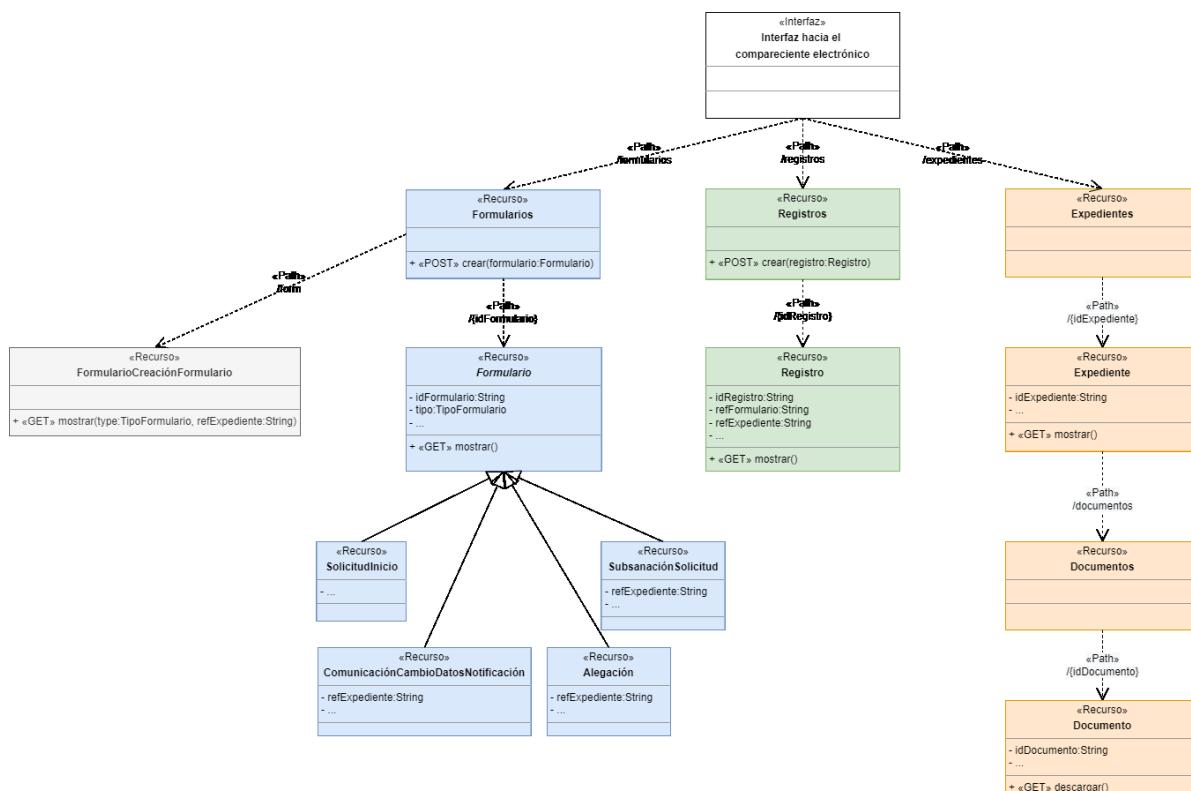


Ilustración 7-2 Diagrama de esquema de navegación

Norma AR-GUI-03

Cada subsistema aplicativo que provea interfaces Web, deberá documentar el **diagrama de esquema de navegación** de éstos, en tiempo de **diseño detallado**, mediante un diagrama de clases similar al de la **Ilustración 7-2**.

Dicho diagrama deberá reflejar, como nodo raíz, el propio interfaz Web, así como los diferentes recursos contemplados en su diseño RESTful, incluyendo sus posibles jerarquías, sus atributos más relevantes, las operaciones disponibles para cada uno y su método HTTP asociado, los diferentes recursos ficticios existentes, así como las relaciones existentes entre recursos a través de paths.

7.3 Internacionalización

Los interfaces Web que se publiquen a través del puerto público del entorno deberán ser internacionalizados.

Norma AR-GUI-04

Los interfaces Web provistos a través del puerto público deben estar **internacionalizados**, debiéndose soportar, como mínimo, las lenguas oficiales:

- Castellano (es)²
- Catalán (ca)³
- Euskera (eu)⁴
- Gallego (gl)⁵
- Occitano-aranés (oc)⁶
- Valenciano (vlca)⁷

Los campos de entrada de texto libre, a lo largo de todo el interfaz, deben igualmente soportar los idiomas anteriores.

El control del idioma de presentación deberá basarse en la elección del usuario, y mantenerse entre peticiones mediante una **cookie** específica.

7.4 Responsividad

Los interfaces Web que se publiquen a través del puerto público deberán disponer de un diseño responsivo que, en todo caso, permita la correcta visualización e interacción desde terminales móviles.

Norma AR-GUI-05

Los interfaces Web provistos a través del puerto público deben disponer de un diseño **responsivo**, que permita su correcta visualización y una experiencia de usuario adecuada desde terminales móviles con pantallas pequeñas como *smartphones* o *tablets*, equipos portátiles con pantallas intermedias, o equipos de sobremesa con pantallas grandes.

7.5 Accesibilidad

Todas las vistas de un interfaz Web, especialmente aquellos que se publiquen a través del puerto público, deberán ser accesibles.

² Lengua oficial en todo el territorio español

³ Lengua oficial en Cataluña e Islas Baleares, de acuerdo a sus respectivos Estatutos de Autonomía

⁴ Lengua oficial en País Vasco y Navarra, de acuerdo a sus respectivos Estatutos de Autonomía

⁵ Lengua oficial en Galicia, de acuerdo a su Estatuto de Autonomía

⁶ Lengua oficial en Cataluña, de acuerdo a su Estatuto de Autonomía

⁷ Lengua oficial en la Comunidad Valenciana, de acuerdo a su Estatuto de Autonomía

Norma AR-GUI-06

Todas las vistas de un interfaz Web provisto a través del puerto público deben cumplir, por lo menos, con el nivel **AA** de la norma [WCAG 2.1](#)

Importante

[The Wave](#) es una herramienta gratuita de validación que se basa en las normas del WCAG 2.1 y en la Sección 508 de EEUU. Está desarrollado por [Web Accessibility in Mind](#) o WebAIM, organización sin ánimo de lucro del [Center for Persons with Disabilities](#) de la [Universidad del Estado de Utah](#).

The Wave se distribuye en forma de herramienta Web online, o bien como extensión de Firefox o Chrome.

Importante

El [Observatorio de Accesibilidad Web](#) proporciona en su [Guía de validación de accesibilidad Web](#) información adicional en materia de accesibilidad y referencias a diferentes recursos.

7.6 Normalización de la apariencia y experiencia de usuario

Los interfaces Web deberán disponer de un estilo visual y experiencia de usuario normalizados.

Norma AR-GUI-07

Los interfaces Web deberán disponer de un estilo visual y experiencia de usuario conformes a la **Guía de estilo de interfaces Web**, donde se indican las líneas guía del diseño visual del interfaz, la especificación de la apariencia de los componentes gráficos más relevantes y de uso más frecuente, las pautas de disposición de éstos en los distintos contenedores, tales como formularios, secciones, divisiones o páginas, así como el modo de interacción del usuario con ellos y su comportamiento.

7.6.1 Maqueta HTML

Las pautas de estilo visual y experiencia de usuario definidas en la guía anterior deben plasmarse en forma de una maqueta HTML navegable, las cuales deberán ser el punto de partida para la construcción de interfaces Web provistos por los subsistemas aplicativos.

Norma AR-GUI-08

La **construcción** de interfaces Web en los subsistemas aplicativos deberá partir de una **maqueta** HTML navegable de carácter reutilizable, la cual deberá cumplir

con los requisitos anteriores de **apariencia y experiencia de usuario, diseño responsive y accesibilidad**.

Los **recursos estáticos** asociados a la maqueta se considerarán de carácter **compartido**, debiendo estar disponibles en los servidores frontales de los diferentes subsistemas aplicativos.

La versión vigente en cada momento de la **Guía de estilo de interfaces Web**, de su **maqueta HTML** asociada y de los **recursos estáticos** asociados a ella queda determinada por la **Política de actualización de estilo de interfaces Web**.

7.6.2 Motor de plantillas

El hecho de partir de maquetas HTML para la construcción de interfaces Web determina la conveniencia de emplear un motor de plantillas para la representación de las vistas que permita tener un control total del código HTML generado.

Thymeleaf es un motor de plantillas compatible con Spring MVC, y que permite el uso de HTML como lenguaje base para definirlas. La lógica de presentación se plasma en ellas mediante el uso de atributos propios, no estándar de HTML. Esto permite visualizar las plantillas en cualquier navegador, aunque la ejecución de la lógica de presentación plasmada en ellas únicamente podrá llevarse a cabo desde Thymeleaf.

Norma AR-GUI-09

Se establece el uso de [Thymeleaf](#) como motor de plantillas para la representación de vistas Web.

8 Arquitectura de seguridad

En el modelo de sistema, un **interfaz** es una abstracción que se materializa en el conjunto de mensajes intercambiados entre el sistema de información y sus usuarios, ya sean usuarios humanos, usando un navegador Web, u otros sistemas de información que actúen como usuarios automáticos, mediante servicios Web.

Por otro lado, un **puerto** representa el punto de acceso a través del cual es posible realizar el intercambio de mensajes anterior, y se materializa en un conjunto de nodos y comunicaciones, propios de la arquitectura física, junto con su configuración necesaria, así como en los componentes lógicos, de carácter técnico, necesarios para gestionar el intercambio de dichos mensajes, propios de la arquitectura lógica.

Finalmente, el **ámbito del sistema** representa la frontera entre los elementos que conforman el sistema de información, tanto físicos como lógicos, y el exterior.

Esta frontera debe estar protegida, así como la información custodiada por el sistema e intercambiada a través de ella. Para lograrlo, es necesario establecer distintas medidas de protección, tanto de ámbito organizativo, como operacional y técnico.

La arquitectura de seguridad establece las medidas de protección específicas que deberán estar presentes en el diseño arquitectónico de los sistemas de información que sean conformes a esta Arquitectura de Referencia.

Importante

Estas medidas de protección tienen carácter de mínimo general. Su cumplimiento no exime de otras responsabilidades derivadas del [Esquema Nacional de Seguridad](#), las cuales pueden motivar medidas de protección adicionales no contempladas en este documento.

8.1 Sistemas o subsistemas frontera

Los sistemas de información se descomponen en varios subsistemas aplicativos, los cuales ya incorporan diversas medidas de protección, algunas derivadas del propio modelo de sistema, y otras plasmadas en su arquitectura física.

Sin embargo, es necesario establecer medidas de protección adicionales, para lograr así una estrategia basada en disponer de diferentes líneas de defensa superpuestas que reduzca el riesgo de ataque.

Algunas de estas medidas de protección recaerán en elementos concretos, físicos o lógicos, de los subsistemas que se ubiquen en la **frontera** con el exterior. Otras, sin embargo, aplicarán a todos los subsistemas aplicativos que conforman el sistema de información.

A efectos de seguridad, se consideran **sistemas o subsistemas frontera**, aquellos que provean **interfaces**, ya sean Web o WS, que sean **directamente accesibles** desde cualquiera de los **puertos** del entorno de producción, incluyendo por tanto al Middleware de integración, tanto el interno como el externo.

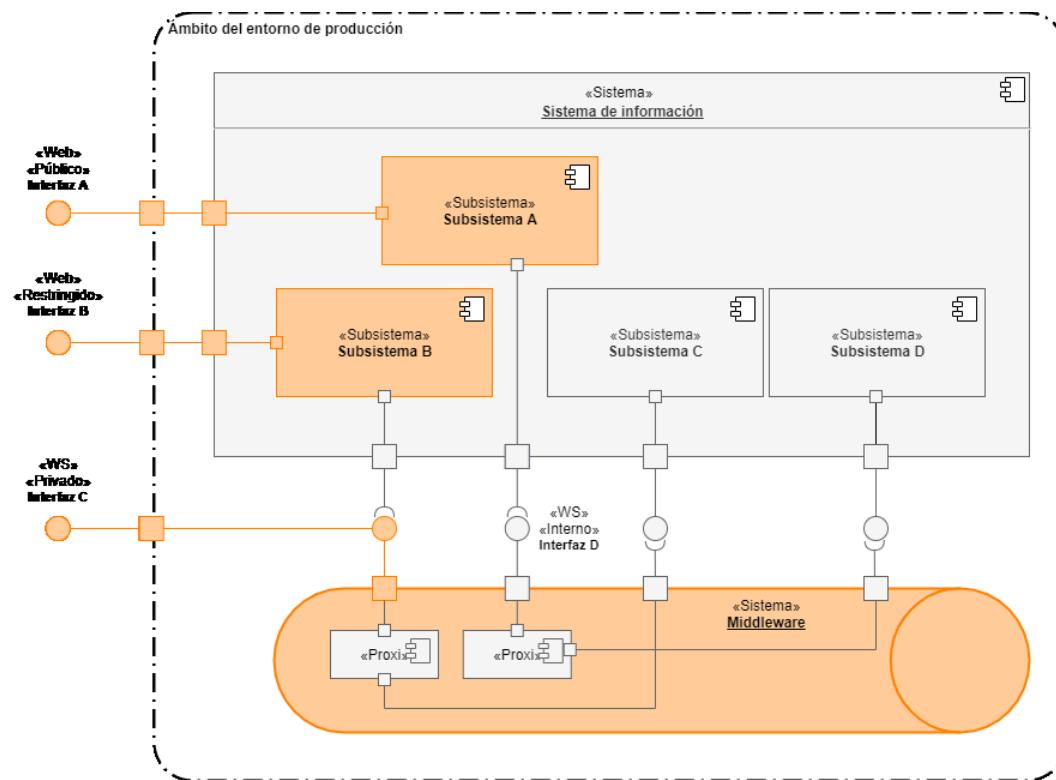


Ilustración 8-1 Sistemas y subsistemas frontera

8.2 Interfaces y niveles de acceso

Se cataloga el **nivel de acceso** de un **interfaz provisto** por un subsistema aplicativo en función del **puerto** del entorno de producción desde donde será accesible, como se indica a continuación.

Norma AR-SEG-01

Cada interfaz provisto por un subsistema aplicativo debe tener asociado un **nivel de acceso** específico, en función del puerto del entorno de producción desde donde será accesible.

- **Nivel de acceso público:** el interfaz provisto es accesible desde el puerto público del entorno de producción, el cual es accesible a través de Internet.
- **Nivel de acceso restringido:** el interfaz provisto es accesible desde el puerto restringido del entorno de producción, el cual es accesible a otras Administraciones Públicas externas al Ministerio de Justicia a través de la red SARA/TESTA.

- **Nivel de acceso privado:** el interfaz provisto es accesible desde el puerto privado del entorno de producción, el cual es accesible únicamente desde determinadas unidades del Ministerio de Justicia y/o sus organismos adscritos.
- **Nivel de acceso interno:** el interfaz provisto no es accesible desde ningún puerto del entorno de producción.

Norma AR-SEG-02

Los interfaces Web deben conectarse al **puerto** del entorno **lo más restrictivo posible**, en función de los usuarios hacia los que se orienta la funcionalidad que proveen.

Norma AR-SEG-03

Se establece el uso de [Spring Security](#) como *framework* base para la implementación de los mecanismos de seguridad.

8.3 Open Web Application Security Project

El *Open Web Application Security Project*, proyecto abierto de seguridad de aplicaciones Web, o OWASP, es un proyecto de código abierto dedicado a determinar y combatir las causas que hacen que el software sea inseguro.

La [Fundación OWASP](#) es un organismo sin ánimo de lucro que apoya y gestiona los proyectos e infraestructura de OWASP.

Periódicamente, la Fundación OWASP publica un documento [OWASP Top 10](#), con las diez vulnerabilidades más críticas de las aplicaciones Web. En el momento de la redacción de este documento, la última versión es la correspondiente a 2021.

Las medidas de protección que determinan la arquitectura de seguridad, y que se desarrollan en los apartados siguientes, vienen derivadas del análisis de estas diez vulnerabilidades más críticas.

8.3.1 Rotura de control de acceso

En muchos sistemas de información, las restricciones sobre lo que los usuarios autenticados pueden hacer, no se aplican correctamente. Los atacantes pueden explotar estos defectos para acceder, de forma no autorizada, a determinadas funcionalidades o datos.

8.3.1.1 Control de autorización

Los recursos accesibles a través de interfaces Web o WS expuestos por los sistemas frontera deben protegerse con mecanismos de control de autorización.

Norma AR-SEG-04

Los **interfaces Web** o **WS** que se consideren **autenticados** deberán incorporar los mecanismos de **control de autorización** proporcionados por Spring Security, aplicándolos sobre la capa de presentación.

Adicionalmente, deben aplicarse mecanismos de control de autorización sobre los elementos gráficos de las vistas.

Estos mecanismos se basan en la especificación de las condiciones que ha de cumplir el sujeto o *principal* para que efectivamente pueda acceder al recurso solicitado, o bien visualizar el fragmento de vista correspondiente. Dichas condiciones se basan, a su vez, en las aserciones obtenidas acerca de su identidad y sus autorizaciones, en tiempo de autenticación.

En caso de que la evaluación de dichas condiciones no fuese satisfactoria, se lanzará una excepción y se impedirá el acceso al recurso solicitado, o bien no se mostrará el fragmento de vista correspondiente.

Norma AR-SEG-05

En tiempo de diseño detallado, se ha de determinar el conjunto de **roles** de que dispondrá el subsistema aplicativo.

En todo caso, se han de limitar los permisos de cada usuario al mínimo estrictamente necesario para acceder a la información requerida y para cumplir sus obligaciones.

8.3.2 Fallos criptográficos

Muchos sistemas de información no protegen adecuadamente datos sensibles, tales como información financiera, de salud, etcétera. Los atacantes pueden robar o modificar estos datos protegidos inadecuadamente para llevar a cabo fraudes con tarjetas de crédito, robos de identidad u otros delitos. Los datos sensibles requieren métodos de protección adicionales, como el cifrado en almacenamiento y tránsito.

8.3.2.1 Transport Layer Security

Los servidores frontales deberán ser accesibles desde el exterior a través del protocolo HTTPS sobre TLS, como se indica a continuación.

Norma AR-SEG-06

Los **servidores frontales externos e internos** deberán configurarse para que la comunicación se realice la forma que se indica a continuación, en función del tipo de interfaz expuesto:

- **Interfaces Web** accesibles desde el puerto público, restringido o privado, provistos y expuestos al exterior por subsistemas aplicativos: **TLS 1.3** o superior, con **certificado de sitio Web**.

- **Interfaces WS** accesibles desde el puerto público, restringido o privado, provistos por subsistemas aplicativos y expuestos al exterior a través del middleware: **TLS 1.3 o superior**, con **certificado de sitio Web** (servidor) y **certificado de sello** (cliente).
- **Interfaces WS** internos, provistos por subsistemas aplicativos y no accesibles desde el exterior del ámbito del entorno de producción: **TLS 1.3 o superior**, con **certificado de sitio Web**.

En ningún caso se debe soportar a la vez **distintas versiones** del protocolo TLS.

8.3.2.2 HTTP Strict Transport Security

HTTP Strict Transport Security, seguridad de transporte estricta HTTP o HSTS, es un mecanismo que, entre otras cosas, re-direcciona automáticamente al cliente cuando intenta acceder a un recurso a través de HTTP, para que lo haga a través de HTTPS de forma segura.

Norma AR-SEG-07

Los **sistemas y subsistemas frontera** deberán configurarse para utilizar el mecanismo **HSTS** como se indica a continuación:

- **Max-Age**: indica el tiempo durante el cual estará activo el mecanismo. Debe indicarse un año.
- **IncludeSubDomains**: debe indicarse, para que queden protegidos igualmente los subdominios.

Los subsistemas aplicativos frontera deberán emplear para ello los mecanismos que proporciona Spring Security.

8.3.2.3 Identificadores de recurso

Los esquemas de URL en diseños RESTful de interfaces Web no deben reflejar datos sensibles.

Norma AR-SEG-08

Los esquemas de URL en **diseños RESTful de interfaces Web** no deben reflejar **datos sensibles**, tales como datos personales o datos confidenciales, en los **identificadores de recurso**, teniendo sentido de clave subrogada.

8.3.3 Inyección

Los ataques de inyección de código, como SQL, NoSQL, OS o LDAP, ocurren cuando se envían datos no confiables a un intérprete, como parte de un comando o consulta. Los datos dañinos del atacante pueden engañar al intérprete para que ejecute comandos involuntarios o acceda a los datos sin la debida autorización.

Para prevenir este tipo de ataque, se habrán de implementar los mecanismos que se indican a continuación.

8.3.3.1 Validación sintáctica de los datos de entrada

Los datos de entrada que se reciban desde fuera del ámbito de cada subsistema aplicativo han de estar definidos sintácticamente, estableciendo para cada uno de ellos unas reglas de formato específicas.

Norma AR-SEG-09

La **especificación sintáctica** de cada uno de los datos recibidos desde el exterior se realizará en tiempo de diseño detallado, para todos los subsistemas aplicativos.

En todo caso, para cada dato recibido, la especificación sintáctica deberá incluir:

- **Tipo de datos**, que deberá ser lo más restringido posible.
- **Obligatoriedad** o no de que se informe su valor.
- Posibles **restricciones de longitud** que puedan aplicar sobre su valor, pudiéndose establecer un límite mínimo, máximo, o ambos.
- **Rango o enumeración** de posibles valores.
- **Posible patrón** que deba cumplir su valor.

Estas reglas de formato han de ser validadas de forma previa a la ejecución de la operativa de negocio correspondiente.

Norma AR-SEG-10

La **validación sintáctica** de los datos recibidos se realizará mediante la evaluación de las reglas que se hayan definido, sobre los datos recibidos.

En el caso de datos recibidos a través de **interfaces Web provistos**, la validación sintáctica deberá realizarse mediante en la **capa Web**.

En el caso de datos recibidos a través de **interfaces WS provistos**, la validación sintáctica deberá realizarse en la **capa de servicios Web**, mediante la validación de los mensajes recibidos contra los **XML Schema** que acompañan al **WSDL** que define el servicio Web correspondiente, donde dichas reglas deberán estar reflejadas.

8.3.3.2 Validación semántica de los datos de entrada

Los datos de entrada que se reciban desde fuera del ámbito de cada subsistema aplicativo han de estar definidos también semánticamente, estableciendo para cada uno de ellos unas reglas específicas de validación en el contexto de la lógica de negocio.

Por ejemplo, si se dispone de un dato de entrada que representa una fecha de fin, entonces esta habrá de ser anterior a la fecha de inicio.

Norma AR-SEG-11

La **especificación semántica** de cada uno de los datos recibidos desde el exterior se realizará en tiempo de diseño detallado, para todos los subsistemas aplicativos. Esta especificación semántica se expresará en forma de reglas de validación del valor de ciertos datos en relación al valor de otros, de forma que el conjunto tenga sentido.

Estas reglas han de ser validadas de forma previa a la ejecución de la operativa de negocio correspondiente.

Norma AR-SEG-12

La **validación semántica** de los datos recibidos se realizará mediante la evaluación de las reglas que se hayan definido, sobre los datos recibidos, en la **capa de presentación**.

8.3.3.3 Validación de archivos

Si un subsistema aplicativo debe procesar archivos recibidos desde el exterior de su ámbito, ya sean adjuntos a peticiones Web o WS, o bien accedidos desde una tarea de mantenimiento, se deben realizar las validaciones que se indican a continuación.

Norma AR-SEG-13

En el caso de que un subsistema aplicativo deba procesar **archivos** recibidos desde el **exterior** de su ámbito, ya sean adjuntos a peticiones Web o WS, o bien accedidos desde una tarea de mantenimiento, debe validar:

- Validación del **número** de **archivos** recibidos, contra un número máximo configurable, o bien predeterminado.

Para cada uno de los archivos que se reciban, deben realizarse las siguientes validaciones, desde la **capa de presentación**:

- Validación del **tipo MIME** del archivo, contra una lista de tipos MIME admitidos.
- Validación de la **extensión** del fichero, comprobándose que se corresponde con el tipo MIME indicado.
- Validación del **tamaño** del archivo, contra un tamaño máximo establecido y configurable.

Para cada uno de los archivos que se reciban, deben realizarse las siguientes validaciones, desde la **fachada de negocio**:

- Validación del **contenido** del archivo contra el **Servicio de Antivirus**, que forma parte de la Plataforma de servicios compartidos.

Si se detectase una violación de dichas reglas, se generará una **excepción**, que se habrá de tratar de la forma habitual.

8.3.3.4 Codificación de datos

Los ataques XSS ocurren cuando un sistema de información toma datos no confiables y los envía al navegador Web sin una validación y codificación apropiada.

Este tipo de ataque permite ejecutar comandos en el navegador Web de la víctima, permitiendo al atacante secuestrar una sesión, modificar un sitio Web, o redireccionar al usuario hacia un sitio malicioso.

Una forma eficaz de evitar los ataques XSS es emplear una adecuada codificación de los datos al generar el código HTML de las vistas.

Norma AR-SEG-14

Las **vistas** deberán utilizar **codificadores de datos y secuencias de escape** al incluir datos que hayan sido recibidos del usuario (directa o indirectamente, a través de la base de datos), o bien de otros sistemas externos.

8.3.3.5 Cabecera Content-Security-Policy

La cabecera Content-Security-Policy o CSP, constituye un mecanismo adicional de seguridad que contribuye a detectar y mitigar ataques de XSS y de inyección.

Norma AR-SEG-15

Los **subsistemas aplicativos frontera** que expongan interfaces Web deberán configurarse para generar la cabecera **Content-Security-Policy** en sus respuestas, indicando una política que restrinja al navegador Web a:

- Descargar recursos únicamente del dominio asociado al subsistema aplicativo, incluidos los subdominios que puedan requerirse.
- Descargar recursos, diferenciados por su tipo, de dominios de confianza.

Para ello deberán emplearse los mecanismos provistos por Spring Security.

8.3.4 Diseño inseguro

No tener en cuenta la seguridad desde el diseño provoca que, en tiempo de construcción, no se dote al sistema de información de todos los mecanismos de protección necesarios.

8.3.4.1 No mostrar información que facilite el fingerprinting

El *fingerprinting*, literalmente toma de huellas dactilares, consiste en recolectar diversos datos acerca de un sistema de información, para aprender acerca de su configuración y funcionamiento, con objeto de organizar un ataque.

Para contribuir a evitarlo, se han de aplicar los mecanismos que se indican a continuación.

Norma AR-SEG-16

Los **archivos generados** automáticamente por subsistemas aplicativos no deben contener metadatos que proporcionen información acerca del sistema o de los mecanismos empleados para su generación.

Norma AR-SEG-17

Los **sistemas y subsistemas frontera** no deben incluir la cabecera **Server** en sus respuestas.

8.3.4.2 No exponer información en campos ocultos

El uso de campos de tipo oculto o *hidden* en formularios Web no evita que un atacante pueda acceder a la información que contienen.

Norma AR-SEG-18

Se debe **evitar el uso** de campos de tipo oculto o *hidden* en formularios Web para el intercambio de datos de negocio.

8.3.4.3 Inventariar la información de carácter personal

Los subsistemas aplicativos que realicen tratamientos de datos de carácter personal deben documentar dicho tratamiento como se indica a continuación.

Norma AR-SEG-19

En el **diseño arquitectónico** del sistema de información debe documentarse el **tratamiento** o tratamientos de datos personales realizados éste. La información para ello deberá obtenerse del [Registro de Actividades de Tratamiento del Ministerio de Justicia](#).

Si el sistema de información debe realizar un tratamiento de datos personales que no haya sido contemplado en el documento anterior, este hecho debe notificarse al responsable del sistema, de forma que sea elevado al responsable del tratamiento correspondiente.

Norma AR-SEG-20

En tiempo de diseño detallado, debe documentarse un **inventario** de los **datos personales** concretos que maneja cada subsistema aplicativo, y dónde localizarlos.

8.3.4.4 Derecho de información acerca del tratamiento de datos personales

Los interesados tienen derecho a ser informados acerca del tratamiento de sus datos personales en tiempo de recogida de datos. Esta información se debe presentar de forma jerárquica, mostrando un bloque de información básica, y proporcionando un enlace a la información completa.

Norma AR-SEG-21

La **información básica** acerca del **tratamiento** debe mostrarse al usuario en los interfaces Web en tiempo de recogida de datos, **al pie del formulario** correspondiente, para dar soporte al derecho de información.

Esta información se encuentra compilada en el documento de [Información básica acerca de los tratamientos](#).

Adicionalmente, debe proporcionarse un **enlace** al [Registro de Actividades de Tratamiento del Ministerio de Justicia](#) para que el interesado, si lo desea, acceda a la **información completa** acerca del tratamiento de datos personales.

8.3.5 Configuración de seguridad incorrecta

La configuración de seguridad incorrecta es un problema muy común, y se debe, en parte, a establecer la configuración de forma manual, ad hoc, por omisión o, directamente, por la falta de configuración.

Para contribuir a prevenir este tipo de ataques, se han de aplicar los mecanismos que se indican a continuación.

8.3.5.1 No mostrar información técnica al usuario al tratar excepciones

En el tratamiento de excepciones, no debe proporcionarse información técnica, ya que puede facilitar la generación de ataques.

Norma AR-SEG-22

En el tratamiento de **excepciones**, no se debe proporcionar al usuario información técnica de ningún tipo, ya sea a través de interfaces Web o WS.

En ningún caso se debe presentar el **stacktrace** o volcado de pila de la excepción lanzada con motivo del error.

8.3.5.2 Protección de las cookies

Las cookies, ya sean de personalización (por ejemplo, la elección de idioma) o para el mantenimiento de la sesión, deben ser protegidas con los atributos que se indican a continuación.

Norma AR-SEG-23

Las **cookies** que se puedan generar deberán disponer de los siguientes atributos:

- **Secure**: el navegador sólo enviará la cookie si la conexión es segura (TLS)
- **HttpOnly**: el navegador no permitirá acceder a la cookie a los scripts que se ejecuten en el cliente.
- **SameSite**: el navegador no enviará la cookie a otros sitios Web.
- **Domain**: no establecer este atributo. En esa situación, la cookie solo se enviará al mismo servidor que la creó.
- **Path**: path base del subsistema aplicativo.
- **Expires**: fecha de vencimiento, en cookies no persistentes.
- **Max-Age**: fecha de vencimiento, en cookies persistentes.

8.3.5.3 Inhabilitar el uso de entidades externas XML

Muchos procesadores XML antiguos, o mal configurados, evalúan referencias a entidades externas en documentos XML.

Las entidades externas pueden utilizarse para revelar archivos internos mediante la URI o archivos internos en servidores no actualizados, escanear puertos de la LAN, ejecutar código de forma remota y realizar ataques de denegación de servicio.

Para contribuir a prevenir este tipo de ataques, se han de aplicar los mecanismos que se indican a continuación.

Norma AR-SEG-24

Los *parsers* o procesadores de XML se han de configurar de modo que se inhabilite el uso de DTD.

8.3.6 Componentes vulnerables y anticuados

Los sistemas de información utilizan librerías, y se ejecutan sobre plataformas, que pueden presentar vulnerabilidades, las cuales pueden debilitar las defensas del sistema de información y permitir diversos ataques.

Importante

La Política de actualización de la línea base de plataformas determina la última versión vigente de la línea base de plataformas.

Por otro lado, la **Política de actualización de APIs, frameworks y librerías** determina la última versión vigente de la línea base de librerías.

Ambas políticas incluyen además los **procedimientos** para revisar la línea base correspondiente, en los que se tiene en cuenta el estudio de vulnerabilidades conocidas, así como la periodicidad con la que se revisarán.

Sin embargo, corresponde a los responsables de cada sistema de información actualizar éste periódicamente, alineándolo con la última versión vigente de ambas líneas base, para reducir los riesgos de seguridad y contribuir a evitar la obsolescencia.

8.3.7 Fallos de identificación y autenticación

A menudo, los mecanismos que dan soporte a la autenticación y gestión de sesiones son implementados incorrectamente, permitiendo a los atacantes comprometer usuarios y contraseñas, identificadores de sesión, o explotar otras fallas de implementación para asumir la identidad de otros usuarios, ya sea temporal o permanentemente.

8.3.7.1 Interfaces anónimos y autenticados

En primer lugar, es necesario determinar si la información que el sistema vaya a proporcionar a través de un determinado interfaz requiere o no de autenticación, y con qué grado de confianza.

Norma AR-SEG-25

En tiempo de **diseño arquitectónico**, los **interfaces provistos** por un subsistema aplicativo **frontera** se deben catalogar, en función del grado de autenticación necesario para el acceso a la información que proporcionan, de acuerdo a las siguientes categorías:

- **Anónimo**: toda la información que proporciona el subsistema aplicativo a través del interfaz se considera de libre acceso.
- **Autenticado**: alguna parte, o bien toda, la información que proporciona el subsistema aplicativo a través del interfaz requiere de un control de autenticación que asegure, con cierto grado de confianza, la identidad del otro extremo de la comunicación.

Norma AR-SEG-26

Los **interfaces Web anónimos** provistos por subsistemas aplicativos **frontera** accesibles desde el **puerto público** deberán protegerse contra ataques de denegación de servicio mediante el uso de un mecanismo de **CAPTCHA**.

En caso de que un interfaz provisto sea considerado como autenticado, se deben incorporar mecanismos de control de autenticación sobre él, como se indica a continuación.

Norma AR-SEG-27

Los **sistemas o subsistemas frontera** que expongan **interfaces autenticados** deben incorporar mecanismos de **control de autenticación** que aseguren la autenticidad del otro extremo del canal de comunicación, con cierto grado de confianza, antes de intercambiar ninguna información.

La elección de los mecanismos que podrán emplearse, así como el tipo de credencial, dependerá del **tipo de interfaz** expuesto, de su **nivel de acceso**, y del **grado de confianza** requerido en dicha autenticación, de acuerdo con la **Tabla 8-1**.

Este grado de confianza se corresponde con el nivel asociado a la dimensión **Autenticidad**, que se obtiene en la realización del análisis de impacto que impone el [Esquema Nacional de Seguridad](#).

Los sistemas de información **no deben implementar mecanismos de control de autenticación diferentes a los anteriores**, ni deben proporcionar otras vías alternativas para obtener la información.

Tipo de interfaz	Nivel de acceso	Nivel ENS A	Mecanismo de autenticación	Tipo de credencial
Web	Público	Bajo	Cl@ve	Clave PIN
				Clave Permanente sin OTP
				Clave Permanente con OTP
			Autentica	Certificado SW
				DNI electrónico
				Certificado HW
		Medio	Cl@ve	Contraseña
				Certificado SW
				Certificado HW
	Alto	Cl@ve	Autentica	Clave PIN
				Clave Permanente con OTP
				Certificado SW
	Restringido	Bajo	Autentica	DNI electrónico
				Certificado HW
				Certificado HW
	Medio	Autentica	Autentica	Autentica
				Contraseña
				Certificado SW
				Certificado HW
				Certificado SW

			Certificado HW
		Alto	Autentica
	Privado	Bajo	Autentica
			Contraseña
			Certificado SW
			Certificado HW
			Active Directory
		Medio	Autentica
			Contraseña
			Certificado SW
			Certificado HW
		Alto	Autentica
WS	Público	N/A	TLS
	Restringido	N/A	TLS
			Certificado cliente y servidor
			Certificado cliente y servidor

Tabla 8-1 Determinación del mecanismo de autenticación y el tipo de credencial

La autenticación en interfaces WS se controla a nivel de transporte en el Middleware, mediante el uso de certificados de servidor y cliente en las conexiones **TLS**.

En interfaces Web, la autenticación puede realizarse contra el directorio activo corporativo, o bien delegarse a proveedores de identificación externos, en concreto a los sistemas [Cl@ve](#) (orientado a la autenticación de ciudadanos) o [Autentica](#) (orientado a la autenticación de empleados públicos).

Ambos sistemas utilizan el protocolo SAML 2.0, el cual coordina un diálogo entre el sistema de información, el proveedor de identidad, y el usuario que se autentica, para realizar el proceso de autenticación de forma segura.

Cuando el cliente solicita cualquier recurso a un interfaz Web que requiere autenticación, éste comprueba si la autenticación se ha realizado previamente. En caso de que no sea así, le responde con una redirección al proveedor de autenticación. Esta redirección incluye un mensaje SAML de petición cifrado, el cual contiene distintos datos acerca del subsistema aplicativo, entre ellos, el grado de confianza requerido en la autenticación, así como la URL donde espera recibir posteriormente el mensaje SAML de respuesta.

El cliente, al seguir la redirección, envía el mensaje SAML de petición al proveedor de identidad. Éste descifra el mensaje, asegurándose de que proviene de un subsistema aplicativo en el que confía, y lo procesa, para iniciar con el cliente un diálogo Web solicitándole sus credenciales y validándolas. Si las credenciales son válidas, el proveedor de identidad envía al cliente una cookie, en torno a la cual se articula un mecanismo de sesión SSO para facilitar próximas autenticaciones del mismo cliente con dicho proveedor de identidad.

Una vez finalizada la validación de credenciales, el proveedor de identidad genera un mensaje SAML de respuesta cifrado, incluyendo distintos datos o aserciones acerca de la identidad del sujeto que se ha autenticado. Este mensaje se incluye en una redirección al subsistema aplicativo, usando para ello la URL indicada por éste en la petición.

El cliente, al seguir la redirección, envía el mensaje SAML de respuesta al subsistema aplicativo. Éste descifra el mensaje, asegurándose de que proviene de un proveedor de identidad en el que confía, y lo procesa, analizando las aserciones recibidas acerca del sujeto o *principal* autenticado.

En este momento, la autenticación ha terminado, y el subsistema aplicativo envía al cliente una redirección al recurso que solicitó inicialmente, y que motivó todo este diálogo Web. Puesto que en esta ocasión sí se ha realizado una autenticación previa, el cliente podrá acceder sin problemas al recurso solicitado.

Más adelante, el cliente puede solicitar el cierre de la sesión SSO con el proveedor de identidad desde el subsistema aplicativo, con un sistema similar de mensajes SAML intercambiados mediante redirecciones.

8.3.7.2 Control de autenticación basado en Cl@ve

Cl@ve es una pasarela basada en un [perfil SAML específico, denominado eIDAS](#), enfocado al intercambio de identidades digitales en el ámbito europeo. En este perfil se determinan diferentes atributos adicionales a los definidos en SAML 2.0, y que definen el modelo de identidad. Por su parte, Cl@ve añade atributos adicionales propios.

eIDAS requiere especificar un *Level of Assurance*, nivel de confianza o LoA mínimo que debe cumplir la autenticación solicitada. En la siguiente tabla se especifica la correspondencia entre el nivel de seguridad en la dimensión Autenticación según el Esquema Nacional de Seguridad, y el LoA eIDAS que es necesario indicar en la petición a Cl@ve.

Nivel Autenticación ENS	Level of Assurance eIDAS
Bajo	Low
Medio	Substantial
Alto	High

Tabla 8-2 Determinación del LoA de Cl@ve

Los siguientes atributos deben estar presentes en la petición, indicando que se desea obtener su valor en la respuesta:

Atributo
CurrentFamilyName
CurrentGivenName
PersonIdentifier
SelectedIdP
FirstSurname
PartialAfirma

Tabla 8-3 Atributos presentes en la petición SAML de Cl@ve

A continuación, el usuario es re-direccinado a Cl@ve, para que elija el proveedor de identificación que deseé entre los que se encuentren disponibles, y presente sus credenciales, en función del LoA establecido.

Los proveedores de identificación posibles son los siguientes:

Proveedor de Identificación	Descripción
AEATIdP	Claves concertadas Cl@ve PIN ⁸
GISSIdP	Claves concertadas Cl@ve Permanente ⁸
AFIRMAIdP	Certificado SW o HW @firma ⁸
eIDASIdP	Nodo de interconexión eIDAS ⁹

Tabla 8-4 Proveedores de identificación disponibles en Cl@ve

Cuando Cl@ve re-direcciona de nuevo al usuario al subsistema aplicativo, éste debe extraer de la respuesta SAML los datos siguientes:

Atributo	Descripción
PersonIdentifier	NIF ¹⁰ o bien [Cpo]/[Cpd]/[id] ¹¹
CurrentFamilyName	Nombre
CurrentGivenName	Apellidos, concatenados
FirstSurname	Primer apellido
SelectedIdP	Proveedor de identificación elegido
PartialAfirma	Datos formateados del certificado digital ¹²

Tabla 8-5 Atributos presentes en la respuesta SAML de Cl@ve

En función del proveedor de identificación elegido y, eventualmente, la clase de certificado empleado (tal como se indica en el valor del atributo PartialAfirma), puede determinarse el tipo de *principal* como se indica a continuación.

Proveedor de identificación	Clase de certificado	Tipo de Principal
eIDAS	N/A	Persona física
Cl@ve PIN	N/A	Persona física
Cl@ve Permanente	N/A	Persona física

⁸ Únicamente para ciudadanos españoles.

⁹ Pasarela que permite a ciudadanos de países de la unión europea elegir un proveedor de identificación de su país de origen.

¹⁰ Solo si el proveedor de identificación elegido fue AEATIdP, GISSIdP o AFIRMAIdP.

¹¹ Solo si el proveedor de identificación elegido fue EIDASIdP. Se trata de un identificador único, aunque no persistente (esto es, el ciudadano podría tener varios identificadores diferentes a lo largo de su vida). [Cpo] y [Cpd] se corresponden con los códigos ISO 3166-1 alpha-2 de país origen y destino del identificador. [id] contiene el valor del identificador, siendo de formato y longitud diferentes en función del país que lo emite. El tamaño máximo del identificador único completo es de 256 caracteres.

¹² Solo si el proveedor de identificación elegido fue AFIRMAIdP.

@firma	0	Persona física
	5	Empleado público
	11	Persona física representante de persona jurídica
	12	Persona física representante de entidad sin personalidad jurídica

Tabla 8-6 Determinación de la clase de Principal en Cl@ve

Finalmente, el subsistema aplicativo deberá generar un objeto autenticación para ser almacenado en el contexto de seguridad de Spring Security. Este objeto autenticación debe contener las siguientes autorizaciones, en función del tipo de *principal* correspondiente.

Tipo de Principal	Autorizaciones
Persona física	ROLE_CIUDADANO ROLE_PERSONA_FISICA
Empleado público	ROLE_CIUDADANO ROLE_EMPLEADO_PUBLICO
Persona física representante de persona jurídica	ROLE_CIUDADANO ROLE_PF_REPRESENTANTE_PJ
Persona física representante de entidad sin personalidad jurídica	ROLE_CIUDADANO ROLE_PF_REPRESENTANTE_ESPJ

Tabla 8-7 Autorizaciones a incluir en la autenticación con Cl@ve

Cada tipo de *principal* deberá reflejar los campos especificados a continuación, cuyo valor se deberá determinar a partir del atributo o atributos SAML indicados.

Tipo de Principal	Campo	Origen
Persona física	Id	PersonIdentifier
	Tipo de id ¹³	SelectedIdP
	Nombre	CurrentFamilyName
	Primer apellido	CurrentGivenName FirstSurname
	Segundo apellido	CurrentGivenName FirstSurname
	Correo electrónico	PartialAfirmation
Empleado público	Id	PersonIdentifier
	Tipo de id	NIF
	Nombre	CurrentFamilyName
	Primer apellido	CurrentGivenName FirstSurname
	Segundo apellido	CurrentGivenName FirstSurname

¹³ NIF o eIDAS.

	Correo electrónico	PartialAfirma
	Número de identificación personal	PartialAfirma
	NIF organismo	PartialAfirma
	Denominación organismo	PartialAfirma
	DIR3 organismo	PartialAfirma
	Unidad	PartialAfirma
	Puesto	PartialAfirma
	Persona física representante de persona jurídica	
	Id	PersonIdentifier
	Tipo de id	NIF
	Nombre	CurrentFamilyName
	Primer apellido	CurrentGivenName FirstSurname
	Segundo apellido	CurrentGivenName FirstSurname
	Correo electrónico	PartialAfirma
	NIF entidad	PartialAfirma
	Razón social entidad	PartialAfirma
	Persona física representante de entidad sin personalidad jurídica	
	Id	PersonIdentifier
	Tipo de id	NIF
	Nombre	CurrentFamilyName
	Primer apellido	CurrentGivenName FirstSurname
	Segundo apellido	CurrentGivenName FirstSurname
	Correo electrónico	PartialAfirma
	NIF entidad	PartialAfirma
	Razón social entidad	PartialAfirma

Tabla 8-8 Estructura del Principal en la autenticación con Cl@ve

Norma AR-SEG-28

Los **sistemas o subsistemas frontera** que expongan **interfaces Web autenticados** con nivel de acceso **público** y orientado a **ciudadanos**, deberán disponer de un mecanismo de control de autenticación basado en Cl@ve.

Dicho mecanismo deberá componer el mensaje SAML de petición de acuerdo a la **Tabla 8-2** y **Tabla 8-3**.

Los mensajes SAML de respuesta correcta deberán procesarse para extraer los datos especificados en la **Tabla 8-5**. A partir de ellos, se determinará el tipo de Principal correspondiente según la **Tabla 8-6**.

Finalmente, deberá construirse un objeto Authentication reflejando el resultado de la autenticación, el cual deberá incluir las autorizaciones especificadas en la tabla **Tabla 8-7** y un objeto Principal reflejando los datos especificados en la tabla **Tabla 8-8**, en función de su tipo.

8.3.7.3 Control de autenticación basado en Autentica

[Autentica](#) es un repositorio horizontal de empleados públicos, altos cargos y personal relacionado, que provee mecanismos de control de acceso para aplicaciones internas en el ámbito de las Administraciones Públicas. Autentica posee un interfaz SAML 2.0 para autenticación, al cual añade un atributo específico contenido toda la información en la que deberá basarse el control de autorización.

En Autentica, el nivel de confianza requerido no se establece en la petición SAML, sino que se predefine para el subsistema aplicativo en tiempo de alta de éste en la integración con Autentica.

En la siguiente tabla se especifica la correspondencia entre el nivel de seguridad en la dimensión Autenticación según el Esquema Nacional de Seguridad, y el nivel de seguridad en Autentica.

Nivel Autenticación ENS	Nivel de seguridad en Autentica
Bajo	Nivel 2 ¹⁴
Medio	Nivel 3
Alto	Nivel 4

Tabla 8-9 Determinación del nivel de seguridad en Autentica

Cuando Autentica re-direcciona de nuevo al usuario al subsistema aplicativo, éste debe extraer de la respuesta SAML los datos siguientes:

Atributo	Descripción
AUTENTICA_USER_XML	Datos formateados del usuario

Tabla 8-10 Atributos presentes en la respuesta SAML de Autentica

Finalmente, el subsistema aplicativo deberá generar un objeto Authentication para ser almacenado en el contexto de seguridad de Spring Security. Este objeto Authentication debe incluir un objeto Principal reflejando los datos acerca del usuario, así como las autorizaciones indicadas en los datos formateados recibidos en el atributo AUTENTICA_USER_XML, debiéndose tener en cuenta únicamente los correspondientes al subsistema aplicativo llamante.

Autentica utiliza un modelo de autorización que distingue ámbitos, en los cuales se ubican perfiles (autorizaciones de grano grueso) y roles (autorizaciones de grano fino). Para generar sus equivalentes en Spring Security se deberá seguir el siguiente convenio:

¹⁴ Debe establecerse el modo de funcionamiento en que, en primer lugar, Autentica intenta usar el certificado del navegador y, si no es posible, entonces solicita usuario y contraseña. La contraseña deberá ser fuerte.

Elemento	Autorización en Spring Security
Ámbito	SCOPE_[ámbito]
Perfil	ROLE_[ámbito]_[perfil]
Rol	ROLE_[ámbito]_[perfil]_[rol]

Tabla 8-11 Correspondencia entre las autorizaciones de Autentica y Spring Security

El tipo de *principal*, en el caso de Autentica, en todo caso será Empleado público, debiendo tener los campos especificados a continuación, cuyo valor se deberá determinar a partir del campo indicado.

Tipo de Principal	Campo	Origen
Empleado público	Id	dir4DocumentID
	Tipo de id	NIF
	Nombre	givenName
	Primer apellido	sn
	Segundo apellido	dir4LastName
	Correo electrónico	dir4Email
	Número de identificación personal	N/D
	NIF organismo	N/D
	Denominación organismo	dir4OrganizationDesc
	DIR3 organismo	dir4OrganizationCode
	Unidad	dir4JobCentreDesc
	Puesto	title

Tabla 8-12 Estructura del Principal en la autenticación con Autentica

Norma AR-SEG-29

Los **sistemas o subsistemas frontera** que expongan **interfaces Web autenticados** con nivel de acceso **público, restringido o privado** y orientado a **empleados públicos**, deberán disponer de un mecanismo de control de autenticación basado en Autentica.

En el alta de la integración del subsistema aplicativo con Autentica deberá tenerse en cuenta la correspondencia de niveles de seguridad especificada en la **Tabla 8-9**

Los mensajes SAML de respuesta correcta deberán procesarse para extraer los datos especificados en la **Tabla 8-10**.

Finalmente, deberá construirse un objeto Authentication reflejando el resultado de ésta, el cual deberá incluir las autorizaciones especificadas en la **Tabla 8-11**, a la que deberá añadirse **ROLE_EMPLEADO_PUBLICO**, y un objeto Principal reflejando los datos especificados en la **Tabla 8-12**.

8.3.7.4 Control de autenticación basado en Active Directory

El control de autenticación basado en Active Directory podrá emplearse únicamente en el acceso a interfaces Web privados, en los escenarios en los que no se requiera un modelo de autorización complejo, como el proporcionado por Autentica.

La autenticación mediante Active Directory se basa en la pertenencia del usuario a uno o más grupos, los cuales tienen en este escenario sentido de autorizaciones. Para generar sus equivalentes en Spring Security, se deberá seguir el convenio de nombres que se describe a continuación.

Grupo de Active Directory	GG_[app]_U_[entorno]_[descripción]
Autorización en Spring Security	ROLE_[descripción]

Tabla 8-13 Correspondencia entre los grupos de Active Directory y las autorizaciones de Spring Security

El tipo de *principal*, en el caso de Active Directory, en todo caso será Empleado público, debiendo tener los campos especificados a continuación, cuyo valor se deberá determinar a partir del campo indicado.

Tipo de Principal	Campo	Origen
Empleado público	Id	EmployeeId
	Tipo de id	NIF
	Nombre	givenName
	Primer apellido	sn
	Segundo apellido	sn
	Correo electrónico	mail
	Número de identificación personal	N/A
	NIF organismo	N/A
	Denominación organismo	Company
	DIR3 organismo	N/A
	Unidad	Department
	Puesto	Title

Tabla 8-14 Estructura del Principal en la autenticación con Active Directory

Norma AR-SEG-30

Los sistemas o subsistemas frontera que expongan interfaces Web autenticados con nivel de acceso privado, podrán disponer de un mecanismo de control de autenticación basado en Active Directory.

Tras la validación de la contraseña, deberán procesarse los nombres de los diferentes grupos a los que esté vinculado el usuario, según se especifica en la Tabla 8-13, donde:

- [app]: código identificador del sistema de información en Active Directory.
- [entorno]: identificador del entorno al que se refiere el grupo, solamente cuando éste sea diferente del de producción:

- **INT**: entorno de integración.
- **QA**: entorno de calidad.
- **SS**: entorno de servicios estables.
- **PRE**: entorno de pre-producción.
- **[descripción]**: nombre largo o descripción del grupo, de hasta 30 caracteres.

Finalmente, deberá construirse un objeto Authentication, reflejando el resultado de ésta, el cual deberá incluir las autorizaciones obtenidas según el procedimiento anterior, a la que deberá añadirse **ROLE_EMPLEADO_PUBLICO**, y un objeto Principal reflejando los datos especificados en la **Tabla 8-14**.

8.3.7.5 Control de autenticación mediante TLS

El control de autenticación basado en TLS deberá emplearse únicamente en el acceso desde el exterior a **interfaces WS** públicos o restringidos, y se basará en la configuración del middleware correspondiente para reconocer el certificado digital del cliente, a partir de su clave pública.

Los servidores frontales del middleware están configurados para terminar el enlace TLS, con doble certificado de servidor y de cliente. La información contenida en el certificado de cliente acerca de éste es volcada, en formato de *distinguished name*, DN o nombre distinguido, a una cabecera HTTP, la cual es propagada al servidor de integración, quien, a su vez, volverá a propagarla al interfaz WS del subsistema aplicativo.

El tipo de *principal*, en el caso de TLS, en todo caso será Entidad, debiendo tener los campos especificados a continuación, cuyo valor se deberá determinar a partir del procesamiento de los siguientes nombres distinguídos relativos, contenidos en el nombre distinguido.

Tipo de Principal	Campo	Origen
Entidad	Id	OI (2.5.4.97) ¹⁵
	Tipo de id	NIF
	Razón social	O (2.5.4.10)
	Componente	CN (2.5.4.3)

Tabla 8-15 Estructura del Principal en la autenticación con TLS

En función del NIF obtenido como identificador, se determinan sus autorizaciones asociadas como se indica a continuación.

¹⁵ El formato de este campo viene determinado por ETSI EN 319 412-1. En el caso de personas jurídicas o entidades sin personalidad jurídica españolas, su valor seguirá el patrón “VATES-[NIF]” donde [NIF] se corresponde con el NIF de la entidad.

Autorizaciones

ROLE_ENTIDAD
ROLE_CIUDADANO¹⁶
ROLE_ADMINISTRACION¹⁷
ROLE_PERSONA_JURIDICA¹⁸
ROLE_ENTIDAD_SIN_PERSONALIDAD_JURIDICA¹⁹
ROLE_ORGANISMO²⁰
ROLE_ORGANO²¹
ROLE_CORPORACION_LOCAL²²

Tabla 8-16 Autorizaciones a incluir en la autenticación con TLS

Norma AR-SEG-31

Los **interfaces WS** expuestos a través del **middleware** externo o interno a los puertos **público** o **restringido**, dispondrán de un mecanismo de control de acceso basado en **TLS** y el uso de **certificado de sitio Web** (servidor) y de **sello** (cliente).

Los servidores frontales del middleware verificarán el certificado del cliente y, en caso de ser válido, propagarán la información del sujeto autenticado, en formato DN, sobre la cabecera HTTP **ssl_client_dn**. Esta cabecera será, a su vez, propagada de nuevo por el servidor de integración al subsistema aplicativo que provee el interfaz WS.

Tras el procesamiento de la cabecera, deberá conformarse un objeto Authentication reflejando el resultado de ésta, el cual deberá incluir las autorizaciones obtenidas de acuerdo a la **Tabla 8-16**, y un objeto Principal reflejando los datos especificados en la **Tabla 8-15**.

8.3.7.6 Control de autenticación mediante WSS

El control de autenticación mediante *Web Services Security*, seguridad de servicios Web o WSS deberá emplearse únicamente en el acceso desde el interior a **interfaces WS** internos provistos por sistemas de información que formen parte de la Plataforma de servicios compartidos.

En este escenario de autenticación se está verificando la autenticidad del sistema de información llamante, permitiendo disponer de un control de los sistemas de información efectivamente integrados con la Plataforma de servicios compartidos.

¹⁶ Si el primer carácter del NIF es diferente de P, Q o S.

¹⁷ Si el primer carácter del NIF es P, Q o S.

¹⁸ Si el primer carácter del NIF es A, B, C, D, F, G, J, Q, R, S, P o V.

¹⁹ Si el primer carácter del NIF es E, H, P, S, U o W.

²⁰ Si el primer carácter del NIF es Q.

²¹ Si el primer carácter del NIF es S.

²² Si el primer carácter del NIF es P.

Norma AR-SEG-32

Los **interfaces WS** internos, provistos por sistemas de información de la Plataforma de servicios compartidos, y expuestos a través del **middleware** interno, dispondrán de un mecanismo de control de acceso basado en **WSS** para autenticar al sistema de información llamante, basado en **username token**.

8.3.7.7 Propagación del contexto de seguridad

Los subsistemas aplicativos que, tras el correspondiente proceso de autenticación, dispongan de un contexto de seguridad, deberán propagar la información contenida en éste en cada una de las llamadas que realicen a través de servicios Web, mediante los correspondientes conectores, a otros subsistemas aplicativos que pertenezcan al mismo sistema de información.

Para ello se empleará un token [JWT](#), con la siguiente estructura.

Claim	Descripción
aud	Identificador único, a efectos de gestión, del sistema de información (véase apartado 9.1).
aum	Medio de autenticación: clave, autentica, ad, o tls .
tpr	Tipo de principal: <ul style="list-style-type: none"> • pf: Persona física • ep: Empleado público • pfr: Persona física representante • e: Entidad
sub	Identificador del principal
tid	Tipo de identificador del principal
nom	Nombre del principal ²³
pap	Primer apellido del principal ²³
sap	Segundo apellido del principal ²³
cel	Correo electrónico del principal ²³
nip	Número de identificación personal del principal ²⁴
nor	NIF organismo del principal ²⁴
dor	Denominación organismo del principal ²⁴
d3o	DIR3 organismo del principal ²⁴
uni	Unidad del principal ²⁴
pto	Puesto del principal ²⁴
nen	NIF entidad del principal ²⁵

²³ Solamente en caso de que el principal sea Persona física, Empleado público, Persona física representante de persona jurídica o Persona física representante de entidad sin personalidad jurídica.

²⁴ Solamente en caso de que el principal sea Empleado público.

²⁵ Solamente en caso de que el principal sea Persona física representante de persona jurídica, Persona física representante de entidad sin personalidad jurídica, o Entidad.

rse	Razón social entidad del principal ²⁵
com	Componente del principal ²⁶
grt	Lista de autorizaciones

Tabla 8-17 Estructura de la cabecera WS-S para la propagación de identidad

Norma AR-SEG-33

Los **subsistemas aplicativos frontera** que provean un **interfaz Web autenticado**, y que realicen llamadas a través de servicios Web, mediante los correspondientes **conectores**, a otros subsistemas aplicativos en el ámbito del mismo **sistema de información**, deberán incluir la información de autenticación que se obtuvo en el correspondiente proceso de autenticación.

Esta información deberá formatearse en forma de **token JWT** de acuerdo a la estructura definida en la **Tabla 8-17**, donde deberán incluirse los **claims** relativos al principal que correspondan, en función del tipo de principal, y de acuerdo con la **Tabla 8-8, Tabla 8-12, Tabla 8-14 y Tabla 8-15**.

El token así generado deberá firmarse mediante **JWS** con el algoritmo **HMAC SHA-256**, y enviarse en la cabecera **HTTP Authorization**, indicando el esquema de autenticación **Bearer**.

El subsistema aplicativo receptor de la llamada, siempre que pertenezca al **mismo sistema de información** que el subsistema aplicativo emisor, deberá volcar la información así recibida en el contexto de seguridad de Spring Security.

8.3.7.8 Gestión de sesiones

Las sesiones Web son áreas de memoria que el sistema utiliza para almacenar información entre las distintas peticiones que un mismo cliente realiza con el sistema, representando así el estado conversacional del cliente con éste. Poseen un identificador único, y un tiempo de vida limitado.

Norma AR-SEG-34

Los **subsistemas aplicativos frontera** que expongan **interfaces Web** deberán disponer de un mecanismo de **control de sesiones Web**.

Este mecanismo de control se debe basar en el uso de una **cookie de sesión**, que deberá ser **no persistente**, generada en el servidor de aplicaciones y almacenada en el cliente. Ésta deberá incorporarla en todas las peticiones que realice al subsistema aplicativo.

La creación de la sesión Web y, por tanto, de la cookie de sesión, se debe realizar inmediatamente **después de la autenticación**, en caso de que ésta sea requerida.

²⁶ Solamente en caso de que el principal sea Entidad.

El concepto de sesión únicamente aplica en el caso de interfaces Web. Los **interfaces WS** carecen de estado conversacional y, por tanto, **no deben disponer de sesiones**.

El uso de cookies de sesión **no requiere consentimiento** por parte del usuario.

Los subsistemas aplicativos que provean uno o más interfaces Web estarán configurados para formar un clúster. Esto permite que las sesiones de cada instancia sean accesibles desde la otra mediante serialización estándar de objetos en Java.

Norma AR-SEG-35

Todo objeto que se almacene en una sesión Web ha de ser **serializable**.

8.3.8 Fallos de integridad de software y datos

Los fallos de integridad de software y datos vienen motivados por la introducción en el sistema, ya sea en su diseño, construcción o implantación, de piezas de software o de datos externos cuya integridad no haya sido verificada.

8.3.8.1 Uso de librerías normalizadas

En tiempo de entrega del código fuente de un subsistema aplicativo debe revisarse que las librerías empleadas por éste sean normalizadas.

Norma AR-SEG-36

En **tiempo de entrega del código fuente** de un subsistema aplicativo, debe revisarse que los **repositorios** de librerías empleados sean los corporativos, así como que las **librerías** efectivamente empleadas formen parte de aquellas determinadas por la **Política de actualización de APIs, frameworks y librerías**.

8.3.8.2 Impedir la deserialización insegura

Los ataques por deserialización insegura ocurren cuando un sistema de información recibe objetos serializados dañinos. Estos objetos pueden ser manipulados por el atacante para realizar ataques de repetición, inyecciones, o elevar sus privilegios de ejecución. En el peor de los casos, podría provocar la ejecución remota de código en el servidor.

Norma AR-SEG-37

No se permite que los subsistemas aplicativos reciban datos desde el exterior de su ámbito en forma de **objetos Java serializados**.

8.3.9 Fallos de registro y monitorización de seguridad

Si el sistema de información no registra suficiente información de actividad, y ésta no es monitorizada adecuadamente, es posible que un eventual ataque se mantenga en el tiempo sin ser detectado.

8.3.9.1 Registros de actividad

Los mecanismos de auditoría se articulan en torno a una serie de registros de actividad, los cuales son generados por los servidores y por los distintos componentes lógicos de cada subsistema aplicativo, así como por los elementos de red y el middleware.

Cada registro de actividad contiene eventos con información relativa a determinado ámbito, distinguiéndose los siguientes registros de actividad.

Registro de actividad	Descripción
Registro de trazas	Contiene eventos con cualquier información acerca de la actividad del subsistema aplicativo que el equipo de desarrollo considere relevante para facilitar su mantenimiento
Registro de seguridad	Contiene eventos con información de seguridad que podría ser útil en la detección y mitigación de eventuales ataques
Registro de intercambios de datos	Contiene eventos con información de intercambios de datos entre diferentes sistemas o subsistemas

Tabla 8-18 Registros de actividad de los componentes lógicos

Norma AR-SEG-38

Los **servidores** de cualquier **subsistema aplicativo**, así como los del **middleware**, deben estar configurados para activar sus registros de actividad.

Norma AR-SEG-39

Los **componentes lógicos** de cualquier subsistema aplicativo deben generar eventos de auditoría para los registros de **trazas** y de **seguridad**.

El **middleware** debe generar eventos de auditoría para el registro de **intercambios de datos**.

Norma AR-SEG-40

Para poder correlacionar en sentido temporal los eventos contenidos en los diferentes registros de actividad, es necesario que se **sincronicen** los **relojes** del sistema de todos los **servidores**.

8.3.9.2 Eventos de traza

Los eventos de traza son generados por los diferentes componentes lógicos durante su ejecución, para informar de hechos, con determinada relevancia o severidad, que se consideran relevantes en el contexto del mantenimiento del subsistema aplicativo.

Los eventos de traza deberán tener la estructura que se indica a continuación.

Campo	Descripción
Timestamp	Fecha y hora en que se generó el evento, con precisión de milisegundos
Subsistema aplicativo	Identificador absoluto del subsistema aplicativo (véase apartado 9.1).
Servidor	Dirección IP o nombre del servidor de aplicaciones
Nodo	Nombre del nodo del servidor de aplicaciones
Identificador de sesión	Identificador único de la sesión Web, con formato UUID
Identificador de petición	Identificador único de la petición, con formato UUID
Hebra	Nombre de la hebra que ejecuta el componente que genera el evento
Clase	Nombre de la clase que genera el evento
Línea	Número de línea de código en la clase anterior
Severidad	Nivel de severidad del evento
Mensaje	Mensaje de texto que describe el hecho que se quiere informar
Excepción	En caso de que el motivo del evento sea la ocurrencia de una excepción, se debe incluir el <i>stacktrace</i> o volcado de pila de dicha excepción

Tabla 8-19 Evento de traza

Los niveles de severidad contemplados en los eventos de traza son los siguientes.

Nivel	Descripción
FATAL	Se utiliza para informar de que una o más funcionalidades fundamentales del subsistema aplicativo no pueden funcionar, provocando que el subsistema aplicativo en su conjunto no pueda dar servicio
ERROR	Se utiliza para informar de que una o más funcionalidades del subsistema aplicativo no funcionan, provocando que algunas de las peticiones se vean afectadas
WARN	Se utiliza para informar de un comportamiento inesperado que no impide que el subsistema aplicativo pueda funcionar normalmente
INFO	Se utiliza para informar de un hecho relevante, con carácter meramente informativo
DEBUG	Se utiliza para indicar información relevante durante la depuración del código de los componentes lógicos

TRACE

Se utiliza para indicar información detallada durante la depuración del código de los componentes lógicos

Tabla 8-20 Niveles de severidad en los eventos de traza

Norma AR-SEG-41

Los eventos de traza generados por los subsistemas aplicativos deberán disponer de la estructura definida en la **Tabla 8-19**, y almacenarse en el registro de trazas.

En función de la relevancia de la información que contengan, deberán emplearse los niveles de severidad definidos en la **Tabla 8-20**.

Como mínimo, deberán generarse eventos de traza en los ámbitos y situaciones que se indican a continuación.

Norma AR-SEG-42

En todo caso, deberán generarse eventos de traza en los ámbitos y situaciones siguientes:

- En **tiempo de arranque** del subsistema aplicativo, deberán generarse los siguientes eventos:
 - Cualquier error que impida el correcto y completo arranque de los *frameworks* base, de cualquiera de los componentes técnicos que proporcionan, o de cualquiera de los componentes aplicativos, con severidad FATAL y el *stacktrace* o volcado de pila de la excepción causante.
 - Correcto arranque del subsistema aplicativo, con severidad INFO.
- En **tiempo de parada** del subsistema aplicativo, deberán generarse los siguientes eventos:
 - Cualquier error que impida la correcta y completa parada de los *frameworks* base, de cualquiera de los componentes técnicos que proporcionan, o de cualquiera de los componentes aplicativos, con severidad ERROR y el *stacktrace* o volcado de pila de la excepción causante.
 - Correcta parada del subsistema aplicativo, con severidad INFO.
- En **tiempo de ejecución** del subsistema aplicativo deberán generarse los siguientes eventos:
 - Cualquier componente lógico deberá notificar cualquier hecho que se considere relevante en el contexto del mantenimiento del subsistema aplicativo.
 - Los componentes de la capa de presentación deben notificar cualquier excepción que se haya propagado hasta ellos, con severidad ERROR y el *stacktrace* o volcado de pila correspondiente.

8.3.9.3 Eventos de seguridad

Los eventos de seguridad son generados por los diferentes componentes lógicos durante su ejecución, para informar de hechos que se consideran relevantes en el ámbito de la seguridad, y que podrían ser útiles en la detección de eventuales ataques.

Se distinguen los tipos de evento de seguridad que se especifican a continuación.

Tipo de evento	Descripción
S1	Inicio de sesión con autenticación correcta
S2	Intento de inicio de sesión con autenticación fallida
S3	Fin de sesión
S4	Invocación de operativa de negocio
S5	Intento de invocación con autorización fallida

Tabla 8-21 Tipos de eventos aplicativos de seguridad

Cada uno de estos tipos de evento deberá tener la estructura que se especifica a continuación.

Campo	Descripción
Timestamp	Fecha y hora en que se generó el evento, con precisión de milisegundos
Subsistema aplicativo	Identificador absoluto del subsistema aplicativo (véase apartado 9.1).
Servidor	Dirección IP o nombre del servidor de aplicaciones
Nodo	Nombre del nodo del servidor de aplicaciones
Tipo de evento	S1
Mecanismo de autenticación	Mecanismo empleado para la realización de la autenticación: <ul style="list-style-type: none"> • clave • autentica • ad
Identificador de usuario	NIF o identificador eIDAS del usuario autenticado
Identificador de sesión	Identificador único de la sesión Web, con formato UUID

Tabla 8-22 Evento de inicio de sesión con autenticación correcta

Campo	Descripción
Timestamp	Fecha y hora en que se generó el evento, con precisión de milisegundos
Subsistema aplicativo	Identificador absoluto del subsistema aplicativo (véase apartado 9.1).

Servidor	Dirección IP o nombre del servidor de aplicaciones
Nodo	Nombre del nodo del servidor de aplicaciones
Tipo de evento	S2
Mecanismo de autenticación	Mecanismo empleado para la realización de la autenticación: <ul style="list-style-type: none">• clave• autentica• ad
Identificador de usuario	Dirección IP del usuario

Tabla 8-23 Evento de intento de inicio de sesión con autenticación fallida

Campo	Descripción
Timestamp	Fecha y hora en que se generó el evento, con precisión de milisegundos
Subsistema aplicativo	Identificador absoluto del subsistema aplicativo (véase apartado 9.1).
Servidor	Dirección IP o nombre del servidor de aplicaciones
Nodo	Nombre del nodo del servidor de aplicaciones
Tipo de evento	S3
Identificador de sesión	Identificador único de la sesión Web, con formato UUID

Tabla 8-24 Evento de fin de sesión

Campo	Descripción
Timestamp	Fecha y hora en que se generó el evento, con precisión de milisegundos
Subsistema aplicativo	Identificador absoluto del subsistema aplicativo (véase apartado 9.1).
Servidor	Dirección IP o nombre del servidor de aplicaciones
Nodo	Nombre del nodo del servidor de aplicaciones
Tipo de evento	S4
Identificador de sesión	Identificador único de la sesión Web, con formato UUID, en caso de que exista
Identificador de petición	Identificador único de la petición, con formato UUID, en caso de que exista
Mecanismo de autenticación	Mecanismo empleado para la realización de la autenticación: <ul style="list-style-type: none">• clave• autentica• ad• tls
Identificador de usuario	NIF o identificador eIDAS
Método HTTP	Método HTTP asociado a la petición

URL	URL asociada a la petición
SOAPAction	Valor de la cabecera SOAPAction
Acción	Acción asociada a la operativa de negocio
Información	Indicación de la información sobre la que se ejecuta la acción, obtenida a partir de los parámetros de entrada o de salida de la misma
Resultado	Resultado de la operativa: <ul style="list-style-type: none">• ERROR: si termina con excepción• OK: en caso contrario
Duración	Duración de la invocación, en milisegundos

Tabla 8-25 Evento de invocación de operativa de negocio

Campo	Descripción
Timestamp	Fecha y hora en que se generó el evento, con precisión de milisegundos
Subsistema aplicativo	Identificador absoluto del subsistema aplicativo (véase apartado 9.1).
Servidor	Dirección IP o nombre del servidor de aplicaciones
Nodo	Nombre del nodo del servidor de aplicaciones
Tipo	S5
Identificador de sesión	Identificador único de la sesión Web, con formato UUID, en caso de que exista
Identificador de petición	Identificador único de la petición, con formato UUID
Mecanismo de autenticación	Mecanismo empleado para la realización de la autenticación: <ul style="list-style-type: none">• clave• autentica• ad• tls
Identificador de usuario	NIF o identificador eIDAS
Método HTTP	Método HTTP asociado a la petición
URL	URL asociada a la petición
SOAPAction	Valor de la cabecera SOAPAction

Tabla 8-26 Evento de intento de invocación con autorización fallida

Norma AR-SEG-43

Se contemplan los tipos de evento de seguridad definidos en la **Tabla 8-21**.

Los eventos de seguridad generados por los subsistemas aplicativos deberán disponer de la estructura definida en la **Tabla 8-22**, **Tabla 8-23**, **Tabla 8-24**, **Tabla 8-25** y **Tabla 8-26**, y almacenarse en el registro de seguridad.

Los eventos de seguridad de tipo **S1**, **S2** y **S5** deberán generarse por los distintos componentes que conformen la configuración de seguridad basada en **Spring Security**.

Para que los servidores de aplicaciones tengan visibilidad de la IP de cliente, los **servidores frontales** deben enviar este dato mediante la cabecera **X-Forwarded-For**.

Los eventos de seguridad de tipo **S4** deberán generarse ante la invocación de cualquiera de las operativas en las **fachadas de negocio** del subsistema aplicativo.

Las operativas de la fachada de negocio de **tareas de mantenimiento** también son susceptibles de generar eventos de seguridad de tipo **S4**, si bien en este caso no va a existir una autenticación previa, ni una petición o sesión Web asociadas.

8.3.9.4 Eventos de intercambio de datos

Los eventos de intercambio de datos son generados por el middleware durante su ejecución, para informar de hechos que se consideran relevantes en el contexto del mantenimiento de los diferentes sistemas de información.

Se distinguen los tipos de evento de intercambio de datos que se especifican a continuación.

Tipo de evento	Descripción
I1	Intercambio de datos general mediante servicios Web
I2	Intercambio de datos mediante la Plataforma de intermediación de datos, a través de petición síncrona

Tabla 8-27 Tipos de eventos de intercambio de datos

Los eventos de intercambio de datos general mediante servicios Web están vinculados lógicamente a un par petición-respuesta, debiendo tener la estructura que se indica a continuación.

Campo	Descripción
Timestamp	Fecha y hora en que el proxy recibió la petición, con precisión de milisegundos
Identificador del subsistema aplicativo	Identificador del subsistema aplicativo llamante, determinado a partir del propio mensaje de petición y en función de cada servicio
Identificador de sesión	Identificador único de la sesión Web en el subsistema aplicativo llamante, con formato UUID, en caso de que exista
Identificador de petición	Identificador único de la petición en el subsistema aplicativo llamante, con formato UUID, en caso de que exista
Método HTTP	Método HTTP

URL	URL del servicio proxy invocado
SOAPAction	Cabecera SOAPAction
Resultado	Código HTTP de respuesta
Duración	Duración de la invocación, en milisegundos
Tipo de evento	I1

Tabla 8-28 Evento de intercambio de datos general mediante servicios Web

Los eventos de intercambio de datos mediante la Plataforma de intermediación de datos están vinculados lógicamente a un par solicitud-transmisión de datos, pudiendo existir varios de ellos en el contexto de un mismo par petición-respuesta en el caso de peticiones asíncronas, o bien un único par solicitud-transmisión de datos, en el caso de peticiones síncronas.

Campo	Descripción
Timestamp	Fecha y hora en que el proxy recibió la petición, con precisión de milisegundos
Identificador del subsistema aplicativo	Identificador del subsistema aplicativo llamante, determinado a partir del propio mensaje de petición y en función de cada servicio
Identificador de sesión	Identificador único de la sesión Web en el subsistema aplicativo llamante, con formato UUID, en caso de que exista
Identificador de petición	Identificador único de la petición en el subsistema aplicativo llamante, con formato UUID, en caso de que exista
Método HTTP	Método HTTP
URL	URL del servicio proxy invocado
SOAPAction	Cabecera SOAPAction
Resultado	Código HTTP de respuesta
Duración	Duración de la invocación, en milisegundos
Tipo de evento	I2
Identificador de petición PID	Identificador de la petición en la Plataforma de intermediación de datos
Código de estado	Código de estado de la respuesta
Código de estado secundario	Código de estado secundario de la respuesta
Descripción del error	Descripción del error
Código del certificado	Identificador del certificado o transmisión solicitado
NIF emisor	NIF del organismo emisor
Nombre emisor	Nombre del organismo emisor
NIF del solicitante	NIF del organismo solicitante
Nombre del solicitante	Nombre del organismo solicitante
Identificador de la unidad trasmitadora	Código DIR3 de la unidad trasmitadora
Nombre de la unidad trasmitadora	Nombre de la unidad trasmitadora

Código del procedimiento	Código SIA del procedimiento en cuya tramitación se realiza el intercambio de datos
Nombre del procedimiento	Nombre del procedimiento en cuya tramitación se realiza el intercambio de datos
Clase de procedimiento	Código de la familia funcional a la que pertenece el procedimiento
Procedimiento automatizado	Indicador de si el procedimiento se realiza de forma automática por un sistema de información, o bien si se realiza de forma manual
Finalidad	Finalidad del tratamiento de datos personales
Legitimación	Legitimación del tratamiento de datos personales
NIF del funcionario	NIF del funcionario que solicita la información, o bien del funcionario responsable técnico del sistema de información
Nombre del funcionario	Nombre completo del funcionario trámitador
Seudónimo del funcionario	Seudónimo del funcionario trámitador
Identificador del expediente	Identificador del expediente concreto en cuya tramitación se solicita el certificado o la transmisión
Tipo de identificador del titular	Tipo de identificador del titular
Identificador del titular	Identificador del titular
Nombre completo del titular	Nombre completo del titular
Nombre del titular	Nombre del titular
Apellido1 del titular	Primer apellido del titular
Apellido2 del titular	Segundo apellido del titular
Identificador de solicitud PID	Identificador de la solicitud en el ámbito de la petición en la Plataforma de intermediación de datos
Identificador de transmisión PID	Identificador de la transmisión en la Plataforma de intermediación de datos, correlacionada con la solicitud correspondiente

Tabla 8-29 Evento de intercambio de datos a través de la Plataforma de intermediación de datos (síncrono)

Importante
Las peticiones asíncronas a la Plataforma de intermediación de datos no tienen sentido cuando la tramitación del procedimiento administrativo se sustenta en un sistema de información. Su uso debería restringirse al ámbito de las tramitaciones manuales.

Norma AR-SEG-44

Se contemplan los tipos de evento de intercambio de datos definidos en la **Tabla 8-27**.

Los eventos de intercambio de datos generados por el middleware deberán disponer de la estructura definida en la **Tabla 8-28** y **Tabla 8-29**, y almacenarse en el registro de intercambios de datos, el cual estará constituido por un sistema de información específico en el ámbito de la Plataforma de servicios compartidos.

8.3.9.5 Identificadores de sesión y de petición

Los eventos de traza y de seguridad poseen identificadores de sesión y de petición, que deberán gestionarse como se indica a continuación.

En tiempo de creación de una nueva sesión Web, deberá generarse un identificador único para ella.

Norma AR-SEG-45

Los **interfaces Web** de los **subsistemas aplicativos frontera** deberán generar un **identificador** único para cada **sesión Web**, a efectos de trazabilidad, con formato **UUID**, tras la correcta autenticación del usuario.

En tiempo de pre-procesamiento de una petición Web, deberá generarse un identificador único para ella.

Norma AR-SEG-46

Los **interfaces Web** de los **subsistemas aplicativos frontera** deberán generar un **identificador** único para cada **petición Web**, a efectos de trazabilidad, con formato **UUID**, de manera previa al procesamiento de ésta.

Puesto que los subsistemas aplicativos se integran con otros sistemas o subsistemas, es necesario disponer de algún mecanismo que permita correlacionar los eventos generados por cada uno de ellos. De esta manera, se facilitan las labores de auditoría y depuración de problemas.

Norma AR-SEG-47

Los **eventos** generados por los diferentes sistemas y subsistemas involucrados en el tratamiento de una petición deberán **correlacionarse** a nivel lógico mediante los **identificadores de sesión y petición**.

En el **subsistema aplicativo origen**, los componentes con estereotipo **Conektor** deberán encargarse de propagar el identificador de sesión y el identificador de petición en las llamadas a servicios Web, empleando para ello las cabeceras HTTP siguientes:

- **X-Session-ID:** identificador de sesión.

- **X-Request-ID:** identificador de petición.

El **middleware** de integración deberá estar configurado de forma que propague dichas cabeceras.

Finalmente, los componentes técnicos que gestionen el tratamiento de peticiones en el **subsistema aplicativo destino**, deberán determinar ambos identificadores a partir de las cabeceras recibidas. En caso de que la cabecera con el identificador de petición no estuviese presente, deberá determinarse un valor para identificar la petición.

8.3.9.6 Recolección y análisis de eventos

Los registros de actividad aplicativos se deberán ubicar en rutas normalizadas del sistema de ficheros de los servidores de aplicaciones, como se indica a continuación.

Norma AR-SEG-48

Los registros de actividad aplicativos se deberán ubicar en los ficheros **trace.log** (registro de trazas) y **security.log** (registro de seguridad), en la siguiente ruta normalizada del sistema de ficheros de los servidores de aplicaciones:

- **/opt/logs/app/[applicationId]/[nodeName]/**

Donde:

- **[applicationId]:** identificador único, a efectos de gestión, del subsistema aplicativo (véase apartado 9.1).
- **[nodeName]:** nombre del nodo del servidor de aplicaciones.

Para facilitar el análisis y tratamiento de los eventos, es conveniente disponer de sistemas colectores que centralicen su almacenamiento.

En el caso de los eventos de traza, suelen emplearse colectores específicos, que se alimentan de los ficheros del registro de trazas de los distintos subsistemas aplicativos, y que disponen de un interfaz de usuario que permite realizar búsquedas y filtrar resultados.

Los eventos de seguridad suelen recolectarse y procesarse en *Security Information Event Management Systems*, sistemas de gestión de eventos e información de seguridad o SIEM, para detectar patrones de ataque. No obstante, también pueden cargarse por los colectores de trazas.

Norma AR-SEG-49

Se establece el uso de [Graylog](#) como sistema **colector de eventos de traza**.

La versión vigente en cada momento del software base anterior queda determinada por la **Política de actualización de la línea base de plataformas**.

Este sistema colector se alimentará de los archivos **trace.log** y **security.log** de los distintos subsistemas aplicativos.

Asociado a cada **sistema de información**, deberá existir un flujo específico para los eventos de traza de todos sus subsistemas aplicativos, y un flujo específico adicional para los eventos de seguridad de todos sus subsistemas aplicativos.

El acceso a dichos flujos deberá ser autenticado.

El sistema colector de trazas deberá configurarse adecuadamente para asegurar, como mínimo, un periodo de **retención** de **6 meses** para los **eventos de traza**, y de **18 meses** para los **eventos de seguridad**.

8.3.9.7 Auditoría de cambios de estado de entidades

En ocasiones, resulta conveniente incluir, en las entidades del modelo de dominio interno, mecanismos que permitan auditar las acciones de creación o, especialmente, de cambio de estado de las mismas.

Norma AR-SEG-50

En las situaciones en las que las entidades del modelo de dominio interno lo requieran, y especialmente cuando estas entidades sustentan el control del estado de las actividades llevadas a cabo por un subsistema aplicativo, deberán aplicarse sobre ellas mecanismos de **auditoría de entidades**.

Estos mecanismos se basarán en la definición, en el ámbito de una entidad, de:

- Los **campos** que representan el **estado** de la actividad.
- Los **campos de auditoría**, como el identificador del usuario que provoca el cambio y el instante de tiempo en que éste se produce.
- Una **tabla** específica de auditoría para albergar los **cambios** de estado de la entidad a lo largo del tiempo, los cuales se modelarán mediante la unión de los campos que representan el nuevo estado y los campos de auditoría.
- Un **trigger** encargado de añadir automáticamente sobre la tabla anterior los datos correspondientes a cada cambio de estado.

La gestión de los campos de auditoría se realizará mediante los mecanismos que proporciona para ello Spring Data.

8.3.10 Falsificación de solicitudes en el servidor

Los ataques de *Server-Side Request Forgery*, falsificación de solicitudes en el servidor o SSRF, ocurren cuando un subsistema aplicativo es instado a descargar datos externos a través de una URL proporcionada por el usuario.

8.3.10.1 Validación de URL

Antes de acceder a un recurso cuya URL haya sido proporcionada por un usuario, debe validarse que ésta es efectivamente externa al entorno de producción.

Norma AR-SEG-51

Si un dato de entrada tiene formato de **URL**, como parte de su **validación semántica** debe comprobarse que la dirección **IP** correspondiente es, efectivamente, **externa al entorno de producción**.

En tal caso, su acceso debe realizarse a través de un **proxy inverso**.

9 Arquitectura del código fuente

Las clases que *manifiestan* los componentes definidos en la arquitectura lógica de un determinado subsistema aplicativo, se organizan en proyectos aplicativos, los cuales facilitan su compilación y ensamblado en forma de *artefactos*.

La arquitectura del código fuente establece convenios para normalizar la estructura interna de estos proyectos aplicativos, así como la nomenclatura de sus elementos más relevantes.

Esta normalización tiene por objeto agilizar el mantenimiento de distintos proyectos aplicativos por un mismo equipo de trabajo, puesto que, al estar organizados sus elementos de la misma forma, se reduce la curva de aprendizaje.

9.1 Identificación de sistemas y subsistemas a efectos de gestión

A efectos de gestión, cada sistema o subsistema deberá disponer de un identificador único, como se indica a continuación.

9.1.1 Identificación de sistemas de información

Los sistemas de información deben disponer de un identificador único a efectos de gestión.

Norma AR-SRC-01

Cada **sistema de información** debe disponer de un **identificador único**.

Este identificador debe ser **alfabético**, expresarse en **minúsculas**, y disponer de **entre tres y doce caracteres**.

Este identificador debe determinarse en tiempo de diseño arquitectónico.

9.1.2 Identificación de subsistemas aplicativos

Los subsistemas aplicativos deben disponer de un identificador relativo al sistema de información al que pertenecen.

Norma AR-SRC-02

Cada **subsistema aplicativo** debe disponer de un **identificador relativo** en el ámbito del sistema de información al que pertenece.

Este identificador debe ser **alfabético**, expresarse en **minúsculas**, y disponer de **entre tres y doce caracteres**.

Este identificador debe determinarse en tiempo de diseño arquitectónico.

Por otro lado, los subsistemas aplicativos también deben disponer de un identificador único a efectos de gestión.

Norma AR-SRC-03

A efectos de gestión, los **subsistemas aplicativos** quedarán identificados de forma unívoca, mediante un **identificador único** con la siguiente estructura:

- **[systemId]-[subsystemId]**

Donde:

- **[systemId]**: identificador del sistema de información.
- **[subsystemId]**: identificador del subsistema aplicativo, en el ámbito del sistema de información al que pertenece.

Importante

En los apartados siguientes, el identificador único anterior será referido como **[applicationId]**.

9.2 Estilos de nomenclatura

Esta Arquitectura de Referencia contempla los estilos de nomenclatura que se especifican a continuación.

9.2.1 Upper Camel Case

El estilo de nomenclatura Upper Camel Case se caracteriza por la determinación del nombre de un elemento a partir de la concatenación de palabras individuales, sin ningún tipo de carácter separador.

Cada una de las palabras que se concatenan se escribe con la letra inicial en mayúscula, y el resto, en minúscula.

Por ejemplo, si quisiéramos escribir “estilo de nomenclatura” con estilo Upper Camel Case, obtendríamos “EstiloDeNomenclatura”.

Norma AR-SRC-04

Se establece la utilización del estilo de nomenclatura **Upper Camel Case** en los siguientes elementos:

- Nombres de tipos de datos complejos en Java
- Nombres de tipos de datos complejos en XML Schema

9.2.2 Lower Camel Case

El estilo de nomenclatura Lower Camel Case se caracteriza por la determinación del nombre de un elemento a partir de la concatenación de palabras individuales, sin ningún tipo de carácter separador.

La primera palabra se escribe con todas sus letras en minúscula, mientras que cada una de las restantes se escribe con la letra inicial en mayúscula, y el resto, en minúscula.

Por ejemplo, si quisiéramos escribir “estilo de nomenclatura” con estilo Lower Camel Case, obtendríamos “estiloDeNomenclatura”.

Norma AR-SRC-05

Se establece la utilización del estilo de nomenclatura **Lower Camel Case** en los siguientes elementos:

- Nombres de proyectos aplicativos
- Nombres de variables en Java
- Nombres de tipos de datos simples en XML Schema
- Nombres de elementos en XML Schema
- Nombres de atributos en XML Schema
- Nombres de ficheros XML
- Nombres de ficheros Properties
- Nombres de vistas
- Nombres de ficheros JSP
- Cualquier otro elemento para el que no se haya especificado un estilo de nomenclatura específico.

9.2.3 Snake Case

El estilo de nomenclatura Snake Case se caracteriza por la determinación del nombre de un elemento a partir de la concatenación de palabras individuales, empleando un guion bajo como carácter separador.

Por ejemplo, si quisiéramos escribir “estilo de nomenclatura” con estilo Snake Case, obtendríamos “estilo_de_nomenclatura”.

Norma AR-SRC-06

Se establece la utilización del estilo de nomenclatura **Snake Case** en los siguientes elementos:

- Nombres de paquetes en Java, debiéndose usar siempre en minúsculas.
- Nombres de constantes en Java, debiéndose usar siempre en mayúsculas.

- Nombres de objetos de la base de datos, debiéndose usar siempre en mayúsculas.

9.2.4 Kebab Case

El estilo de nomenclatura Kebab Case se caracteriza por la determinación del nombre de un elemento a partir de la concatenación de palabras individuales, empleando un guion medio como carácter separador.

Por ejemplo, si quisiéramos escribir “estilo de nomenclatura” con estilo Kebab Case, obtendríamos “estilo-de-nomenclatura”.

Norma AR-SRC-07

Se establece la utilización del estilo de nomenclatura **Kebab Case** en los siguientes elementos:

- URI, debiéndose usar siempre en minúsculas.
- Artefactos, debiéndose usar siempre en minúsculas.

9.2.5 Dot Case

El estilo de nomenclatura Dot Case se caracteriza por la determinación del nombre de un elemento a partir de la concatenación de palabras individuales, empleando un punto como carácter separador.

Por ejemplo, si quisiéramos escribir “estilo de nomenclatura” con estilo Dot Case, obtendríamos “estilo.de.nomenclatura”.

Norma AR-SRC-08

Se establece la utilización del estilo de nomenclatura **Dot Case** en los siguientes elementos:

- Nombres de propiedades de configuración definidas en los ficheros y recursos de propiedades de configuración, debiéndose usar siempre en minúsculas.
- Identificadores de grupos de artefactos, debiéndose usar siempre en minúsculas.

9.3 Licenciamiento

El código fuente de los distintos proyectos aplicativos relacionados con un sistema de información deberá estar sujeto a las condiciones de licenciamiento que se indican a continuación.

Norma AR-SRC-09

El código fuente de los distintos proyectos aplicativos relacionados con un sistema de información deberá licenciarse de acuerdo a la licencia [EUPL](#).

Todos los **proyectos** aplicativos deberán contener un fichero con el [contenido íntegro](#) de la licencia.

Adicionalmente, cada uno de los **ficheros fuente java** deberá disponer del siguiente **encabezado**:

```
/*
 * Copyright [año] Ministerio de Justicia
 *
 * Licencia con arreglo a la EUPL, Versión [versión] o -en cuanto
 * sean aprobadas por la Comisión Europea- versiones
 * posteriores de la EUPL (la «Licencia»);
 * Solo podrá usarse esta obra si se respeta la Licencia.
 * Puede obtenerse una copia de la Licencia en:
 *
 * https://joinup.ec.europa.eu/software/page/eupl
 *
 * Salvo cuando lo exija la legislación aplicable o se acuerde
 * por escrito, el programa distribuido con arreglo a la
 * Licencia se distribuye «TAL CUAL»,
 * SIN GARANTÍAS NI CONDICIONES DE NINGÚN TIPO, ni expresas
 * ni implícitas.
 * Véase la Licencia en el idioma concreto que rige
 * los permisos y limitaciones que establece la Licencia.
 */
```

9.4 Arquetipos de proyecto aplicativo

Los proyectos aplicativos se gestionarán con la herramienta Maven, distinguiéndose los arquetipos de proyecto que se determinan a continuación.

Norma AR-SRC-10

Los proyectos aplicativos se gestionarán mediante la herramienta [Maven](#), contemplándose los arquetipos de proyecto definidos en la **Tabla 9-1, Tabla 9-2, Tabla 9-3, Tabla 9-4, Tabla 9-5, Tabla 9-6 y Tabla 9-7**.

Todos los proyectos aplicativos que formen parte del mismo sistema de información, se organizarán en un único proyecto multi-módulo.

Los distintos ficheros fuente deberán estar codificados con **UTF-8**.

9.4.1 Arquetipo de proyecto de Sistema de información

Un proyecto Maven de arquetipo Sistema de información constituye el proyecto multi-módulo que albergará los proyectos de los diferentes subsistemas aplicativos que conformen el sistema de información, así como los de las librerías que puedan determinarse.

La estructura interna de este tipo de proyectos deberá ser la siguiente.

Estructura de proyectos de Sistema de información	
pom.xml	Configuración de gestión del proyecto.
README.txt	Información acerca del proyecto.
[módulo]	Una carpeta para cada uno de los módulos contemplados, ya sean subsistemas aplicativos o librerías. Su nombre deberá coincidir con el identificador del artefacto correspondiente.

Tabla 9-1 Estructura de proyectos de Sistema de información

9.4.2 Arquetipo de proyecto de Aplicación Web

Un proyecto Maven de arquetipo Aplicación Web deberá considerarse un módulo en el ámbito del proyecto Maven multi-módulo del sistema de información al que pertenece, el cual además deberá considerarse su proyecto padre.

La estructura interna de este tipo de proyectos deberá ser la siguiente.

Estructura de proyectos de Aplicación Web	
pom.xml	Configuración de gestión del proyecto.
README.txt	Información acerca del proyecto.
NOTICE.txt	Información sobre librerías de terceros utilizadas.
LICENSE.txt	Información de licenciamiento.
src/assembly/	Configuración de ensamblado.
src/test/java/ src/test/resources/	Código y recursos de los test unitarios.
src/main/java/ src/main/resources/	Código y recursos del artefacto principal.

src/main/resources/i18n	
src/main/sql/	Scripts de base de datos.
src/main/webapp/	Recursos Web.
src/main/webapp/WEB-INF/	Descriptores de despliegue.
src/main/webapp/WEB-INF/templates/	Plantillas de vistas.
src/main/webapp/static	Recursos estáticos Web.

Tabla 9-2 Estructura de proyectos de Aplicación Web

9.4.3 Arquetipo de proyecto de Fachada Web

Un proyecto Maven de arquetipo Fachada Web deberá considerarse un módulo en el ámbito del proyecto Maven multi-módulo del sistema de información al que pertenece, el cual además deberá considerarse su proyecto padre.

La estructura interna de este tipo de proyectos deberá ser la siguiente.

Estructura de proyectos de Fachada Web	
pom.xml	Configuración de gestión del proyecto.
README.txt	Información acerca del proyecto.
NOTICE.txt	Información sobre librerías de terceros utilizadas.
LICENSE.txt	Información de licenciamiento.
src/assembly/	Configuración de ensamblado.
src/test/java/ src/test/resources/	Código y recursos de los test unitarios.
src/main/java/ src/main/resources/ src/main/resources/i18n	Código y recursos del artefacto principal.
src/main/webapp/	Recursos Web.
src/main/webapp/WEB-INF/	Descriptores de despliegue.
src/main/webapp/WEB-INF/templates/	Plantillas de vistas.
src/main/webapp/static	Recursos estáticos Web.

Tabla 9-3 Estructura de proyectos de Fachada Web

9.4.4 Arquetipo de proyecto de Aplicación de servicios Web

Un proyecto Maven de arquetipo Aplicación de servicios Web deberá considerarse un módulo en el ámbito del proyecto Maven multi-módulo del sistema de información al que pertenece, el cual además deberá considerarse su proyecto padre.

La estructura interna de este tipo de proyectos deberá ser la siguiente.

Estructura de proyectos de Aplicación de servicios Web	
pom.xml	Configuración de gestión del proyecto.
README.txt	Información acerca del proyecto.
NOTICE.txt	Información sobre librerías de terceros utilizadas.
LICENSE.txt	Información de licenciamiento.
src/assembly/	Configuración de ensamblado.
src/test/java/ src/test/resources/	Código y recursos de los test unitarios.
src/main/java/ src/main/resources/ src/main/resources/i18n	Código y recursos del artefacto principal.
src/main/sql/	Scripts de base de datos.
src/main/webapp/	Recursos Web.
src/main/webapp/WEB-INF/	Descriptores de despliegue.

Tabla 9-4 Estructura de proyectos de Aplicación de servicios Web

9.4.5 Arquetipo de proyecto de Aplicación híbrida

Un proyecto Maven de arquetipo Aplicación híbrida deberá considerarse un módulo en el ámbito del proyecto Maven multi-módulo del sistema de información al que pertenece, el cual además deberá considerarse su proyecto padre.

La estructura interna de este tipo de proyectos deberá ser la siguiente.

Estructura de proyectos de Aplicación híbrida	
pom.xml	Configuración de gestión del proyecto.
README.txt	Información acerca del proyecto.
NOTICE.txt	Información sobre librerías de terceros utilizadas.
LICENSE.txt	Información de licenciamiento.
src/assembly/	Configuración de ensamblado.

src/test/java/ src/test/resources/	Código y recursos de los test unitarios.
src/main/java/ src/main/resources/	Código y recursos del artefacto principal.
src/main/sql/	Scripts de base de datos.
src/main/webapp/	Recursos Web.
src/main/webapp/WEB-INF/	Descriptores de despliegue.
src/main/webapp/WEB-INF/templates/	Plantillas de vistas.
src/main/webapp/static	Recursos estáticos Web.

Tabla 9-5 Estructura de proyectos de Aplicación híbrida

9.4.6 Arquetipo de proyecto de Librería de componentes compartidos

Un proyecto Maven de arquetipo Librería de componentes compartidos deberá considerarse un módulo en el ámbito del proyecto Maven multi-módulo del sistema de información al que pertenece, el cual además deberá considerarse su proyecto padre.

La estructura interna de este tipo de proyectos deberá ser la siguiente.

Estructura de proyectos de Librería de componentes compartidos	
pom.xml	Configuración de gestión del proyecto.
README.txt	Información acerca del proyecto.
NOTICE.txt	Información sobre librerías de terceros utilizadas.
LICENSE.txt	Información de licenciamiento.
src/assembly/	Configuración de ensamblado.
src/test/java/ src/test/resources/	Código y recursos de los test unitarios.
src/main/java/ src/main/resources/ src/main/resources/i18n	Código y recursos del artefacto principal.

Tabla 9-6 Estructura de proyectos de Librería de componentes compartidos

9.4.7 Arquetipo de proyecto de Librería de especificación de servicios Web

Un proyecto Maven de arquetipo Librería de especificación de servicios Web deberá considerarse un módulo en el ámbito del proyecto Maven multi-módulo del sistema de información al que pertenece el correspondiente interfaz WS provisto, el cual además deberá considerarse su proyecto padre.

La estructura interna de este tipo de proyectos deberá ser la siguiente.

Estructura de proyectos de Librería de especificación de servicios Web	
pom.xml	Configuración de gestión del proyecto.
README.txt	Información acerca del proyecto.
NOTICE.txt	Información sobre librerías de terceros utilizadas.
LICENSE.txt	Información de licenciamiento.
src/assembly/	Configuración de ensamblado.
src/main/wsdl/	Recursos WSDL y XSD.

Tabla 9-7 Estructura de proyectos de Librería de especificación de servicios Web

9.5 Codificación

Deberá emplearse la codificación que se indica a continuación.

Norma AR-SRC-11

Los distintos ficheros fuente deberán estar codificados con **UTF-8**.

9.6 Artefactos

Los artefactos, generados como resultado del proceso de ensamblado de los proyectos aplicativos, se normalizarán en base al arquetipo del proyecto aplicativo, como se indica a continuación.

9.6.1 Artefactos POM

Los artefactos POM se generarán de la manera en que se indica a continuación.

Coordenada	Descripción
groupId	es.mjusticia.[systemId]
artifactId	[systemId]
packaging	pom
classifier	N/A

Tabla 9-8 Coordenadas Maven de artefactos POM

Norma AR-SRC-12

El proceso de **ensamblado** del arquetipo de proyecto aplicativo **Sistema de información** debe generar como resultado un artefacto con las coordenadas Maven definidas en la **Tabla 9-8**.

9.6.2 Artefactos WAR

Los artefactos WAR se generarán de la manera en que se indica a continuación.

Path	Descripción
/META-INF/	Archivos README.txt , NOTICE.txt y LICENSE.txt .
/WEB-INF/templates/	Plantillas de vistas (excepto aplicación de servicios).
/WEB-INF/classes/	Archivos .class resultado de compilar las clases aplicativas ubicadas bajo src/main/java/ . Recursos ubicados bajo src/main/resources/ .
/WEB-INF/lib/	Librerías de las que dependan las clases aplicativas.

Tabla 9-9 Estructura interna de artefactos WAR

Coordenada	Descripción
groupId	es.mjjusticia.[systemId]
artifactId	[systemId]-[subsystemId]
packaging	war
classifier	N/A

Tabla 9-10 Coordenadas Maven de artefactos WAR

Norma AR-SRC-13

El proceso de **ensamblado** de los siguientes arquetipos de proyecto aplicativo debe generar como resultado un artefacto **WAR** con una estructura interna de acuerdo a la **Tabla 9-9**:

- **Aplicación Web.**
- **Fachada Web.**
- **Aplicación de servicios.**
- **Aplicación híbrida.**

El artefacto **WAR** debe quedar identificado por las coordenadas Maven definidas en la **Tabla 9-10**.

9.6.3 Artefactos JAR

Los artefactos JAR no son directamente desplegables, aunque sí podrán formar parte de artefactos WAR, que sí lo son.

Los artefactos JAR se generarán de la manera en que se indica a continuación.

Path	Descripción
/	Archivos .class resultado de compilar las clases aplicativas ubicadas bajo src/main/java/ .

	Recursos ubicados bajo src/main/resources/ .
/META-INF/	Archivos README.txt , NOTICE.txt y LICENSE.txt .

Tabla 9-11 Estructura interna de artefactos JAR de librerías de componentes compartidos

Path	Descripción
/	Archivos .class resultado de compilar las clases generadas y ubicadas bajo target/generated-sources/java/ .
/META-INF/	Archivos README.txt , NOTICE.txt y LICENSE.txt .
/META-INF/wsdl/	Recursos ubicados bajo src/main/wsdl/ .

Tabla 9-12 Estructura interna de artefactos JAR de librerías de especificación de servicios Web

Coordinada	Descripción
groupId	es.mjusticia.[systemId]
artifactId	Según se haya determinado en el diseño arquitectónico, con estilo Kebab Case en caso de ser varias palabras, y empleando únicamente minúsculas
packaging	jar
classifier	En caso de librerías de especificación de servicios Web, ws-spec

Tabla 9-13 Coordenadas Maven de artefactos JAR

Norma AR-SRC-14
El proceso de ensamblado de los siguientes arquetipos de proyecto aplicativo debe generar como resultado un artefacto JAR :
<ul style="list-style-type: none"> • Librería de componentes compartidos. • Librería de especificación de servicios Web.
El proceso de ensamblado de los proyectos de librería de especificación de servicios Web parte del archivo WSDL y de los XML Schema referenciados por él, y ubicados en el proyecto aplicativo bajo src/main/wsdl , como único código fuente.
Las clases java correspondientes se han de generar automáticamente mediante herramientas en el proceso de ensamblado, almacenando su código bajo la siguiente ruta del proyecto aplicativo:
<ul style="list-style-type: none"> • target/generated-sources/java/
Las clases java resultantes deben estar decoradas con la anotación @Generated para denotar que son auto-generadas.
Los artefactos JAR correspondientes a librerías de componentes compartidos deben tener la estructura interna indicada en la Tabla 9-11 .

Los artefactos **JAR** correspondientes a **librerías de especificación de servicios Web** deben tener la siguiente **estructura interna** especificada en la **Tabla 9-12**.

En los dos casos, el artefacto **JAR** debe quedar identificado por las coordenadas Maven definidas en la **Tabla 9-13**.

9.6.4 Artefactos de scripts de base de datos

Los artefactos de scripts de base de datos son archivos ZIP agrupando los diferentes scripts que se hayan definido.

Path	Descripción
/	Recursos ubicados bajo src/main/sql/ .

Tabla 9-14 Estructura interna de artefacto de scripts de base de datos

Coordenada	Descripción
groupId	es.mjjusticia.[systemId]
artifactId	[systemId]-[subsystemId]
packaging	zip
classifier	sql

Tabla 9-15 Coordenadas Maven de artefactos de scripts de base de datos

Norma AR-SRC-15

El proceso de **ensamblado** de los siguientes arquetipos de proyecto aplicativo debe generar como resultado un **artefacto ZIP de scripts de base de datos**:

- **Aplicación Web.**
- **Aplicación de servicios.**
- **Aplicación híbrida.**

Los artefactos **ZIP de scripts de base de datos** deben tener la **estructura interna** especificada en la **Tabla 9-14**.

El artefacto **ZIP de scripts de base de datos** debe quedar identificado por las coordenadas Maven definidas en la **Tabla 9-15**.

9.6.5 Artefactos de recursos estáticos Web

Los artefactos de recursos estáticos Web son archivos ZIP agrupando los diferentes recursos estáticos Web que se hayan definido.

Path	Descripción
/	Recursos ubicados bajo src/main/webapp/static .

Tabla 9-16 Estructura interna de artefacto de recursos estáticos Web

Coordinada	Descripción
groupId	es.mjjusticia.[systemId]
artifactId	[systemId]-[subsystemId]
packaging	zip
classifier	static

Tabla 9-17 Coordenadas Maven de artefactos de scripts de base de datos

Norma AR-SRC-16

El proceso de **ensamblado** de los siguientes arquetipos de proyecto aplicativo debe generar como resultado un artefacto **ZIP de recursos estáticos Web**:

- **Aplicación Web.**
- **Fachada Web.**
- **Aplicación híbrida.**

Los artefactos **ZIP de recursos estáticos Web** deben tener la **estructura** interna especificada en la **Tabla 9-16**.

El artefacto **ZIP de recursos estáticos Web** debe quedar identificado por las coordenadas Maven definidas en la **Tabla 9-17**.

9.7 APIs y frameworks base

Los APIs y *frameworks* base siguientes se consideran lo suficientemente maduros, estables, consolidados y adecuados a las especificaciones de diseño definidas en este documento, por lo que se establece normalizar su uso.

Norma AR-SRC-17

Se establece el uso de los siguientes **APIs y frameworks base** para el desarrollo de la parte software de los sistemas de información que se diseñen y construyan bajo esta Arquitectura de Referencia:

- [Spring](#) como *framework* base para la implementación de componentes lógicos y la gestión de su ciclo de vida.
 - El control transaccional se realizará mediante los mecanismos proporcionados por Spring a tal efecto.
 - La gestión de propiedades de configuración se realizará mediante los mecanismos proporcionados por Spring a tal efecto.
 - La planificación de tareas programadas se realizará mediante los mecanismos proporcionados por Spring a tal efecto.
 - La gestión de controladores Web según el patrón MVC se realizará mediante los mecanismos proporcionados por Spring a tal efecto.
- [JavaBeans](#) como especificación para la definición de modelos de dominio no persistentes.

- [Java Persistence API](#) como especificación para la definición de modelos de dominio persistentes.
- [Spring Data](#) como *framework* para facilitar el acceso a datos.
- [Spring Security](#) como *framework* para realizar el control de autorización y autenticación.
- [SLF4J](#) para la escritura de trazas, independiente del *framework* de gestión de trazas subyacente.
- [Thymeleaf](#) como motor de plantillas para la implementación de vistas.
- [Bean Validation API](#) como especificación para la definición de las reglas de validación sintáctica de los datos de entrada en la capa Web.
- Especificación [JAX-WS](#) para la definición de servicios Web y clientes de servicios Web.
- [Quartz](#) como *framework* de planificación de tareas.
- [JUnit](#) como *framework* de pruebas unitarias.

El listado anterior se extenderá y completará con otros APIs, *frameworks*, librerías de utilidad y herramientas, junto con la especificación de sus versiones vigentes en cada momento, conformando así la **Política de actualización de APIs, frameworks y librerías**. Esta política de actualización tiene por objeto colaborar para evitar la obsolescencia tecnológica de los sistemas de información, liberando versiones frecuentes que mantengan actualizada dicha línea base.

9.8 Paquetes

Las clases e interfaces deberán organizarse en paquetes, cuya nomenclatura refleja el sistema, subsistema y capa lógica correspondientes.

Norma AR-SRC-18

Se establece la siguiente organización en **paquetes** de las clases e interfaces de los proyectos aplicativos:

- **Paquete base**: determina la ubicación raíz, bajo la cual se ubicarán el resto de sub-paquetes, así como las clases que determinan la configuración del contexto de aplicación:
 - **es.mjjusticia.[systemId].[subsystemId]**
- Modelo de **dominio privado**, relativo al paquete base:
 - **model**
- Capa de **persistencia**, relativo al paquete base:
 - **persistence**
- Capa de **integración**, relativo al paquete base:
 - **eis**
- Capa de **negocio** y modelo de **dominio público**, relativo al paquete base:
 - **business**

- Capa **Web**, relativo al paquete base:
 - **web**
- Capa de **servicios Web**, relativo al paquete base:
 - **ws**

Donde:

- **[systemId]**: identificador del sistema de información.
- **[subsystemId]**: identificador del subsistema aplicativo, en el ámbito del sistema de información al que pertenece.

9.9 Componentes

En Spring, los componentes quedan definidos por un nombre de componente, junto con la clase que lo implementa. Esta clase, además, debe implementar el interfaz que define a dicho componente.

9.9.1 Interfaz

Un componente queda definido mediante un interfaz, que especifica los diferentes métodos que determinan su comportamiento.

Norma AR-SRC-19

Se establece que el **estereotipo, patrón de diseño o naturaleza** de cada componente quede reflejado en el **nombre del interfaz** que lo define, para identificar rápidamente su cometido y contribuir de esta manera a facilitar el mantenimiento.

Para ello, se establece que el nombre del interfaz contenga un **sufijo** representativo, como se indica a continuación:

- **Repository**: indica que el componente posee el estereotipo DAO, y que, por tanto, se ubica en la capa de persistencia.
- **Connector**: indica que el componente posee el estereotipo Conector, y que, por tanto, se ubica en la capa de integración.
- **Component**: indica que el componente posee el estereotipo Componente, y que, por tanto, se ubica en la capa de negocio. Es posible refining este sufijo aún más, especificando más información acerca de la naturaleza, función o patrón de diseño aplicado al componente, por ejemplo:
 - **BusinessFacadeComponent**: indica que el componente es una fachada de negocio.
- **WebController**: indica que el componente posee el estereotipo Controlador, y que, por tanto, se ubica en la capa Web.
- **WSController**: indica que el componente posee el estereotipo Servicio Web, y que, por tanto, se ubica en la capa de servicios Web.

- **Filter:** indica que el componente posee el estereotipo Filtro Web, y que, por tanto, se ubica en la capa Web o en la capa de servicios Web.

Los nombres de interfaz deben indicarse con estilo Upper Camel Case.

9.9.2 Implementación

Es posible disponer de una o más implementaciones de cada componente, aunque en tiempo de ejecución únicamente se empleará una de ellas.

Norma AR-SRC-20

Se establece que, en el caso de existir varias **implementaciones** de un mismo componente, ésta quede reflejada en la clase correspondiente, para identificar rápidamente su cometido y contribuir de esta manera a facilitar el mantenimiento.

Para ello, se establece que el **nombre** de la clase que implementa el componente se determine concatenando los nombres siguientes.

- **[nombre-interfaz]:** nombre del interfaz que define al componente.
- **[nombre-variante]:** nombre de la variante de la implementación. Por ejemplo, puede usarse:
 - Ningún nombre: indica que se trata de la implementación por defecto del componente.
 - **Mock:** indica que se trata de una implementación *mock* o ficticia del componente, para ser usada en el contexto de unas pruebas unitarias.
- **Impl:** indica que se trata de la implementación del componente, distinguiéndolo del interfaz.

Los nombres de clase deben indicarse con estilo Upper Camel Case.

9.9.3 Nombre

Es posible configurar Spring para que genere automáticamente el nombre de cada componente a partir del nombre de la clase que lo implementa, aunque también puede indicarse manualmente.

Si el nombre se determina automáticamente, Spring lo obtendrá a partir del nombre de la clase correspondiente, pero expresado en Lower Camel Case.

Norma AR-SRC-21

En el caso de que los **nombres** de los componentes se indiquen manualmente, éstos se deben determinar concatenando los nombres siguientes:

- **[nombre-interfaz]:** nombre del interfaz que define al componente.

- [nombre-variante]: nombre de la variante de la implementación. Por ejemplo, puede usarse:
 - Ningún nombre: indica que se trata de la implementación por defecto del componente.
 - Mock: indica que se trata de una implementación *mock* o ficticia del componente, para ser usada en el contexto de unas pruebas unitarias.

Los nombres de componente deben indicarse con estilo Lower Camel Case.

9.9.4 Estereotipo

Spring dispone de una serie de anotaciones que permiten denotar que una determinada clase se corresponde con la implementación de un componente, a la vez que especifican de qué tipo de componente se trata.

Norma AR-SRC-22

Se establece que las **anotaciones** que Spring proporciona para estereotipar componentes se utilicen como se indica a continuación.

- La implementación de los componentes de la capa de persistencia debe decorarse con la anotación **@Repository**.
- La implementación de las fachadas de negocio, en la capa de negocio, deben decorarse con la anotación **@Service**.
- La implementación de los controladores de la capa Web debe decorarse con la anotación **@Controller**.
- Para el resto de los casos, la implementación de los componentes debe decorarse con la anotación **@Component**.

9.10 Configuración del contexto de aplicación

Contexto de aplicación es la denominación que utiliza Spring para referirse a la definición del conjunto de componentes lógicos que conforman un determinado subsistema aplicativo, así como a las relaciones existentes entre ellos. Por tanto, constituye un fiel reflejo de su arquitectura lógica.

Norma AR-SRC-23

Los proyectos aplicativos de arquetipos Aplicación Web, Fachada Web, Aplicación de servicios Web o Aplicación híbrida deberán establecer la configuración de contexto a partir de una clase denominada **ApplicationConfiguration**, decorada con la anotación **@Configuration**, y ubicada directamente en el paquete base.

9.11 Modelo de dominio

Los modelos de dominio representan, mediante objetos, los distintos elementos que conforman el dominio del subsistema aplicativo.

Norma AR-SRC-24

Se establece que los **modelos de dominio** se implementen mediante clases planas o **POJO** que sigan las normas y convenios de nomenclatura determinadas por la especificación [JavaBeans](#).

Cada elemento del dominio deberá modelarse con una clase. Los atributos que definen dicho elemento del dominio deberán modelarse como un atributo privado en la correspondiente clase.

Los nombres de clase deben indicarse con estilo Upper Camel Case, mientras que los nombres de atributo deben indicarse con estilo Lower Camel Case.

El tipo de datos de los atributos podrá ser:

- Un tipo simple estándar de Java.
- Una clase que represente un objeto-valor, ya sea estándar de Java (**String**, **Integer**, **File**, etc.) o aplicativa.
- Una enumeración.
- Una clase que represente otro elemento del dominio.
- Colecciones de los tipos anteriores.

La clase deberá:

- Implementar el interfaz **Serializable**.
- Proporcionar un constructor público por defecto.
- Disponer de métodos públicos **get** (o **is** para atributos de tipo booleano), y **set** para cada atributo definido.
- Proporcionar una implementación de los métodos **hashcode**, **equals** y **toString** basada en el valor de los atributos.

Estas clases no deben disponer de ninguna lógica adicional.

Los elementos del modelo de dominio que se consideren **persistentes** deberán indicar en su nombre el sufijo **Entity**.

9.12 Excepciones

De acuerdo a su tratamiento, las excepciones podrán ser técnicas, transitorias o de negocio.

Norma AR-SRC-25

Se establece que el **nombre** de clase de cualquier **excepción** aplicativa se especifique con estilo Upper Camel Case, indicando en su nombre el sufijo **Exception**.

Las excepciones **técnicas** o **transitorias** se deberán modelar preferentemente con excepciones estándar, ya sean propias de Java, Spring u otros *frameworks*. En todo caso, deberán ser excepciones no chequeadas, esto es, subclases de **RuntimeException**.

Las excepciones de **negocio** se deberán modelar mediante clases aplicativas. En todo caso deberán ser excepciones chequeadas, esto es, subclases directas de **Exception**.

9.13 URL base

La URL base de un subsistema aplicativo en un puerto del entorno de producción constituye la raíz a través de la cual los interfaces provistos por éste son accesibles desde dicho puerto.

Esta URL base se determinará como se indica a continuación.

Norma AR-SRC-26

La **URL base** con que un subsistema aplicativo expone sus interfaces provistos a través de un puerto concreto del entorno de producción puede determinarse de dos formas.

Si el subsistema aplicativo **posee un nombre de dominio propio**, la URL base se determinará de la siguiente forma:

- **https://[[entorno]-][nombre].[sufijo]**

Si el subsistema aplicativo **comparte dominio** con otros, como es el caso de la sede electrónica, la URL base se determinará de la siguiente forma:

- **https://[[entorno]-][nombre].[sufijo]/[applicationId]**

Donde:

- **[entorno]**: identificador del entorno, solamente cuando éste sea diferente del de producción:
 - **int**: entorno de integración.
 - **qa**: entorno de calidad.
 - **ss**: entorno de servicios estables.
 - **pre**: entorno de pre-producción.
- **[nombre]**: nombre asociado al subsistema aplicativo o a la agrupación de subsistemas aplicativos.
- **[sufijo]**: parte final del nombre de dominio, asociado al puerto del entorno de producción a través del cual se realiza el acceso.
- **[applicationId]**: identificador del subsistema aplicativo.

9.14 Interfaces Web

Se establecen las políticas de normalización de interfaces Web que se indican a continuación.

9.14.1 URL

La URL base de acceso al interfaz Web deberá tener la estructura que se indica a continuación.

Norma AR-SRC-27

La URL base del interfaz Web que provea un subsistema aplicativo será:

- [url-base]/

Donde:

- [url-base]: URL base del subsistema aplicativo en el puerto correspondiente.

La anterior URL deberá constituir el **punto de entrada** al interfaz Web, debiendo tener, por tanto, una vista asociada, o bien respondiendo con una re-dirección a otro path más adecuado.

9.14.2 Mecanismos de autenticación

La ubicación de los mecanismos de autenticación quedará normalizada como se indica a continuación.

Norma AR-SRC-28

Los diferentes mecanismos de autenticación tendrán reservados paths específicos bajo la URL base del subsistema aplicativo.

El mecanismo de autenticación mediante **Cl@ve** empleará las siguientes URL:

- [url-base]/auth/clave/login/request
- [url-base]/auth/clave/login/response
- [url-base]/auth/clave/logout/request
- [url-base]/auth/clave/logout/response

El mecanismo de autenticación mediante **Autentica** empleará las siguientes URL:

- [url-base]/auth/autentica/login/request
- [url-base]/auth/autentica/login/response
- [url-base]/auth/autentica/logout/request
- [url-base]/auth/autentica/logout/response

El mecanismo de autenticación mediante **Active Directory** empleará las siguientes URL:

- [url-base]/auth/ad/form
- [url-base]/auth/ad/login
- [url-base]/auth/ad/logout

Los mecanismos anteriores re-direcccionarán a las siguientes URL para la visualización de los **errores** de autenticación o autorización:

- [url-base]/auth/error/authentication-error
- [url-base]/auth/error/authorization-error

Donde:

- [url-base]: URL base del subsistema aplicativo en el puerto correspondiente.

9.14.3 Recursos estáticos Web compartidos

Pueden definirse recursos estáticos Web compartidos, asociados a las maquetas empleadas por los distintos interfaces Web provistos por todos los subsistemas aplicativos que los provean a través de un mismo puerto del entorno de producción.

La ubicación de estos recursos estáticos compartidos quedará normalizada como se indica a continuación.

Norma AR-SRC-29

Las maquetas y, por tanto, los recursos estáticos compartidos vinculados a ellas, podrán emplearse en interfaces Web que comparten nombre de dominio.

Dichos recursos se ubicarán bajo:

- `https://[[entorno]-][nombre].[sufijo]/static/[maqueta]/[vs]/`

Donde:

- [entorno]: identificador del entorno, solamente cuando éste sea diferente del de producción:
 - int: entorno de integración.
 - qa: entorno de calidad.
 - ss: entorno de servicios estables.
 - pre: entorno de pre-producción.
- [nombre]: nombre asociado a la agrupación de subsistemas aplicativos.
- [sufijo]: parte final del nombre de dominio, asociado al puerto del entorno de producción a través del cual se realiza el acceso.
- [maqueta]: nombre o identificador de la maqueta correspondiente.
- [vs]: nombre o identificador de la versión de los recursos estáticos asociados a la maqueta correspondiente.

Los formatos de los recursos estáticos deben estar incluidos en el catálogo de estándares del Esquema Nacional de Interoperabilidad.

9.14.4 Recursos estáticos Web aplicativos

La ubicación de los recursos estáticos quedará normalizada como se indica a continuación.

Norma AR-SRC-30

Los recursos estáticos aplicativos se ubicarán bajo:

- [url-base]/static

Donde:

- [url-base]: URL base del subsistema aplicativo en el puerto correspondiente.

9.15 Interfaces WS

Para conseguir cierto grado de consistencia entre los diferentes servicios Web, se establecen las políticas de normalización que se indican a continuación.

9.15.1 URL

La URL de acceso a los interfaces WS deberá tener la estructura que se indica a continuación.

Norma AR-SRC-31

La **URL** de cada **interfaz WS** accesible desde un determinado **puerto** del entorno de producción será la siguiente:

- [url-base]/ws/[fachada]

Donde:

- [url-base]: URL base del subsistema aplicativo en el puerto correspondiente.
- [fachada]: nombre o identificador de la fachada de negocio expuesta como servicio Web, con estilo Kebab Case empleando únicamente minúsculas.

9.15.2 Espacios de nombres

Los espacios de nombres XML, o *namespaces*, son mecanismos que permite evitar conflictos de nombres mediante la diferenciación de elementos o atributos en un documento XML que pueden tener nombres idénticos, pero diferentes

especificaciones. Para ello se establecen prefijos únicos, con sentido global, y con formato de URI.

Norma AR-SRC-32

Se establecen los siguientes convenios de nomenclatura relativos a los **espacios de nombres**:

- **URI base** para los espacios de nombres de un subsistema aplicativo:
 - `http://[subsystemId].[systemId].mjusticia.es/ws`
- Ítems especificados en **WSDL** (mensajes, puertos, bindings y servicios), relativo al URI base:
 - `spec/[fachada]`
- Ítems especificados en **XML Schema** (tipos de datos y elementos XML) **específicos** de una fachada de negocio, relativo al URI base:
 - `spec/[fachada]`
- Ítems especificados en **XML Schema** (tipos de datos y elementos XML) que sean **compartidos** por diferentes fachadas de negocio, relativo al URI base:
 - `spec/model`

Donde:

- **[systemId]**: identificador del sistema de información.
- **[subsystemId]**: identificador del subsistema aplicativo, en el ámbito del sistema de información al que pertenece.
- **[fachada]**: nombre o identificador de la fachada de negocio expuesta como servicio Web, con estilo Kebab Case empleando únicamente minúsculas.

9.15.3 Cabecera SOAPAction

La cabecera SOAPAction que acompaña las peticiones a los servicios Web también es susceptible de normalización.

Norma AR-SRC-33

Se establecen los siguientes convenios de nomenclatura relativos las cabeceras SOAPAction:

- **URI base**:
 - `http://[subsystemId].[systemId].mjusticia.es/ws`
- **Cabecera SOAPAction** asociada a una operativa de negocio expuesta por una fachada de negocio, relativa al URI base:
 - `[fachada]/[operativa]`

Donde:

- **[systemId]**: identificador del sistema de información.

- **[subsystemId]**: identificador del subsistema aplicativo, en el ámbito del sistema de información al que pertenece.
- **[fachada]**: nombre o identificador de la fachada de negocio expuesta como servicio Web, con estilo Kebab Case empleando únicamente minúsculas.
- **[operativa]**: nombre o identificador de la operativa de negocio, con estilo Kebab Case empleando únicamente minúsculas.

9.16 Objetos de la base de datos

Los nombres de los objetos de la base de datos se determinarán mediante las pautas generales que se indican a continuación.

Norma AR-SRC-34

Los **nombres** de los **objetos de la base de datos** poseen las siguientes restricciones:

- Solamente pueden contener caracteres ASCII alfanuméricos.
- Deben comenzar por un carácter alfabético.
- Se deben expresar con estilo Snake Case.
- No se distinguen mayúsculas de minúsculas, debiéndose utilizar mayúsculas por convenio.
- No deben tener más de 30 caracteres.

Si un nombre excede el tamaño máximo de 30 caracteres, debe determinarse una **abreviatura** para usar en su lugar, siguiendo para ello las pautas siguientes:

- Si el nombre solo consta de una palabra: la abreviatura se determinará seleccionando cuatro caracteres representativos de ésta.
- Si el nombre consta de dos palabras: la abreviatura se determinará concatenando dos caracteres representativos de cada una.
- Si el nombre consta de tres palabras: la abreviatura se determinará concatenando dos caracteres representativos de la primera palabra, y un carácter representativo de cada una de las restantes.
- Si el nombre consta de cuatro palabras o más: la abreviatura se determinará concatenando un carácter representativo de cada una de las palabras.

Los nombres de los objetos de la base de datos se normalizarán como se indica a continuación.

Norma AR-SRC-35

El **esquema** de un subsistema aplicativo se denominará de la siguiente forma:

- **[applicationId]**

Donde:

- **[applicationId]**: identificador único del subsistema aplicativo, o su abreviatura.

Norma AR-SRC-36

Las **tablas** se denominarán en plural, empleando Snake Case.

Norma AR-SRC-37

Las **columnas** se denominarán en singular, empleando Snake Case. No se utilizarán prefijos.

- Las **claves primarias subrogadas** se denominarán de la siguiente forma:
 - **[nombre-tabla]_PK**
- Las **claves foráneas** se denominarán de la siguiente forma:
 - Si la clave de la tabla referenciada es **subrogada**:
 - **[nombre-tabla-dependiente]_[nombre-tabla-referenciada]_FK**
 - Si la clave de la tabla referenciada es **natural**:
 - **[nombre-tabla-dependiente]_[nombre-tabla-referenciada]_[nombre-columna]**

Donde:

- **[nombre-tabla]**: nombre o abreviatura de la tabla.
- **[nombre-tabla-dependiente]**: nombre o abreviatura de la tabla que contiene la clave foránea.
- **[nombre-tabla-referenciada]**: nombre o abreviatura de la tabla referenciada.
- **[nombre-columna]**: nombre de la columna que constituye la clave natural.

Norma AR-SRC-38

Los **índices** se denominarán de la forma siguiente:

- Los **índices sobre claves foráneas** se denominarán:
 - **[nombre-clave-foránea]_I**
- El resto de índices se denominarán de la siguiente forma:
 - **[nombre-tabla]_[nombre Campos]_I**

Donde:

- **[nombre-clave-foránea]**: nombre de la columna correspondiente a la clave foránea.
- **[nombre-tabla]**: nombre o abreviatura de la tabla a la que se refiere el índice.

- [nombre-campos]: nombre o abreviatura de los campos a los que se refiere el índice, en SnakeCase. Si el nombre resultase demasiado largo, utilizar un nombre representativo del conjunto de los campos.

Norma AR-SRC-39

Los **triggers** se denominarán de la forma siguiente:

- [nombre-tabla]_[nombre-acción]_TRG

Donde:

- [nombre-tabla]: nombre o abreviatura de la tabla.
- [nombre-acción]: nombre o abreviatura de la acción realizada por el trigger.

Norma AR-SRC-40

Las **secuencias** se denominarán de la forma siguiente:

- [nombre-secuencia]_SEQ

Donde:

- [nombre-secuencia]: nombre o abreviatura de la secuencia.

Norma AR-SRC-41

Las **constraints** o restricciones se denominarán de la forma siguiente:

- Restricción de **comprobación**:
 - [nombre-comprobación]_CHK
- Restricción de **clave primaria**:
 - [nombre-tabla]_PK
- Restricción de **clave foránea**:
 - [nombre-tabla]_FK[i]
- Restricción de **unicidad**:
 - [nombre-tabla]_U[i]

Donde:

- [nombre-comprobación]: nombre o abreviatura de la comprobación.
- [nombre-tabla]: nombre o abreviatura de la tabla.
- [i]: índice numérico que garantice que todas las restricciones de este tipo son únicas, comenzando en 1.

9.17 Direcciones de correo

Los subsistemas aplicativos que requieran realizar envíos de correo electrónico, deberán emplear una cuenta de correo como se indica a continuación.

Norma AR-SRC-42

Los subsistemas aplicativos que requieran de envío de **correo electrónico** deben emplear para ello una cuenta de correo cuyo nombre siga el siguiente patrón:

- **no-responder.[systemId].[.entorno]@[dominio]**

Donde:

- **[systemId]**: identificador del sistema de información
- **[entorno]**: identificador del entorno donde se encuentra implantado el sistema de información que realiza el envío, solamente cuando éste sea diferente del de producción:
 - **int**: entorno de integración.
 - **qa**: entorno de calidad.
 - **ss**: entorno de servicios estables.
 - **pre**: entorno de pre-producción.
- **[dominio]**: nombre de dominio del correo corporativo.

10 Arquitectura de la configuración

La arquitectura de la configuración establece la especificación, ubicación y tratamiento de los recursos de configuración asociados a los diferentes subsistemas aplicativos.

Se consideran los siguientes recursos de configuración:

- Propiedades de configuración.
- Configuración de trazas.
- Recursos provistos por el servidor de aplicaciones.

Cada recurso de configuración habrá de tener además un propietario, que será el equipo encargado de determinar su valor.

10.1 Categorización de los recursos de configuración

Los recursos de configuración se pueden categorizar en base a los aspectos que se definen a continuación.

10.1.1 Ámbito

Cada recurso de configuración tiene un ámbito asociado, en función de la naturaleza de los componentes lógicos que la consumirán.

Los recursos de configuración de ámbito **técnico** permiten la configuración de componentes técnicos que forman parte de la arquitectura lógica del subsistema aplicativo. Estos componentes vienen proporcionados por los distintos *frameworks*, y los puede utilizar cualquier subsistema aplicativo.

Los recursos de configuración de ámbito **funcional** permiten la configuración de los distintos componentes de negocio que forman parte de la arquitectura lógica del subsistema aplicativo. Estos componentes, por tanto, son aplicativos, esto es, han sido concebidos durante el diseño del subsistema aplicativo y serán implementados durante su construcción.

10.1.2 Visibilidad

En función de su visibilidad, los recursos de configuración se pueden clasificar según se indica a continuación.

- **Entorno:** los recursos con visibilidad de entorno son compartidos por todos los subsistemas aplicativos desplegados en el entorno. Sus valores pueden diferir de un entorno a otro.
- **Aplicación:** los recursos con visibilidad de aplicación son propios de un subsistema aplicativo concreto, no siendo visibles por los demás subsistemas aplicativos desplegados en el entorno.

- **Test:** los recursos con visibilidad de test son propios de los test unitarios de un subsistema aplicativo concreto, no siendo visibles por ningún subsistema aplicativo desplegado en el entorno.

10.1.3 Mutabilidad

La mutabilidad es la posibilidad de cambiar el valor de un recurso de configuración en tiempo de explotación. En función de esta característica, los recursos de configuración se pueden clasificar según se indica a continuación.

- **Mutables:** su valor puede ser modificado en tiempo de explotación, realizando el cambio correspondiente en una intervención, seguido de un reinicio ordenado de los servidores de aplicaciones del subsistema o subsistemas aplicativos afectados, para que el nuevo valor tenga efecto.
- **Inmutables:** su valor no puede ser modificado en tiempo de explotación.

10.2 Propiedades de configuración

Las propiedades de configuración son pares nombre-valor que los subsistemas aplicativos leen en tiempo de arranque, desde diferentes fuentes u orígenes.

Estas fuentes constituyen contenedores donde su propietario puede definir propiedades de configuración categorizadas de cierta manera predeterminada, distinguiéndose las fuentes que se describen en los apartados siguientes.

Los valores de las propiedades de configuración podrán hacer uso de variables para referirse al valor de otras propiedades, como se indica a continuación.

Norma AR-CFG-01

Los valores de las propiedades de configuración podrán hacer uso de **variables** para referirse al valor de otras propiedades de configuración cargadas con anterioridad, incluso durante el mismo proceso de inicialización de la configuración.

Para denotarlas, se utilizarán los delimitadores \${ } y } para contener el nombre de la propiedad referenciada.

Norma AR-CFG-02

Los valores de las propiedades de configuración podrán indicarse **encriptados**.

Para denotarlo, se utilizarán los delimitadores ENC() para contener el valor encriptado.

10.2.1 Recurso application.properties

Permite definir propiedades de configuración relativas a los componentes lógicos de negocio del subsistema aplicativo.

Norma AR-CFG-03

El recurso de configuración **application.properties**, especificado en la **Tabla 10-1**, permite a sus propietarios declarar propiedades de configuración relativas a los componentes lógicos de negocio del subsistema aplicativo.

En este recurso de configuración se deberán incluir las siguientes propiedades:

- **application.systemId**: identificador del sistema de información.
- **application.subsystemId**: identificador del subsistema aplicativo, en el ámbito del sistema al que pertenece.
- **application.version**: versión del artefacto WAR asociado al subsistema aplicativo.
- **application.buildDate**: fecha de ensamblado del artefacto WAR asociado al subsistema aplicativo.

Este recurso de configuración debe generarse automáticamente en tiempo de ensamblado de proyectos aplicativos que generen un artefacto WAR, formando parte del mismo.

Los valores de las propiedades anteriores deben obtenerse a partir de **variables** definidas en el archivo **pom.xml**, mediante el **filtrado de recursos**.

application.properties		
Ubicación	Proyecto	/src/main/resources/application.properties
	Artefacto	/application.properties
	Servidor	N/A
Propietario	Equipo de desarrollo y mantenimiento del subsistema aplicativo	
Categoría	Ámbito	Funcional
	Visibilidad	Aplicación
	Mutabilidad	Inmutable

Tabla 10-1 Especificación del recurso application.properties

10.2.2 Recurso environment.properties

Permite definir propiedades de configuración relativas a parámetros físicos propios del entorno, tales como URL, DNS, puertos, rutas de recursos compartidos, nombres de usuario, contraseñas, etc., que serán utilizadas por componentes lógicos de carácter técnico y que serán comunes a todos los subsistemas aplicativos.

Norma AR-CFG-04

El recurso de configuración **environment.properties**, especificado en la **Tabla 10-2**, permite a sus propietarios declarar propiedades de configuración relativas a

componentes lógicos de carácter técnico y de uso compartido por todos los subsistemas aplicativos desplegados en el mismo entorno.

Podrán existir diferentes versiones de este recurso de configuración, distinguidas por el número de versión [versión].

Las propiedades de configuración definidas en este fichero están normalizadas y documentadas en la **Especificación de propiedades de configuración compartidas**.

Los valores de estas propiedades están determinados y mantenidos por el equipo administrador de cada entorno, ya que, en general, serán distintos en cada uno.

En este recurso de configuración se deberán incluir las siguientes propiedades:

- **env.environmentId**: identificador del entorno. Los valores posibles son:
 - **dev**: entorno de desarrollo (local).
 - **int**: entorno de integración.
 - **qa**: entorno de calidad.
 - **ss**: entorno de servicios estables.
 - **pre**: entorno de pre-producción.
 - **pro**: entorno de producción.
- **env.seedLocation**: ruta del fichero que contiene la semilla con la que resolver las propiedades cuyo valor se indica encriptado.
- **env.serverName**: nombre de host del servidor de aplicaciones.
- **env.nodeName**: nombre de nodo del servidor de aplicaciones.

Este recurso de configuración se deberá ensamblar en el artefacto de recursos de configuración compartida.

environment.properties		
Ubicación	Proyecto	N/A
	Artefacto	/[entorno]/environment.properties
	Servidor	/opt/config/[versión]/environment.properties
Propietario	Equipo administrador del entorno	
Categoría	Ámbito	Técnico
	Visibilidad	Entorno
	Mutabilidad	Mutable

Tabla 10-2 Especificación del recurso environment.properties

10.2.3 Recurso administrado application.properties

Permite declarar propiedades de configuración específicas de un subsistema aplicativo concreto, que serán utilizadas por componentes lógicos de carácter técnico.

Las propiedades de configuración definidas en este recurso estarán especificadas y documentadas por el equipo de desarrollo y mantenimiento del subsistema aplicativo, pero estarán administradas por el equipo administrador del entorno.

El equipo de desarrollo y mantenimiento del subsistema aplicativo deberá solicitar, mediante petición al equipo administrador del entorno, la creación, modificación o eliminación de las propiedades de configuración declaradas en este fichero.

Norma AR-CFG-05

El recurso de configuración administrado **application.properties**, especificado en la **Tabla 10-3**, permite a sus propietarios declarar propiedades de configuración relativas a componentes lógicos de carácter técnico y de uso específico por un subsistema aplicativo concreto.

Este recurso de configuración será administrado por el equipo administrador del entorno, bajo petición de su propietario, que será el equipo de desarrollo y mantenimiento del subsistema aplicativo.

Podrán existir diferentes versiones de este recurso de configuración, distinguidas por el número de versión **[versión]**.

Este recurso de configuración se deberá ensamblar en el artefacto de recursos de configuración administrada aplicativa.

application.properties		
Ubicación	Proyecto	N/A
	Artefacto	/[entorno]/application.properties
	Servidor	/opt/config/app/[applicationId]/[versión]/application.properties
Propietario	Equipo de desarrollo y mantenimiento del subsistema aplicativo.	
Administrador	Equipo administrador del entorno	
Categoría	Ámbito	Técnico
	Visibilidad	Aplicación
	Mutabilidad	Mutable

Tabla 10-3 Especificación del recurso administrado application.properties

10.2.4 Recurso test-application.properties

Permite declarar propiedades de configuración específicas de un subsistema aplicativo concreto, que serán utilizadas por componentes lógicos de carácter técnico, válidas para la ejecución de test unitarios.

Norma AR-CFG-06

El recurso de configuración **test-application.properties**, especificado en la **Tabla 10-4**, permite a sus propietarios declarar propiedades de configuración relativas a componentes lógicos de carácter técnico y de uso específico por un subsistema aplicativo concreto.

test-application.properties		
Ubicación	Proyecto	/src/test/resources/test-application.properties
	Artefacto	N/A
	Servidor	N/A
Propietario	Equipo de desarrollo y mantenimiento del subsistema aplicativo.	
Categoría	Ámbito	Técnico
	Visibilidad	Test
	Mutabilidad	N/A

Tabla 10-4 Especificación del recurso test-application.properties

10.2.5 Carga de las propiedades de configuración

Las propiedades de configuración deberán cargarse en tiempo de arranque como se indica a continuación.

Norma AR-CFG-07

Los recursos que definen propiedades de configuración deberán cargarse en el **orden** que se indica a continuación, utilizando los mecanismos que proporciona **Spring** para ello.

En un entorno de ejecución:

- Recurso **application.properties**, desde el classpath.
- Recurso **environment.properties**, desde el sistema de archivos, debiéndose cargar la versión requerida por el subsistema aplicativo.
- Recurso administrado **application.properties**, desde el sistema de archivos, debiéndose cargar la última versión disponible.

En test unitarios:

- Recurso **application.properties** desde el classpath.
- Recurso **test-application.properties** desde el classpath.

Si una propiedad está repetida, deberá prevalecer el último valor, según el orden de procesamiento anterior.

Durante la carga de las propiedades deberán resolverse las variables que puedan estar incluidas en su valor, o bien des-criptar los valores que estén cifrados.

Una vez arrancado el contenedor, los distintos componentes podrán utilizar las propiedades, bien durante su arranque, o bien en tiempo de ejecución, usando los mecanismos que proporciona Spring para ello.

10.3 Configuración de trazas

La configuración de trazas dispondrá de una parte genérica, común a todos los subsistemas aplicativos, y una parte aplicativa, específica de cada subsistema aplicativo.

10.3.1 Recurso logging.properties

Permite definir propiedades de configuración de trazas de carácter genérico, constituyendo una plantilla de configuración donde se definen componentes técnicos de uso general para la escritura de trazas, pre-configurados para dar cabida a las trazas generadas tanto por el código aplicativo como por los diferentes *frameworks*.

Norma AR-CFG-08

El recurso de configuración **logging.properties**, especificado en la **Tabla 10-5**, determina la configuración básica para la escritura de trazas por parte de cualquier subsistema aplicativo.

Podrán existir diferentes versiones de este recurso de configuración, distinguidas por el número de versión **[versión]**.

Para adaptar la configuración a un subsistema aplicativo concreto, se podrá hacer uso de **variables**, referenciando cualquiera de las propiedades del recurso **application.properties**.

Las propiedades de configuración definidas en este recurso de configuración están normalizadas y documentadas en la **Especificación de propiedades de configuración de trazas**.

Este recurso de configuración se deberá ensamblar en el artefacto de recursos de configuración compartida.

logging.properties		
Ubicación	Proyecto	N/A
	Artefacto	/[entorno]/logging.properties
	Servidor	/opt/config/[versión]/logging.properties
Propietario	Equipo administrador del entorno	
Categoría	Ámbito	Técnico
	Visibilidad	Entorno
	Mutabilidad	Inmutable

Tabla 10-5 Especificación del recurso logging.properties

10.3.2 Recurso administrado logging.properties

Permite completar la configuración de trazas, añadiendo configuración específica para un subsistema aplicativo concreto.

Norma AR-CFG-09

El recurso de configuración administrado **logging.properties**, especificado en la **Tabla 10-6**, completa la configuración de trazas, añadiendo configuración específica para un subsistema aplicativo concreto.

Las propiedades de configuración que es posible definir en este recurso de configuración están normalizadas y documentadas en la **Especificación de propiedades de configuración de trazas**.

Este recurso de configuración se deberá ensamblar en el artefacto de recursos de configuración administrada aplicativa.

logging.properties		
Ubicación	Proyecto	N/A
	Artefacto	/[entorno]/logging.properties
	Servidor	/opt/config/app/[applicationId]/[versión]/logging.properties
Propietario	Equipo de desarrollo y mantenimiento del subsistema aplicativo.	
Administrador	Equipo administrador del entorno	
Categoría	Ámbito	Técnico
	Visibilidad	Aplicación
	Mutabilidad	Mutable

Tabla 10-6 Especificación del recurso administrado logging.properties

10.3.3 Recurso test-logging.properties

Permite definir propiedades de configuración de trazas, válidas para la ejecución de test unitarios.

Norma AR-CFG-10

El recurso de configuración **test-logging.properties**, especificado en la **Tabla 10-7**, determina la configuración de escritura de trazas en la ejecución de test unitarios.

Test-logging.properties		
Ubicación	Proyecto	/src/test/resources/test-logging.properties
	Artefacto	N/A
	Servidor	N/A
Propietario	Equipo administrador del entorno	
Categoría	Ámbito	Técnico
	Visibilidad	Test
	Mutabilidad	N/A

Tabla 10-7 Especificación del recurso test-logging.properties

10.3.4 Carga de la configuración de trazas

La carga de la configuración de trazas y la inicialización del componente lógico de gestión de trazas deben realizarse de forma previa a la inicialización del contenedor de inversión de control.

Norma AR-CFG-11

La **carga** de la configuración de trazas, así como la **inicialización** del componente lógico encargado de su gestión, deben realizarse de forma **previa** al arranque del contenedor de inversión de control, como parte del arranque del artefacto WAR en los servidores de aplicaciones.

Para obtener la configuración de trazas, se deben cargar los siguientes recursos de configuración en el orden indicado.

En un entorno de ejecución:

- Recurso **logging.properties**, desde el sistema de archivos, debiéndose cargar la versión requerida por el subsistema aplicativo.
- Recurso administrado **logging.properties**, desde el sistema de archivos, debiéndose cargar la última versión disponible.

En test unitarios:

- Recurso **test-logging.properties** desde el classpath.

Si una propiedad está repetida, deberá prevalecer el último valor, según el orden de procesamiento anterior.

Durante la carga de las propiedades deberán resolverse las variables que puedan estar incluidas en su valor.

Finalizada la determinación de la configuración de trazas, deberá inicializarse el componente lógico de gestión de trazas.

Si no fuese posible iniciar el componente lógico de gestión de trazas, se deberá lanzar una excepción, impidiendo el arranque del artefacto WAR en el servidor de

aplicaciones. De lo contrario, se perdería la trazabilidad del subsistema aplicativo, lo cual constituye una falla de seguridad.

Una vez inicializado con éxito el componente lógico de gestión de trazas, se proseguirá con la inicialización del contenedor de inversión de control.