Guía de desarrollo, Anexo 36

Estándar de Arquitectura y Diseño

Área de Arquitectura

GERENCIA INFORMÁTICA
JOSEFA VALCÁRCEL, 44
28027-MADRID

Índice General

1 INTRODUCCIÓN 4

1.1 OBJETIVO 4
1.2 AUDIENCIA 4
1.3 GLOSARIO 4
1.3.1 Acrónimos 4

2 ARQUITECTURA FÍSICA DGT 5

3 ARQUITECTURA LÓGICA DEL SISTEMA 7

3.1 ARQUITECTURA DEL SISTEMA 7
3.1.1 Entregable de Arquitectura 8
3.2 ARQUITECTURA COMÚN DGT 11
3.2.1 Presentación 12
3.2.2 Adaptadores de Servicios (Capa de servicios) 14
3.2.3 Negocio 15
3.2.4 Persistencia 16
3.2.5 Capas transversales 19
3.2.6 Adaptador Salida 20
3.3 DATAPOWER 21
3.4 INTEGRACIÓN DE LOS FRAMEWORKS DE CÓDIGO ABIERTO CON EL SERVIDOR J2EE 22
3.4.1 Spring 22
3.5 REGLAS DE DISEÑO 25
3.5.1 Reglas generales 25
3.5.2 Reglas de referencia a recursos JNDI 27
3.5.3 Reglas para despliegue en clúster 28
3.5.4 Reglas en la capa de presentación 28
3.5.5 Reglas en la capa de servicios 30
3.5.6 Reglas en la capa de negocio 30
3.5.7 Reglas en la capa de persistencia 30
3.5.8 Reglas de persistencia en Host 30
3.5.9 Reglas en las capas Host 31
3.5.10 Excepciones 32

4 MODELO DE DISEÑO 33

4.1 NOMENCLATURA DE MÓDULOS 33
4.1.1 Módulos en el contexto de la especificación DGTE-001: Desarrollo de servicios de componentes de negocio y de acceso a datos para las aplicaciones Java EE. 38
4.2 NOMENCLATURA DE LOS PAQUETES 42
4.3 FRAMEWORK DGT 45
4.3.1 Componentes Comunes 45
4.3.2 Servicios de Infraestructura 45

5 ESTÁNDAR DE CONSTRUCCIÓN 48

5.1 CONFIGURACIÓN DE DESPLIEGUE 48
5.2 SCRIPTS DE BASE DE DATOS 48
5.3 FUENTES DEL HOST 50
5.4 IMPLEMENTACIÓN DE USUARIO DE CONTROL/ PÁGINA DE TEST 50
5.4.1 Diseño 50
5.4.2 Implementación 51
5.4.3 Monitorización 52

5.5 CONTENIDOS ESTÁTICOS **52**
5.6 FICHEROS DE LOGS **53**
5.6.1 Eventos.log *53*
5.6.2 Actividad.log *54*
5.6.3 Debug.log *54*
5.7 COLAS JMS **54**
5.8 FICHEROS DE PROPIEDADES **55**
5.8.1 Estructura directorios *55*
5.8.2 Ficheros de recursos *55*
5.8.3 Ficheros de configuración *55*
5.9 PROCESOS BATCH **56**
5.9.1 Tipologías *56*
5.9.2 Estructura de los procesos batch *57*
5.10 SERVICIO CORPORATIVO DE DIRECTORIO LDAP **58**
5.10.1 Entornos y configuración de clientes / conexiones *58*
5.10.2 Políticas aplicadas en los LDAP sobre los usuarios *58*
5.10.3 Políticas aplicadas en los LDAP sobre proyectos y perfiles *58*

6 CAMBIOS Y NOVEDADES *60*

6.1 GUÍA DE DESARROLLO 2019. ESTÁNDAR DE ARQUITECTURA Y DISEÑO **60**
6.1.1 Documentación de Diseño de Arquitectura: *60*
6.1.2 Capas lógicas de Arquitectura lógica de aplicaciones *60*
6.1.3 Nomenclatura de módulos. *61*
6.1.4 Configuración de aplicaciones *61*
6.1.5 Componentes y servicios comunes: *61*

Índice de Ilustraciones y Tablas

Ilustración 1. Arquitectura física DGT 5
Ilustración 2. Arquitectura común DGT 12
Ilustración 3. Ejemplo de adaptación de distribución de contenidos 13
Ilustración 4. Resumen Nomenclatura de Módulos y Paquetes 44
Ilustración 5. Ejemplo de definición de grupos LDAP 59

Tabla 1. Nombrado de recursos JNDI 28
Tabla 2. Estructura general del repositorio para código fuente 37
Tabla 3. Estructura repositorio de código fuente para la especificación DGTE-001 40
Tabla 4. Estructura repositorio de código fuente para adaptadores de servicios web 41
Tabla 5. Estructura repositorio de código fuente para adaptadores EJB 41
Tabla 6. Estructura general de carpetas de archivos de configuración para el despliegue 48
Tabla 7. Estructura general de carpetas de archivos de base de datos 49
Tabla 8. Estructura general de carpetas de archivos de Host 50

1 Introducción

1.1 Objetivo

El presente documento detalla los entregables pertenecientes a la fase de diseño.

1.2 Audiencia

Este documento está dirigido a todas las personas que colaboren en labores relacionadas con la gestión, desarrollo, auditoría, implantación y explotación de los sistemas de información de la gerencia de informática de la Dirección General de Tráfico.

1.3 Glosario

1.3.1 Acrónimos

En este apartado se incluyen, organizados por orden alfabético, los acrónimos que se utilizan en el propio documento para facilitar su lectura y comprensión.

- **DGT:** Dirección General de Tráfico
- **LDAP:** Lightweight Directory Access Protocol
- **TWS:** Tivoli Workload Scheduler
- **VPN:** Virtual Private Network

2 Arquitectura Física DGT

A continuación se muestra un diagrama del diseño de la arquitectura física de la DGT:

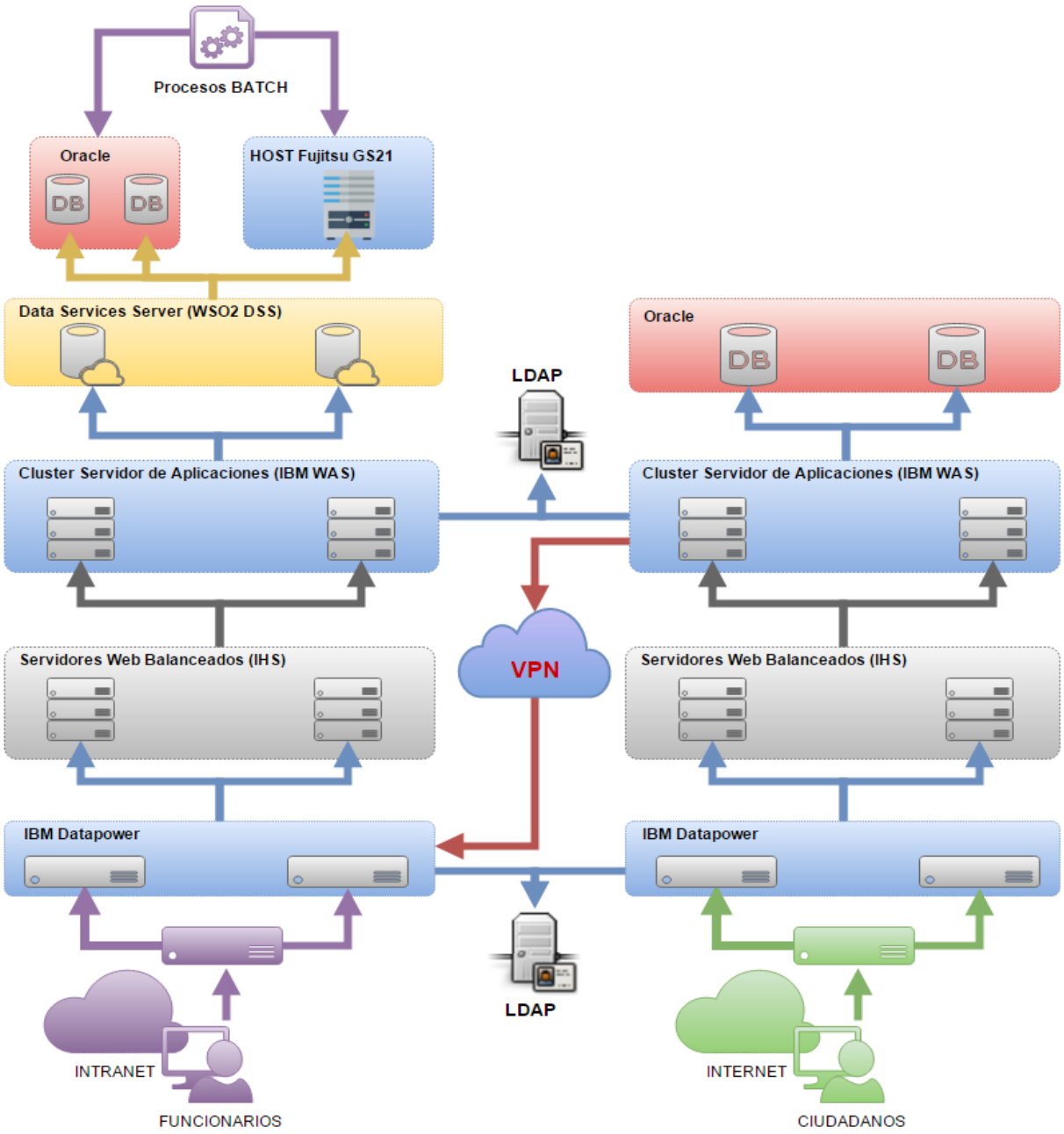

Ilustración 1. Arquitectura física DGT

Podemos identificar dos partes claramente diferenciadas, en primer lugar tenemos la arquitectura usada para la intranet corporativa y las aplicaciones para el ejercicio de potestades de la DGT compuesta de:

- Bases de datos Oracle 11g conectadas a un sistema de almacenamiento masivo.
- Servidores de aplicaciones WebSphere 7.0 instalados en un clúster activo.
- Servidores de acceso a datos WSO2 DSS 3.2.2 instalados en un clúster activo.
- Granja de servidores web balanceados por Datapower.
- Enterprise Service Bus (ESB) Datapower (balanceados por un balanceador físico).
- Un servidor que ejecuta, de forma independiente, los procesos batch de carga o consulta de la base de datos o el host.
- Un host propietario al que se accede usando el componente común y solicitando, previamente, el desarrollo del script que obtenga los datos necesarios para la aplicación.

Para las aplicaciones servidas a los ciudadanos a través de Internet se dispone de un centro externo que replica la estructura definida para la red interna con las siguientes peculiaridades:

- Las bases de datos no contienen información especialmente protegida, siendo necesario solicitar esta información a través de servicios a los servidores web de la red interna.
- Para comunicarse con la red interna se dispone de acceso a través de VPN, pero únicamente a los servicios publicados en los servidores de Intranet.
- El LDAP está totalmente replicado.
- No se pueden desplegar procesos batch.

Es importante destacar que existe un entorno de preproducción donde realizar las pruebas funcionales y de carga del sistema como paso previo a la recepción e instalación definitiva del proyecto.

3 Arquitectura Lógica del Sistema

3.1 Arquitectura del Sistema

Se establecerá un modelo de n-capas que sea válido tanto para uso con clientes ligeros como pesados, locales o remotos, sin necesidad de reescribir ningún código. Dichas capas estarán claramente separadas y tendrán que ser fáciles de mantener, así como ser intercambiables por otra implementación de la capa sin afectar al código escrito ni a las otras capas. Para permitir esto último se trabajará con interfaces para la comunicación entre capas. El modelo de capas se basará en el patrón MVC (Modelo-Vista-Controlador) y en el patrón de arquitectura hexagonal (también llamado de puertos y adaptadores).

El diseño orientado a objetos se basará en la garantía tanto de una alta cohesión como de un bajo acoplamiento, respetando el patrón SOLID de diseño, el cual agrupa los siguientes patrones universalmente aceptados:

- SRP: Single Responsability Principle
- OCP: Open/Closed Principle
- LSP: Liskov Substitution Principle
- ISP: Interface Segregation Principle
- DIP: Dependency Inversion Principle

Deberán tenerse en cuenta los siguientes principios a la hora de crear la estructura del proyecto, de cara a asegurar la cohesión de los módulos:

- CCP: Common Closure Principle. Las clases que cambian juntas deben ser empaquetadas juntas.
- CRP: Common Reuse Principle. Las clases que se usan juntas deben empaquetarse juntas.
- ADP: Acyclic Dependency Principle. No deben crearse referencias cíclicas entre módulos.

Del seguimiento de los anteriores principios se desprende que las aplicaciones deben diseñarse de forma modular para que cuando se realicen los distintos subsistemas a nivel de código (o en su

caso, se divida el código entre distintas unidades de despliegue) se evite que haya código no utilizado dentro de cada unidad de despliegue.

3.1.1 Entregable de Arquitectura

Deberá entregarse un documento "Diseño de Arquitectura" en el que se especifique la arquitectura lógica del sistema, que deberá adaptarse a las normas de arquitectura establecidas en la DGT. El documento debe incorporar la descripción de los siguientes aspectos.

3.1.1.1 Capas lógicas y entorno tecnológico

Se deben enumerar y describir las capas lógicas en la que se encuentra dividida la unidad de despliegue. En la descripción de cada capa se debe de describir la tecnología utilizada (librerías destacadas, framework utilizado, protocolos soportados, etc.)

3.1.1.1.1 Adaptadores de Entrada

- Tipo de Interfaz (REST, SOAP, EJB)
- Tipo de estándar (JAX-RS, JAX-WS, JAX-RPC, …)
- Tecnología (Jersey, RestEasy, Ingeniería webservices proporcionada por el servidor de aplicaciones, etc.)
- Otras librerías de especial interés.

3.1.1.1.2 Presentación

- Tipo de Cliente (Ligero, pesado)
- Framework (JSF, …)
- Librerías de implementación (Apache Tomahawk, Icefaces, etc.)
- Otras librerías de especial interés.

3.1.1.1.3 Negocio

- Framework (EJB, Spring)
- Otras librerías de especial interés.

3.1.1.1.4 Persistencia

- Mecanismos de persistencia (JPA, JDBC, DAAS, HOST, …)

- Otras librerías de especial interés.

3.1.1.1.5 Adaptadores de Salida

- Framework utilizado para la generación del cliente de servicio.
- Aspectos de interés desde el punto de vista de integración: transformaciones, orquestaciones de servicio,…
- Otras librerías de especial interés.

3.1.1.2 Patrones de Diseño

Descripción de aquellos patrones de diseño utilizados y que sean de especial interés desde el punto de vista del arquitecto de la aplicación.

3.1.1.3 Excepciones

Se debe de incluir el catálogo de excepciones donde se recoge el conjunto de errores relacionados con el diseño de la arquitectura del sistema.

Las excepciones se utilizan como mecanismo de comunicación de errores entre las diferentes capas de la arquitectura, se debe describir la política en el manejo de excepciones diseñada en la aplicación, indicando la responsabilidad de cada capa en aspectos como la propagación y encapsulado de las excepciones.

Si la jerarquía de excepciones diseñada es suficientemente compleja, es conveniente incluir un diagrama de jerarquía de excepciones.

3.1.1.4 Requisitos no funcionales

Se deben de incluir aspectos de diseños destacados e interesantes desde el punto de vista de Diseño de Arquitectura.

3.1.1.5 Componentes / Servicios Comunes

Se deben de enumerar los componentes / Servicio Comunes utilizados por la aplicación así como aspectos destacables en su integración desde el punto de vista arquitectónico.

3.1.1.6 Integración con DAAS

Se debe indicar si la aplicación se integra con el servicio corporativo de acceso a Datos (DAAS) así como aspectos destacables en su integración desde el punto de vista arquitectónico.

3.1.1.7 Integración con Datapower

Se debe indicar los servicios utilizados por el ESB corporativo:

Servicio	Uso obligado	Utilizado
Autenticación de usuarios en aplicaciones web (seguridad declarativa)	SI	S/N
Autenticación de usuarios en servicios web (Si se exponen servicios)	SI	S/N
Cifrado / Descifrado	NO	S/N
Autorización cuando se usa firma electrónica para autenticar	NO	S/N
Validación de firma electrónica	NO	S/N
Firma con el certificado de la DGT	NO	S/N
Proxy de servicio web (Si se expones servicios)	SI	S/N
Protección contra intrusiones	NO	S/N
Definición de políticas de nivel de servicio	NO	S/N
Transformación de protocolos	NO	S/N
Intercambios de información	NO	S/N
Transformación de mensajes XML	NO	S/N
Encaminamiento en función del contenido del mensaje	NO	S/N

3.1.1.8 Diagrama de componentes

El objetivo del diagrama de componentes es reflejar la arquitectura modular del proyecto (módulos que lo componen y dependencias entre ellos) para todas y cada una de las unidades que se despliegan del mismo (normalmente se corresponden a distintos subsistemas). También debe mostrar

la distribución de las capas arquitectónicas en los distintos módulos de construcción del proyecto, así como las interfaces requeridas y proporcionadas en cada uno de dichos módulos.

3.1.1.9 Mapa de Arquitectura

El objetivo de este diagrama es reflejar las unidades que se despliegan y que conforman la aplicación (normalmente se corresponden a distintos subsistemas) y cómo se distribuyen entre los distintos nodos que conforman la arquitectura física. Además mostrará la relación de cada unidad de despliegue con los componentes comunes y servicios externos que está utilizando.

3.2 Arquitectura común DGT

Se definirán las siguientes capas: Presentación (o adaptador de presentación), Servicios (adaptadores de integración y puertos de servicio), Negocio, Adaptador de Salida y Persistencia. Además existirán capas transversales: Seguridad, Componentes Comunes y Dominio. Si bien las capas de presentación y persistencia, en el patrón de arquitectura hexagonal, se pueden considerar también adaptadores de integración, se prefiere enumerarlas como tales para mejor comprensión.

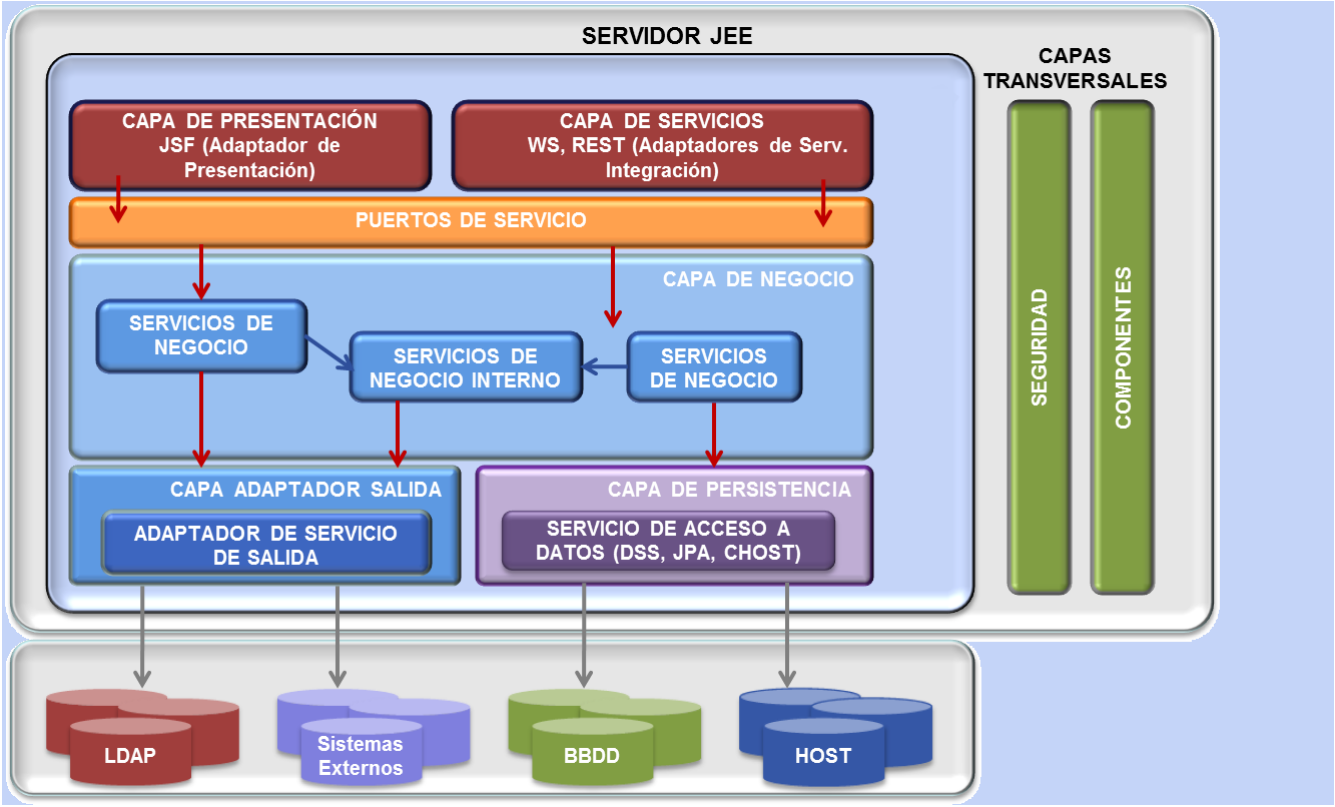

Ilustración 2. Arquitectura común DGT

3.2.1 Presentación

Basada en el estándar JavaServer Faces (JSF). JSF ofrece un modelo de componentes de interfaz de usuario, un modelo de manejo de eventos, un modelo de validación y un modelo de traducción a lenguajes de marcas, que, entre otras cosas, permite generar un mismo componente de interfaz de usuario de diversas maneras (característica que lo diferencia de otros frameworks de presentación). Permite la creación de componentes de interfaz de usuario complejos (menús, árboles, calendarios, etc.). Como implementación de JSF se usará la disponible en el servidor de aplicaciones.

Como librerías de componentes, pueden utilizarse las siguientes librerías, cuya compatibilidad con la implementación elegida ha sido testada: Apache Tomahawk, Icefaces y Richfaces. No se descartan componentes interesantes que puedan ofrecer otras implementaciones si son compatibles con la implementación elegida. Los estilos y la identidad corporativa serán aportados por la DGT.

Esta capa debe ser lo más fina posible, limitándose a invocar al puerto de servicio y/o a las interfaces de negocio de servicios de dominio (en la capa de negocio) y mostrar los resultados.

3.2.1.1 Desarrollo de aplicaciones WEB

Como norma general la estrategia de desarrollo de aplicaciones web para dispositivos móviles se basará en un diseño de interfaz adaptativo (Responsive Design) a los diferentes dispositivos que sean destinatarios de la aplicación y no en desarrollos paralelos de interfaces de usuario en función de la tipología del dispositivo (desktop, mobile, etc.).

Dependiendo de las necesidades del proyecto, es posible que el enfoque de diseño adaptativo con un único desarrollo no sea el adecuado, en cuyo caso el equipo de desarrollo deberá justificar al Dpto. de Arquitectura su NO uso para que así figure en la Hoja de Acuerdos del Proyecto.

3.2.1.1.1 Responsive Design

Responsive Design es una técnica de diseño web que consiste en crear una estructura de una página web que en función del tamaño de la o ventana en la que se visualice, variará su contenido para que siempre sea visible y usable tanto desde PC, tablets y smartphones.

Este tipo de diseño es el que deberán de realizar los equipos de desarrollo para cubrir el requisito de movilidad de las aplicaciones web.

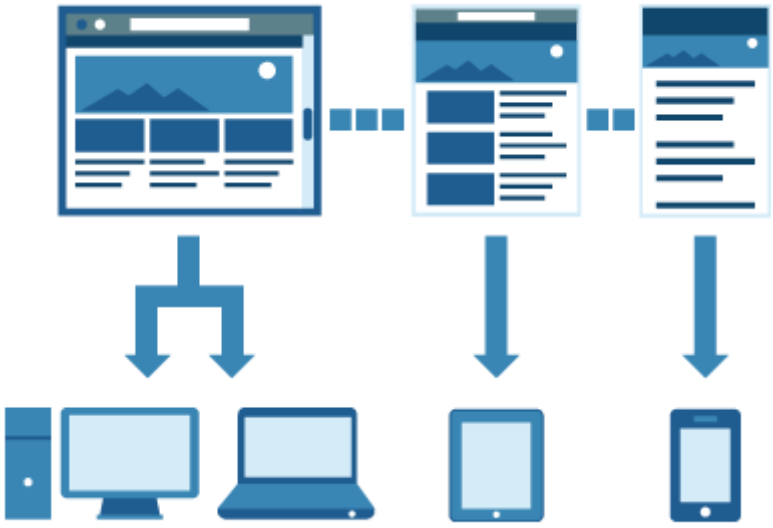

Ilustración 3. Ejemplo de adaptación de distribución de contenidos

3.2.2 Adaptadores de Servicios (Capa de servicios)

Basada en Servicios Web (SOAP/REST), en esta capa figurarán los servicios que se exponen al resto de aplicaciones, así como a actores externos a la DGT. Todo servicio que ofrezca una interfaz remota debe formar parte de esta capa y figurar en el catálogo de servicios publicados. Se ofrecerá al menos una implementación basada en SOAP/REST y opcionalmente se puede proporcionar otra implementación basada en EJB3 (sólo permitida para código legado que tenga clientes activos haciendo uso de dicha interfaz). Para las implementaciones SOAP, deben diseñarse interfaces WSDL de forma que cumplan con el estándar WS-I.

Para nuevos proyectos se dará prioridad a la implementación de servicios REST sobre SOAP, tecnología impulsada desde la DGT.

La interacción entre la capa de presentación desplegada en Internet y el negocio desplegado en Intranet se realizará a través de invocaciones a esta capa.

Asimismo, se construirán adaptadores para las invocaciones a servicios externos a la unidad de despliegue (quedan incluidos por lo tanto como tales los servicios ofrecidos por otras unidades de despliegue) de forma que el negocio quede desacoplado de los artefactos tecnológicos concretos que sea necesario utilizar para realizar la invocación a dichos servicios. Esto es extensible a las utilidades que la aplicación necesite utilizar (como por ejemplo mecanismos de caché, y en general extensiones del servidor de aplicaciones o uso de frameworks fuera del estándar JEE) de forma que la aplicación no quede acoplada a los servicios de infraestructura o a implementaciones concretas.

El aislamiento del negocio tanto en la entrada como en la salida facilitará las pruebas y mejorará la gestión de las dependencias de otros servicios que necesite y se estén desarrollando en paralelo.

La forma de construcción de las capa de servicios de negocio y datos, los posibles escenarios que se pueden dar a la hora de integrar aplicaciones ubicadas en la zona de internet con la intranet, así como la forma en la que se deben construir los adaptadores de servicios están descritos en la especificación ***DGTE-001: Desarrollo de servicios de componentes de negocio y de acceso a datos para las aplicaciones Java EE***.

Cada Área de negocio de la DGT será responsable del desarrollo de los servicios de negocio que le correspondan. Aquellos servicios cuya trivialidad no aconseje la exposición de una interfaz

remota se podrán distribuir como componentes, previa consulta y acuerdo con el Departamento de Arquitectura (*dep.arquitectura@dgt.es*).

3.2.3 Negocio

Se opta por una solución basada en servicios, manteniendo aislada la capa de persistencia, a la cual se accederá a través de invocaciones a servicios de acceso a datos. Los servicios de negocio serán EJB3 con interfaz local (consultar especificaciones de servicios de negocio y acceso a datos). Deberán usarse los mecanismos de inyección de dependencias aportados por el estándar JEE para resolver las dependencias, complementando con Spring el resto de dependencias. Spring es un framework de amplio espectro, del cual se utilizarán principalmente el contenedor de IoC (Inversion of Control) para desacoplar las dependencias entre clases y entre capas y el contenedor de AOP (Aspect Oriented Programming) complementado con el uso de AspectJ cuando sea necesario, para gestionar todos los aspectos que traspasan objetos y capas (transacciones, trazas, auditoría, etc.). Se pondrá muy especial cuidado de utilizar Spring dentro de los contenedores J2EE de manera que no interfiera con el servidor de aplicaciones, de la forma en que se define más adelante en esta guía; en concreto, no se usará para sustituir los servicios más robustos de alto rendimiento del servidor de aplicaciones (seguridad, transacciones, pool de conexiones, temporizadores, motor de servicios web, recursos manejados por el contenedor, etc.), sino que a través de Spring se delegará en el servidor de aplicaciones dicha funcionalidad. Tampoco se deberá usar Spring para las inyecciones de dependencia que puedan realizarse con los mecanismos ofrecidos por el contenedor J2EE. De este modo aprovechamos las interesantes aportaciones que ofrece sin el riesgo que supone la interferencia o la sustitución de los qualities of service del contenedor.

La interacción entre entornos se detalla en la especificación ***DGTE-001: Desarrollo de servicios de componentes de negocio y de acceso a datos para las aplicaciones Java EE***. Dicha especificación identifica los posibles escenarios.

El diseño de arquitectura de los servicios de negocio se basará en principios de diseño de arquitectura orientada a servicios, con especial énfasis en aquellos casos en que haya de ofrecerse un alto rendimiento y/o abordar transacciones de larga duración, en cuyo caso deberá diseñarse basándose en patrones de procesamiento asíncrono, de cara a asegurar la escalabilidad manteniendo unos tiempos de respuesta dentro de los márgenes establecidos por la Oficina de Pruebas. Aparte de

cualquier otra estrategia que se pueda adoptar, a modo de referencia se sugiere el uso de las siguientes estrategias (o una combinación de ellas):

- Invocación desacoplada. Se dará respuesta inmediata a la petición de servicio sin esperar a realizar el procesamiento necesario. Podrá informarse al actor invocador del resultado de la petición bien a través de mecanismos de polling bien a través de mecanismos de callback (se recomienda implementar ambos para mayor flexibilidad).
- Ejecución en paralelo. Cuando sea de aplicación, podrán lanzarse procesos simultáneos para resolver partes del procesamiento que no dependan de la ejecución previa de otros procesamientos. Esto podrá hacerse básicamente de dos formas:
    - Dividiendo el problema en sub-tareas que podrán ser acometidas en hilos paralelos solicitados al contenedor, los cuales pueden devolver un resultado de procesamiento (futuro) o no.
    - Poniendo una cola entre cada sub-tarea
- One-way (Assured Delivery). El invocador podrá desentenderse del resultado del procesamiento si éste no es vinculante para continuar con su propio procesamiento y/o se tiene una garantía por parte del servicio invocado de la ejecución satisfactoria de la funcionalidad solicitada. Puede implementarse por ejemplo a través de JMS o WS-ReliableMessaging.

Por parte de cada equipo de desarrollo deberá determinarse la relación coste-beneficio de la implementación de estos diseños, pero en cualquier caso la solución adoptada deberá garantizar que los requisitos de rendimiento no caen fuera del rango aceptado.

3.2.4 Persistencia

Estará basada en la plataforma de servicios de acceso a datos. En una arquitectura orientada a servicios, los servicios de acceso a datos, aparte de las ventajas generales de la arquitectura orientada a servicios, aportan lo siguiente:

- Abstracción del uso de los datos, con un bajo acoplamiento que permite la evolución independiente y sin impacto de las capas de negocio y datos.
- Centralización de las reglas que garanticen la calidad y la privacidad del dato.
- Simplificación de acceso a datos heterogéneos y/o a múltiples fuentes de datos.
- Facilitan la auditoría, la monitorización y la distribución de los datos.

La abstracción que representa la capa de servicios de acceso a datos y su bajo acoplamiento con la capa de negocio permite la evolución física de los datos sin afectar a ninguna aplicación, ya que lo cambios se centralizan en la propia capa de servicios de acceso a datos, independiente de todas las aplicaciones.

Se cuenta con una plataforma de acceso a datos que ofrece los datos a través de servicios web (REST y/o SOAP). Los servicios de acceso a datos sólo podrán ser invocados de forma directa por la aplicación responsable del modelo de datos accedido; otras aplicaciones que necesiten acceder a esos datos tendrán que hacerlo a través del correspondiente servicio de negocio que ofrezca la aplicación responsable del modelo de datos (en ausencia de dicho servicio se pondrá en disposición en el ESB una primera versión del mismo que será un proxy del servicio de acceso a datos y que posteriormente la aplicación responsable del modelo de datos evolucionará para añadir el resto del negocio). Las capacidades de la plataforma de servicios de acceso a datos incluyen lo siguiente:

- Acceso a múltiples repositorios de datos (incluyendo adaptación para el HOST DGT)
- Combinación de datos de múltiples repositorios
- Control de seguridad de acceso a los datos
- Servicios de consulta y de escritura
- Control transaccional
- Disparo de eventos, útil para la sincronización de datos
- Caché incorporada
- Soporte a los estándares WS (WS-Reliable Messaging, WS-Policy, MTOM, WS-Security...)
- Políticas de Quality Of Service (control de accesos a nivel global, de servicio y de operación)
- Alto rendimiento, escalabilidad y disponibilidad
- Implementación muy ágil de los servicios basada en descriptores XML

El uso de los servicios de acceso a datos será obligatorio para toda nueva aplicación y para la construcción de nuevos accesos a datos en aplicaciones ya existentes. El uso de servicios de acceso a datos proporcionados por la plataforma corporativa constituyen la base para la migración de los datos de sistemas basados en HOST a bases de datos relacionales basadas en sistemas abiertos.

Las aplicaciones que tengan que implementar la capa de persistencia a través del uso de la plataforma de servicios de acceso a datos se limitarán a construir, como implementación de su capa de persistencia, un cliente de servicio hacia los servicios que necesiten acceder. Para la implementación

de los servicios por parte del equipo de desarrollo de la plataforma de servicios de acceso a datos será necesario que cada aplicación informe de los requisitos de acceso a datos que tenga, poniéndose en contacto con el "Departamento de Arquitectura DAAS" a través de la Herramienta de Gestión de Servicios TI de la DGT.

Aquellas aplicaciones que sean evolutivos y ya tengan unan capa de persistencia basada en JPA, podrán seguir con dicha estrategia pero deberán planificar el cambio de implementación de su capa de persistencia en cuanto sea posible. JPA es la base de la persistencia manejada por contenedor en Java EE 5 y posteriores. JPA supone una estandarización de la persistencia y el mapeo objeto-relacional, aportando un modelo común para:

- Versionado de objetos para control de concurrencia.
- Generación de claves de base de datos.
- Carga perezosa.
- Herencia.

JPA permite realizar el mapeo objeto-relacional sin tener que implementarlo directamente en JDBC, a la vez que nos desacopla de la base de datos subyacente. Es interesante poder seguir teniendo acceso a la capa inmediatamente inferior para poder tener acceso a la funcionalidad que no aporte JPA, además de poder usar directamente SQL si se considera necesario por cuestiones de funcionalidad o rendimiento. A través de los mecanismos establecidos por la especificación JEE, se obtendrá la versión proporcionada por el servidor, que es la 2.0, la cual es compatible con el código de las versiones 1.x salvo casos muy puntuales que pueden consultarse en

http://pic.dhe.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=%2Fcom.ibm.websphere.jpafep.multiplatform.doc%2Finfo%2Fae%2Fae%2Frejb_jpa2mig.html.

Es obligatorio programar para la versión 2.0 para todas las aplicaciones nuevas, pudiendo los evolutivos mantener el código existente de las versiones 1.x a través del mecanismo de compatibilidad (mantener un fichero persistence.xml de la versión 1.x); para más información pueden consultarse los siguientes documentos:

- http://pic.dhe.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.jpafep.multiplatform.doc/info/ae/ae/tjpa2_ep.html
- \\informatica-cpd\Calidad\V02_v01_R003\doc\documentacionadicional\Arquitectura\JPA-OSGI-FeaturePack.pdf

La construcción de las capas de persistencia, que se basará en la construcción de servicios de acceso a datos, se detalla en la especificación **DGTE-001: Desarrollo de servicios de componentes de negocio y de acceso a datos para las aplicaciones Java EE**.

3.2.5 Capas transversales

La seguridad estará basada en la seguridad declarativa de JEE. No se deberá implementar ningún tipo de seguridad programática dentro de las aplicaciones, ni siquiera a través de frameworks existentes. Los frontales Datapower se encargarán de la autenticación de los usuarios, tanto en los casos en que se realice mediante claves usuario/password (autenticación contra directorio LDAP corporativo), certificado digital (autenticación contra @firma) como con la plataforma . Todo módulo web (aplicación o servicio) deberá indicar la protección de acceso en los descriptores de despliegue y usar las páginas de login que facilite la DGT. Las aplicaciones que tengan que pasar a través de Datapower (todas las nuevas y los evolutivos en que así se determine), deberán configurar el mecanismo de autenticación como "BASIC" en su descriptor de despliegue. La autorización se hará mediante la definición de roles, quedando expresamente prohibida la creación de repositorios de usuarios propios de una aplicación, así como de tablas de asignación de permisos. La especificación completa se puede consultar en **DGTE-002: Desarrollo de aplicaciones seguras en la plataforma Java™ EE. Modelo declarativo de seguridad**. Cualquier necesidad en cuanto a autenticación y autorización de acceso a los recursos que no esté cubierta por los repositorios de usuarios actuales debe consultarse con el Departamento de Arquitectura a través de la Herramienta de Gestión de Servicios TI de DGT.

Existirán distintos dominios en función de los requisitos de seguridad establecidos para acceder a las aplicaciones (Cl@ve, certificado digital o usuario/password), debiendo desplegarse las aplicaciones en el dominio que les corresponda según el tipo de credencial que sea necesario aportar para acceder a ellas.

Todo servicio que tenga una interfaz remota debe estar protegido contra accesos no autorizados, en línea con lo establecido en el Esquema Nacional de Seguridad y en la Política de Firma de la DGT. Sólo en aquellos casos en que la información a que se accede o la operación que se ejecuta no sea susceptible de cumplir por lo establecido en la legislación vigente en materia de seguridad y protección de datos se admitirán excepciones a esta norma, para lo cual será necesaria la

solicitud pertinente por parte del Jefe de Área correspondiente de la Dirección General de Tráfico al Comité de Seguridad, quien examinará la justificación y dictará resolución en consecuencia.

Existirá además una capa transversal de componentes que cubran funcionalidad común a cualquier aplicación. En cualquier aplicación se deberán usar obligatoriamente los componentes comunes que la DGT proporciona. De igual forma, cualquier funcionalidad que los equipos de desarrollo consideren candidata a ser reutilizada de forma general en más de un proyecto, deberá ser puesta en conocimiento de la DGT para que se evalúe y se desarrolle un componente común o servicio si se considera oportuno. Toda funcionalidad que sea susceptible de ser compartida entre capas y sea propia del negocio de cada aplicación debe ser aislada en un componente y pasar a formar parte de esta capa transversal.

Todos estos componentes se encuentran disponibles junto con su documentación en el siguiente directorio \\dgt.red\sc\Gi\Calidad\SOA.

Se recomienda la adaptación de los desarrollos para utilizar las últimas versiones disponibles de los componentes comunes, conforme vayan siendo facilitadas. Además pueden estarse desarrollando nuevos componentes comunes, tanto nuevas versiones de los existentes, como otros componentes nuevos. Estas dependencias deberán ser gestionadas adecuadamente a lo largo del desarrollo de cada proyecto.

3.2.6 Adaptador Salida

Se construirán adaptadores para las invocaciones a servicios externos a la unidad de despliegue de forma que el negocio quede desacoplado de los artefactos tecnológicos concretos que sea necesario utilizar para realizar la invocación a dichos servicios. Esto es extensible a las utilidades que la aplicación necesite utilizar (como por ejemplo mecanismos de caché, y en general extensiones del servidor de aplicaciones o uso de frameworks fuera del estándar JEE) de forma que la aplicación no quede acoplada a los servicios de infraestructura o a implementaciones concretas.

El aislamiento del negocio tanto en la entrada como en la salida facilitará las pruebas y mejorará la gestión de las dependencias de otros servicios que necesite y se estén desarrollando en paralelo.

3.3 Datapower

Datapower es un appliance que se encarga de realizar las labores de autenticación, centralizando la seguridad y el control de acceso. Toda nueva aplicación debe utilizar seguridad declarativa según lo marcado por la especificación **DGTE-002: Desarrollo de aplicaciones seguras en la plataforma Java™ EE. Modelo declarativo de seguridad**.

Aparte de los cometidos de seguridad, funciona como un firewall XML y como un ESB. Las funcionalidades más importantes que ofrece son las siguientes:

- Habilitación de servicios virtuales (enrutamiento dinámico y estático, incluyendo búsqueda en registros UDDI) que permite arquitecturas Hub and Spoke.
- Integración de aplicaciones (intercambios de información y transformación de mensajes)
- Validación de certificados
- Protección contra intrusiones (asignación de accesos máximos por servicio o unidad de tiempo, balanceo de recursos…)
- Intermediación (conversión de protocolos)
- Aceleración de XSL
- Web Services Security
- Web Services Proxy
- Cifrado y descifrado (incluyendo cookies)
- Protección contra inyección SQL
- Validación y procesado de documentos XML

La delegación de la seguridad en Datapower es obligatoria para toda aplicación, así como la intermediación de servicios (tanto en la entrada como en la salida). Cualquier aplicación que potencialmente pueda necesitar del resto de la funcionalidad descrita debe comunicarlo al equipo de implantación de Datapower a través de la Herramienta de Gestión de Servicios TI o excepcionalmente en el buzón de Arquitectura Datapower *arquitectura.datapower@dgt.es*).

3.4 Integración de los frameworks de código abierto con el servidor J2EE

3.4.1 Spring

El uso de Spring debe quedar subordinado a aquellas situaciones donde la funcionalidad no esté proporcionada por el contenedor J2EE.

3.4.1.1 Acceso a recursos

Cuando se necesite acceder un recurso, se hará referenciando los recursos manejados por el contenedor:

```
<jee:jndi-lookup id="dataSource" jndi-name="jdbc/springdb" cache="true" resource-ref="true"
lookupOnStartup="false" proxy-interface="javax.sql.DataSource"/>

<jee:local-slsb id="ejbReference" jndi-name=""ejb/EjemploBean" business-
interface="es.trafico.ejemplo.Ejemplo" cache-home="true" lookup-home-on-startup="true" resource-
ref="true"/>
```

De este modo, el recurso se localizará usando la referencia configurada y por tanto manejada por el servidor de aplicaciones. Previamente se habrá declarado el uso del recurso en el fichero de despliegue del módulo correspondiente (notar que la referencia declarada en el descriptor de despliegue del módulo debe coincidir con el valor del atributo "value" del elemento <property name="*jndiName*">:

```
<resource-ref>
        <res-ref-name>jdbc/springdb</res-ref-name>
        <res-type>javax.sql.DataSource</res-type>
        <res-auth>Container</res-auth>
        <res-sharing-scope>Shareable</res-sharing-scope>
</resource-ref>

<ejb-local-ref>
        <ejb-local-ref id="EjbRef_2_1">
    <ejb-ref-name>ejb/EjemploBean</ejb-ref-name>
    <ejb-ref-type>Session</ejb-ref-type>
    <local-home>es.trafico.ejemplo.EjemploLocalHome</local-home>
    <local>es.trafico.ejemplo.EjemploLocal</local>
</ejb-local-ref>
```

3.4.1.2 Uso de conexiones nativas JDBC

Spring tiene un mecanismo para acceder a conexiones nativas cuando varias operaciones JDBC requieren interacción con el recurso nativo JDBC. El uso de implementaciones de *NativeJdbcExtractor* no está soportado por WebSphere y su uso por lo tanto no está permitido.

3.4.1.3 Uso de transacciones

WebSphere no expone la interfaz *javax.transaction.TransactionManager* a las aplicaciones o *frameworks* desplegados en el servidor. Se deberá usar el modelo de transacciones declarativo de Spring para asegurar que un contexto de transacción global o local controlada por el servidor esté siempre disponible cuando se accede a un recurso; esta es la única forma permitida de usar transacciones en Spring, a través de la referencia a *WebSphereUowTransactionManager* que implementa Spring para sacar partido de la interfaz *UOWManager* de WebSphere. Se usará la siguiente declaración:

```
<bean id="transactionManager"
 class="org.springframework.transaction.jta.WebSphereUowTransactionManager"/>
```

Un *bean* de Spring que referencie la anterior declaración usaría inyección de dependencias para usar el soporte transaccional de la siguiente manera:

```
<bean id="someBean" class="some.class">
        <property name="transactionManager">
                <ref bean="transactionManager"/>
        </property>
</bean>
```

No se permite pues la forma de acceso de versiones de Spring anteriores a la 2.1, ya que ha dejado de estar soportada en WebSphere.

3.4.1.4 Uso de JMS

Al igual que al acceder a otros recursos, habrá que asegurarse de que se usan los proveedores de recursos de JMS manejados por WebSphere. Habrá que seguir el mismo patrón que se explicó en el apartado 3.4.1.1 para acceder a un recurso JDBC.

3.4.1.5 Uso de OpenJPA (instrucciones para su uso en Websphere 6.1)

Se deberán usar las clases de OpenJPA directamente en lugar de los *helpers* de Spring (los ubicados en el paquete *org.springframework.orm.jpa*). Se usará el modelo de transacciones

declarativo de Spring según se explicó en el apartado 3.4.1.3. Por ejemplo, un fichero *persistence.xml* debería quedar tal como así:

```
<persistence
  xmlns="http://java.sun.com/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
  http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd" version="1.0">
        <persistence-unit name="default" transaction-type="JTA">
        <provider> org.apache.openjpa.persistence.PersistenceProviderImpl </provider>
        <jta-data-source> java:comp/env/jdbc/springdb </jta-data-source>
        <properties>
                <property name="openjpa.TransactionMode" value="managed" />
                <property name="openjpa.ConnectionFactoryMode"value="managed" />
                <property name="openjpa.jdbc.DBDictionary" value="db2" />
        </properties>
        </persistence-unit>
</persistence>
```

En WAS 7.0 deberá usarse JPA de la forma definida en la especificación EJB 3.0 (JSR-220). Se usarán los mecanismos definidos en la especificación para inyectar las referencias a los *EntityManagerFactory* y *EntityManager*, las cuales se obtendrán del contexto del servidor de aplicaciones.

3.4.1.6 Uso de JMX y MBeans

Spring JMX MBeans están soportados sólo cuando son registrados a través de *MBeanServer* de WebSphere. No se debe usar *MBeanServerFactory* para instanciar un *MBeanServer* e inyectar en él el *MBeanExporter*. Sí puede usarse *WebSphereMBeanServerFactoryBean* (ya que obtiene el *MBeanServer* de Websphere), aunque su uso es opcional ya que *MBeanExporter* es capaz de detectar el *MBeanServer* cuando se ejecuta dentro de un servidor de aplicaciones. No está soportado por WebSphere el uso de *ConnectorServerFactoryMBean* o *JMXConnectorServer* de Spring para exponer a los clientes a través de puertos de entrada JMX.

Los *MBeans* de WebSphere son identificados a través de un *javax.management.ObjectName* cuando son instalados. Cuando son desinstalados, necesitan ser localizados con el mismo nombre completamente cualificado en vez de simplemente con el nombre de propiedad *MBean*. Lo mejor es implementar *org.springframework,jmx.export.naming.ObjectNamingStrategy*, que es una interfaz que encapsula la creación de una instancia de *ObjectName* y es usada por el *MBeanExporter* para obtener *ObjectNames* cuando se instalan *beans*. Se puede añadir la instancia de *ObjectNamingStrategy* al *bean*

cuando se instala: esto asegurará que el *MBean* es desinstalado de forma correcta al desinstalar la aplicación.

3.4.1.7 Uso de Planificadores, Temporizadores e Hilos.

Spring provee varias clases *TaskExecutor*, pero la única que soporta WebSphere para la ejecución de tareas asíncronas es *WorkManagerTaskExecutor*, que utiliza correctamente las agrupaciones de hilos manejadas por el servidor.

Se deben evitar escenarios en los que se creen hilos descontrolados y desconocidos para el servidor, como por ejemplo el uso del método *registerShutdownHook* en *AbstractApplicationContext* o sus subclases, lo cual puede crear hilos de ese tipo. Para cerrar explícitamente un *ApplicationContext* se deben usar los eventos de ciclo de vida recibidos del contenedor correspondiente.

Por último, Spring provee o se integra con cierto número de paquetes de planificación, pero el único soportado por WebSphere es el *CommonJ WorkManager* (el único que funciona con hilos controlados por el servidor). Las tareas planificadas a desplegar en el servidor de aplicaciones deberán planificarse usando el temporizador del servidor de aplicaciones, quedando expresamente prohibido en uso de planificadores externos (tales como *Quartz*).

3.4.1.8 Versión de Spring

En Websphere 7 debe usarse una versión de Spring superior o igual a la 2.5.5 debido a incompatibilidades de las versiones anteriores con la JDK de IBM.

3.5 Reglas de Diseño

A continuación se enumeran una serie de reglas que serán auditadas para asegurar su completo cumplimiento.

3.5.1 Reglas generales

Usar patrones de diseño. Los patrones de diseño ofrecen soluciones probadas a problemas conocidos.

Usar preferentemente la composición de objetos en vez de la herencia. Resulta mucho más flexible implementar interfaces que extender clases. La herencia debe considerarse tanto a nivel de diseño como de implementación, usándose únicamente cuando exista una clara parte del dominio del negocio compartido por las clases que heredan. El patrón *Template* es un buen ejemplo de uso de herencia, siempre y cuando aporte alguna ventaja sobre el uso del patrón *Strategy*.

Usar el patrón *Observer*. Cuando nos encontremos con un flujo importante de operaciones que puedan dar lugar a múltiples eventos, dichos eventos pueden ser publicados para ser manejados de forma independiente, consiguiendo de este modo una separación de aspectos. Puede conseguirse lo mismo aplicando técnicas de programación orientada a aspectos. Sólo los flujos de operaciones grandes deben usar esta técnica. En los contenedores EJB, el uso de JMS proporciona esta funcionalidad.

Excepciones informativas. Diferentes problemas deben dar lugar a diferentes excepciones; no debemos limitarnos a suministrar un mensaje de error diferente. Deben incluirse códigos de error en las excepciones, y distinguir entre mensajes destinados al usuario (los cuales son susceptibles de ser usados para presentación en pantalla) y mensajes destinados al programador (los cuales son susceptibles de ser usados para *logging*); estos últimos deben contener el código de error, el cual deberá figurar en el diccionario de errores. Se debe limitar el número de excepciones lanzadas por cada método, ya que un número elevado de excepciones no aporta más información a la clase que usa el servicio, únicamente supone ruido.

Generación de trazas. Para la generación de trazas deberá usarse el componente común correspondiente. En el manual de usuario figura toda la información sobre el uso del mismo, así como los niveles de traza definidos (respetar escrupulosamente en lo que concierne a qué nivel de traza utilizar en cada caso concreto). Cabe reseñar aquí que debe utilizarse como *layout* el definido en el componente común (*es.trafico.framework.trazas.log4j.DGTLayout*), lo cual asegurará que las trazas resultantes puedan ser monitorizadas por los sistemas automáticos de monitorización (ver manual de usuario para consultar ejemplos); asimismo es necesario indicar, aparte del mensaje de error, su código de error según el catálogo de excepciones (lo cual se fuerza a través de la interfaz del componente, que lo incluye como parámetro).

***Endpoints* de Servicio**. Deberán configurarse los *endpoint* de los servicios en el servidor de aplicaciones, en vez de hacerlo en ficheros *.properties*. Para ello se declararán recursos URL en los

descriptores de despliegue, y el Departamento de Sistemas se encargará de configurar en el servidor el *endpoint* físico de cada servicio. Se hará de esta manera hasta que se ponga en marcha un registro de servicios el cual dará pleno soporte a los servicios virtuales.

Configuración de almacenes de certificados. Cualquier aplicación que necesite manejar certificados para firma digital de documentos XML lo hará a través de la incorporación de dichos certificados a los almacenes de claves del servidor de aplicaciones o bien delegando la firma en Datapower o en el servicio de firma digital. Cualquier otra necesidad de firma digital que no pueda satisfacerse por estos medios debe consultarse con el Departamento de Arquitectura (dep.arquitectura@dgt.es).

3.5.2 Reglas de referencia a recursos JNDI

Cuando se utilicen recursos disponibles a través de JNDI se respetarán las recomendaciones de la especificación J2EE en cuanto a su nombrado. Para recursos no previstos en la especificación, la DGT ha establecido un estándar de nombrado, el cual se incluye en la siguiente tabla junto con el recordatorio de lo recomendado en la especificación.

No se permiten la utilización directa de los recursos físicos definidos en el servidor. Dichos recursos tienen que ser enlazables en la aplicación a través del descriptor de despliegue.

Aparte, en aplicaciones J2EE no deben configurarse las referencias a dichos recursos en ficheros de configuración, ya que supone una doble abstracción que no aporta nada a la abstracción provista por la propia especificación J2EE (el descriptor de despliegue).

TIPO DE RECURSO	CONTEXTO BASE	SUBCONTEXTO
Conexiones a bases de datos relaciones (*Datasource*)	java:comp/env	jdbc
Conexiones a sistemas de información de empresa a través de JCA	java:comp/env	eis
URL	java:comp/env	url
Colas JMS	java:comp/env	jms

TIPO DE RECURSO	CONTEXTO BASE	SUBCONTEXTO
Proveedores de entorno de recursos	java:comp	env
Temporizadores	java:comp/env	servicios/tm
Caché dinámica (ante-memoria)	java:comp/env	servicios/cache
Gestores de trabajo	java:comp/env	servicios/wm
Servidores de correo	java:comp/env	mail
EJB	java:comp/env	ejb
Transacciones	java:comp	UserTransaction
ORB	java:comp	ORB
Contextos de persistencia	java:comp/env	persistence

Tabla 1. Nombrado de recursos JNDI

3.5.3 Reglas para despliegue en clúster

La arquitectura física de los servidores de aplicaciones está basada en clúster, lo cual impone ciertas reglas para evitar la falta de sincronización entre los distintos nodos del clúster:

- No usar variables estáticas para guardar información de estado
- Usar la caché dinámica de Websphere para compartir objetos entre nodos
- No cachear los manejadores EJBHome, ya que ello provocaría que siempre se obtuviesen instancias del mismo servidor, perdiéndose así el balanceo de carga y provocando un posible punto de fallo ante la caída de ese nodo concreto.

3.5.4 Reglas en la capa de presentación

Inhabilitación de la ejecución duplicada de formularios. Deberá impedirse que un usuario pueda ejecutar un formulario repetidamente sin haber concluido la ejecución anterior (algo frecuente en interfaces de usuario basadas en navegador), a no ser que dicha ejecución duplicada no tenga efectos nocivos (Consultas). Se recomienda utilizar soluciones basadas en el *token* sincronizador

(establecer un valor en la sesión de usuario e incluirlo como campo oculto en el formulario, de forma que se envíe al servidor en cada ejecución del formulario para poder comparar con el existente en la sesión, debiendo coincidir para que se permita la ejecución). Se recomienda asimismo complementar lo anterior con un control mediante *Javascript* de la doble pulsación del botón de envío del formulario.

Validación de los campos de formulario basada en servidor. Se recomienda no realizar únicamente la validación de campos de formulario en cliente, sino también en servidor, ya que el cliente puede desactivar la ejecución de lenguajes de *script* en su máquina.

Simplificación de las vistas. Las vistas deben usarse únicamente para formateo de los datos a mostrar, delegando cualquier lógica de programación en el controlador y en clases auxiliares (implementadas como *JavaBeans* o etiquetas personalizadas). Es altamente recomendable que las páginas JSP usadas como vistas no contengan código Java embebido, desplazando el mismo a etiquetas personalizadas.

Paso de estructuras de datos de la capa de presentación a la capa de negocio. Se prohíbe el paso de estructuras de datos exclusivas de la capa de presentación (como por ejemplo *HttpServletRequest*) a la capa de negocio, ya que ello incrementa el acoplamiento y destruye la independencia de las capas entre sí.

Abuso de la información de sesión. Debe evitarse el uso de sesiones *http* en la medida de lo posible, y en caso de usarse guardar ahí sólo lo imprescindible cuando no se pueda resolver de otro modo.

Invocación directa de las vistas. Debe evitarse que los usuarios puedan invocar directamente las vistas en aplicaciones con interfaz web.

Evitar redirección. En la medida de lo posible, debe evitarse que el direccionamiento del controlador a las vistas se defina mediante el mecanismo de "*sendRedirect*".

Aumento del rendimiento de páginas web. Deben tenerse en cuenta, cuando sean de aplicación, las recomendaciones de *Akamai* para el incremento de rendimiento de las páginas web.

3.5.5 Reglas en la capa de servicios

3.5.5.1 Adaptadores SOAP

Se cumplirá con lo especificado en **DGTE-001: Desarrollo de servicios de componentes de negocio y de acceso a datos para las aplicaciones Java EE**.

3.5.5.2 Adaptadores REST

Se cumplirá con lo especificado en la especificación **DGTE-008: Diseño de API**.

3.5.6 Reglas en la capa de negocio

Se cumplirá con lo especificado en **DGTE-001: Desarrollo de servicios de componentes de negocio y de acceso a datos para las aplicaciones Java EE**.

3.5.7 Reglas en la capa de persistencia

Pool de conexiones. Es obligatorio el uso del pool de conexiones provisto por el servidor de aplicaciones para obtener conexiones con los repositorios de datos. No se permiten accesos directos a repositorios de datos desde los contenedores del servidor de aplicaciones.

Bloqueos dentro de transacciones. Cuando se inicie una transacción, se terminará lo antes posible evitando procesamiento intermedio de larga duración que bloquee al resto de usuarios en espera de una conexión.

Acceso reiterado a datos que no van a cambiar. Cuando la información a consultar varía muy de cuando en cuando es preferible mantenerla cacheada en memoria.

Consultar **DGTE-001: Desarrollo de servicios de componentes de negocio y de acceso a datos para las aplicaciones Java EE**.

3.5.8 Reglas de persistencia en Host

Para acceder al host, será obligatorio utilizar el componente común de acceso a host, respetando el esquema fijado por la DGT.

3.5.9 Reglas en las capas Host

- **Generales**: La nomenclatura de los componentes de host se especifica en el "*Anexo 13 Normas de Codificacion PL1*".
- **Servicios Online:** se utilizará un diseño en 2 capas, una de NEGOCIO y otra de DATOS. La lógica de negocio de la primera capa será la mínima imprescindible, realizando en las capas javas la máxima funcionalidad posible.

Existen plantillas para cada tipo de módulo en \\Publico\Calidad\dev\Host

Se utilizará el protocolo JCA para comunicación con sistemas abiertos, según la arquitectura disponible en el "*Anexo 14 Guia JCAs HOST*":

- JCAR servicios sólo lectura.

- JCAU servicios lectura/escritura

Cada servicio se diseñará como módulo independiente y será invocado desde cada JCA como ENTRY con el siguiente paso de parámetros OBLIGATORIO:

DATOS_ENTRADA.- POR POSICIONES - CHAR(32760)VAR

DATOS_SALIDA .- FORMATO XML - CHAR(32760)VAR

LOGAIM_S .- REGISTRO PARA INSERTAR EN LOG

IND_LOG .- INDICADOR PARA ESCRIBIR EN LOG.

IND_ROLLBACK .- INDICADOR PARA HACER ROLLBACK.

Cada empresa dispondrá de una INCLUDE para añadir los servicios que necesite a cada conector. Su nomenclatura es: IJRCeee

I = include

J = JCA

R = conector read

C = conductores

eee = empresa

Y se ubican en las librerías EXPJCA.AAAA.FUENTES (AAAA representa la aplicación, por ejemplo, COND)

Existen plantillas en el entorno de desarrollo en la librería DES.CALIDAD.PL1.

No está permitido el uso de SQL dinámico salvo casos muy justificados.

Es obligatorio el uso de la función DEBUG para mostrar trazas, que deberán desactivarse en producción.

3.5.10 Excepciones

Las excepciones definidas en las aplicaciones tendrán que heredar de la jerarquía de excepciones base definida por la DGT.

Se deben reflejar todas las excepciones posibles que arroja el sistema, con un código identificador único, de forma que el usuario pueda identificarlas.

Las excepciones se describirán incluyendo, al menos, los siguientes conceptos:

- Código identificativo de la excepción (acrónimo de 4 letras + código numérico de 4 dígitos. Ejemplo: CCAL0001,etc)
- Subsistema
- Tipo y descripción de la excepción
- Condiciones previas del sistema de información
- Elemento afectado (nodo, módulo, clase)
- Respuesta del sistema de información y elemento asociado a la respuesta

Nota: La forma de empaquetar las excepciones se deja a elección de los proyectos en la parte que no definen las especificaciones de la DGT.

3.5.10.1 Normas básicas para el tratamiento de excepciones

Las normas básicas para el tratamiento de excepciones son:

- Todos los tipos de excepciones capturadas deben ser manejadas con su propia cláusula catch. [AvoidInstanceofChecksInCatchClause]
- Al disparar una excepción desde un bloque catch, pasar siempre la excepción original a la nueva para mantener el stack trace. [PreserveStackTrace]
- No usar Catch Throwable. [AvoidCatchingThrowable]
- No usar throws Exception en la declaración de método, usar una clase derivada de RuntimeException o una excepción chequeada. [SignatureDeclareThrowException]
- No usar las excepciones como control de flujo. [ExceptionAsFlowControl]

- No capturar excepciones NullPointerException, ni dispararlas. [AvoidCatchingNPE, AvoidThrowingNullPointerException]
- No disparar tipos de errores o excepciones primarias, en su lugar lanzar subclases de ellas. [AvoidThrowingRawExceptionTypes]
- No relanzar excepciones desde bloques catch. [AvoidRethrowingException]

4 Modelo de Diseño

El diseño se validará en la fase de construcción mediante el análisis con Sonarqube, el cual comprobará el cumplimiento de las reglas de diseño detalladas en este documento.

4.1 Nomenclatura de módulos

A la hora de diseñar los módulos físicos de la aplicación deberá tenerse en cuenta lo indicado en el punto 3.1 y el requisito no funcional de modularidad. En negrita se indica la parte fija y en cursiva la parte variable en el nombrado de módulos:

Módulo EAR: ear-nombre_resumen_topico

Módulos de presentación:

Elementos de presentación (css, js, etc.): web-*nombre_resumen_topico*

- Clases de presentación: pojo-pres-*nombre_resumen_topico*

Módulos de negocio: lógica de negocio de la aplicación proveedora del servicio. En estos módulos se encontrará la implementación de los Puertos de Servicios de Negocio.

- Clases de la capa de negocio en formato POJO (Plain old Java Object): pojo-*nombre_resumen_topico*

Servicios Internos de Negocio: Lógica de Negocio interno que no se expone fuera de la aplicación. No es obligatorio segregar en módulos diferentes las interfaces y las implementaciones**.** Si no se realiza la segregación indicada, se utilizará la nomenclatura destinada a las implementaciones.

Interfaces del servicio interno de Negocio: pojo-*nombre_resumen_topico*-local-interno

Implementaciones del servicio interno de Negocio: pojo-*nombre_resumen_topico*-interno

Módulos de acceso a datos: Conceptos relacionados con la persistencia de la información. No es obligatorio realizar la segregación de módulos que indica la nomenclatura**.** Si no se realiza la segregación indicada, se utilizará la nomenclatura destinada a las implementaciones.

- Interfaces locales de acceso a datos: dao-*nombre_resumen_topico*-local
- Implementaciones: dao-*nombre_resumen_topico*
- Entidades de persistencia: pojo-entidades-*nombre_resumen_topico*

Módulos de Dominio Externo o DGT: conceptos y entidades que conforman el dominio DGT de la aplicación (sólo aplica si existe integración mediante EJB Remotos cuyo uso queda restringido a código legado).

- **pojo-***nombre_resumen_topico***-dominio**

Módulos de Dominio Interno: conceptos y entidades que conforman en dominio interno (modelo) de la aplicación

- **pojo-***nombre_resumen_topico***-dominio-interno**

Módulo de utilidades: clases de utilidades de la capa transversal.

- **pojo**-*nombre_resumen_topico*-**util**

Módulos de Servicios:

- **Capa de Servicios de Integración**:
    - **Puerto de Servicio** con las interfaces locales necesarias para invocar a los Servicios de Negocio Locales**: pojo**-nombre_resumen_topico**-local.**

(Para los servicios Web desarrollados en J2EE 1.4 que no siguen la especificación dgte-001-spec-1_1_Servicios de Negocio y Acceso a Datos el puerto se denominará **pojo-servicio**-nombre_resumen_topico).

- **Adaptadores de Servicios Web**: ws-*nombre_resumen_topico*
- **Adaptadores EJB**: Este tipo de adaptadores, sólo aplica a código legado.
    - Adaptador para la publicación de servicios de negocio EJB para integración entre aplicaciones. Contiene la implementación de los “Beans” de Servicios de Negocio que la aplicación expone al exterior; **ejb**-*nombre_resumen_topico*
    - Cliente de servicio con las interfaces remotas necesarias para invocar a los Servicios de Negocio publicados: **ejb**-*nombre_resumen_topico***-client**
- **Adaptadores REST**: rest-*nombre_resumen_topico*

- **Adaptadores SCA**: adaptador-*nombre_resumen_topico*

- **Capa de Servicios de Salida**: Mecanismos de integración con servicios externos a la aplicación. No es obligatorio realizar la segregación de módulos que indica la nomenclatura**.** Si no se realiza la segregación indicada, se utilizará la nomenclatura destinada a las implementaciones. Por la escasa complejidad de sus implementaciones, basadas en el desarrollo de clientes de servicios, se permite agrupar en un único módulo todos los adaptadores de salida que usa la capa de negocio de la aplicación.
    - **Interfaces de salida**: pojo-*nombre_resumen_topico*-local-salida

      Adaptadores de Salida: pojo-*nombre_resumen_topico*-adaptador-salida

Módulos Cliente (opcionales):

- **client-[acrónimo-servicio]**-*nombre_resumen_topico*

ELEMENTO	LOCALIZACIÓN	ARCHIVOS
Construcción del EAR	dev/**ear-***NombreModulo*/src/main/java/	En este directorio se almacenarán las clases java que deban construirse junto al EAR. Normalmente no habrá ninguna.
	dev/**ear-***NombreModulo*/src/main/application/	Los ficheros de configuración del servidor y todos aquellos ficheros que tengan que ser incluidos dentro del EAR
		/WEB-INF documentos/archivos a incluir en este subdirectorio del EAR
		/META-INF el manifest.mf y otros documentos/archivos
Módulos WEB[1]	dev/**web-***NombreModulo/*src/main/webapp/	**/css** para hojas de estilo
		/js para javascripts
		/images para las imágenes
		Las jsp/html podrán ubicarse en el directorio raiz o en subdirectorios siempre que no sean de struts o JSF
		/WEB-INF archivos xml, xsl y jsps/html redireccionados por struts/JSF
	dev/**web-***NombreModulo*/src/main/java/	En este directorio se almacenarán las clases java de negocio y controladoras necesarias para el MVC usado.

[1] Habrá más de uno si hay parte en internet e intranet, cada uno en su propio desplegable (EAR). Además, en caso de que haya parte de la presentación segura y otra que no lo es, se deberán separar los módulos web también de acuerdo a dicho criterio.

ELEMENTO	LOCALIZACIÓN	ARCHIVOS
Módulos POJO de presentación	dev/**pojo-pres-***NombreModulo*/src/main/java/	Módulos que contienen únicamente clases de la capa de presentación. Este módulo no es obligatorio, pudiendose ubicar las clases java de presentación directamente en el módulo WEB.
Módulos BATCH	dev/**batch-***NombreModulo/*src/main/webapp/	Configuración del módulo web que agrupa los trabajos Batch
	dev/**batch-***NombreModulo*/src/main/java/	Clases java con la implementación de los trabajos.
Módulos Servicios Web SOAP	dev/**ws-***NombreModulo*/{Igual estructura que para módulos web}	
	dev/**ws-***NombreModulo*/src/main/wsdl/ Nota: Se admitirá la ubicación en src\main\webapp\WEB-INF\wsdl	Archivos WSDL
	dev/**ws**-*NombreModulo*/src/main/resources/ESB	Archivos de transformación de los servicios asociados a DataPower
Módulos Servicios Web REST	dev/**rest-***NombreModulo*/{Igual estructura que para módulos web}	
Módulos POJO	dev/**pojo-***NombreModulo*/src/main/java/	Módulos que contienen únicamente clases de negocio en formato POJO (Plain Old Java Object).
Módulos DAO	dev/**dao-***NombreModulo*/src/main/java/	Fuentes del modulo de acceso a datos y script de construccion del modulo

Tabla 2. Estructura general del repositorio para código fuente

4.1.1 Módulos en el contexto de la especificación DGTE-001: Desarrollo de servicios de componentes de negocio y de acceso a datos para las aplicaciones Java EE.

De uso general dentro de la especificación:

NOTA: La nomenclatura NOMBRE_RESUMEN_TOPICO es la recomendada pero se admitirá su sustitución por NombreModulo si el proyecto lo estima conveniente.

ELEMENTO	LOCALIZACIÓN	ARCHIVOS
Módulo POJO de dominio interno.	dev/**pojo**-*nombre_resumen_topico*-**dominio-interno**/src/main/java/	Conceptos y entidades que conforman el dominio interno (modelo) de la aplicación.
Módulo POJO de negocio.	dev/**pojo**-*nombre_resumen_topico*-/src/main/java/	Lógica de negocio de la aplicación proveedora del servicio. En este proyecto estarán la implementación de los "Beans" EJB 3.0 de los Servicios de Negocio (tanto los de aplicación como los internos, caso de existir), que invocarán a diversos Servicios de Acceso a Datos. Si la complejidad del proyecto no requiere la necesidad de segregar sus interfaces en un módulo aparte, este módulo puede incliur también las interfaces de los servicios de negocio implementados.

ELEMENTO	LOCALIZACIÓN	ARCHIVOS
Módulo POJO de puerto de servicio Nota: Ver comentarios relativos al módulo de puerto de servicio para caso de adaptadores de servicios web en j2ee 1.4.	dev/**pojo**-*nombre_resumen_topico*-**local**/src/main/java/	[OPCIONAL] Puerto de servicio con las interfaces locales necesarias para invocar a los Servicios de Negocio Locales.
Módulo DAO de acceso a datos.	dev/**dao**-*nombre_resumen_topico*-/src/main/java/	Proyecto que agrupa las implementaciones de "Beans" de los Servicios de Acceso a Datos. Si la complejidad del proyecto no requiere la necesidad de segregar sus interfaces en un módulo aparte, este módulo puede incliur también las interfaces de los servicios de acceso a datos implementados.
Módulo DAO de operaciones.	dev/**dao**-*nombre_resumen_topico*-**local**/src/main/java/	[OPCIONAL] Interfaces que definen las operaciones de los Servicios de Acceso a Datos, solo accesibles a través de los Servicios de Negocio internos.

ELEMENTO	LOCALIZACIÓN	ARCHIVOS
Módulo POJO de implementación del Adaptador de SALIDA	dev/**pojo**-*nombre_resumen_topico*-**adaptador-salida**/src/main/java/	Proyecto que agrupa las implementaciones de los clientes de acceso a los servcios externos. Si la complejidad del proyecto no requiere la necesidad de segregar sus interfaces en un módulo aparte, este módulo puede incliur también las interfaces de los adaptadores de salida implementados.
Módulo POJO de interfaz del Adaptador de SALIDA	dev/**pojo**-*nombre_resumen_topico*-**local-salida**/src/main/java/	[OPCIONAL] Interfaces que definen las operaciones de los Servicios Externos, solo accesibles a través de los Servicios de Negocio internos

Tabla 3. Estructura repositorio de código fuente para la especificación DGTE-001

Para adaptadores de servicios web:

ELEMENTO	LOCALIZACIÓN	ARCHIVOS
Módulos Cliente	dev/**client-***NombreModulo*/src/main/java/	Fuentes del modulo aplicación cliente y script de construccion del modulo
Módulos Servicios Web (Adaptador del servicio).	Sigue las rutas y el nombre de módulo histórico pero bajo la nueva especificación solamente podrán existir bajo su estructura lo indicado en la especificación para el adaptador de servicio web.	

ELEMENTO	LOCALIZACIÓN	ARCHIVOS
Módulos POJO puerto de servicio.	dev/**pojo-servicio**-*nombre_resumen_topico*/src/main/java/	Puerto de servicio con las interfaces de negocio publicadas. Nota: Aplicará este POJO si el servicio Web se desarrolla en J2EE 1.4 porque si no es el caso el puerto de servicio será el marcado por la especificación en forma general.

Tabla 4. Estructura repositorio de código fuente para adaptadores de servicios web

Para adaptadores EJB: Sólo está permitida para código legado que tenga clientes activos haciendo uso de un EJB remoto.

ELEMENTO	LOCALIZACIÓN	ARCHIVOS
Módulo Adaptador EJB	dev/**ejb**-*nombre_resumen_topico*/src/main/java/	Adaptador para la publicación de servicios de negocio EJB para integración entre aplicaciones. Contiene la implementación de los "Beans" de Servicios de Negocio que la aplicación expone al exterior.
Módulo CLIENTE EJB	dev/**ejb**-*nombre_resumen_topico* **-client**/src/main/java/	Cliente de servicio con las interfaces remotas necesarias para invocar a los Servicios de Negocio publicados.

Tabla 5. Estructura repositorio de código fuente para adaptadores EJB

4.2 Nomenclatura de los paquetes

Los paquetes deberán nombrarse de acuerdo a la siguiente nomenclatura:

Paquetes de acceso a datos:

es.trafico.Acrónimo_Proyecto.<Nombre_Subsistema>.*dao*.[Nombre_Paquete]*

Paquetes de las interfaces locales de acceso a datos:

es.trafico.Acrónimo_Proyecto.<Nombre_Subsistema>.*dao.local*.[Nombre_Paquete]*

Paquetes de lógica de negocio:

es.trafico.Acrónimo_Proyecto.<Nombre_Subsistema>.*neg*.[Nombre_Paquete]*

- **Servicios internos**:
  - Interfaces:

  es.trafico.Acrónimo_Proyecto.<Nombre_Subsistema>.neg.servicios.[Nombre_Paquete]*
  - Implementaciones:

  es.trafico.Acrónimo_Proyecto.<Nombre_Subsistema>.neg.servicios.[Nombre_Paquete]*.impl

Paquete de servicios (adaptadores de integración**):**

- **Adaptadores de Servicios Web:**
  - Adaptador de servicio web:

  es.trafico.servicios.Area_de_negocio.[Topico]*.webservices.[Nombre_Paquete]*
- **Adaptador de EJB:**
  - Adaptador de Servicio EJB remoto - J2EE 1.5:

  es.trafico.servicios.Area_de_negocio.[Topico]*.remote.[Nombre_Paquete]*

  es.trafico.servicios.Area_de_negocio.[Topico]*.beans.[Nombre_Paquete]*
- **Adaptador REST:**
  - *es.trafico.servicios*.Area_de_negocio.[Topico]*.*rest*.[Nombre_Paquete]*
- **Adaptadores SCA:**
  - es.trafico.servicios.Area_de_negocio.[Topico]*.adaptador.[Nombre_Paquete]*
- **Puerto de Servicio:**

  es.trafico.servicios.Area_de_negocio.[Topico]*.local.[Nombre_Paquete]*

Paquetes de la capa de salida:

  - Interfaces:

es.trafico.Acrónimo_Proyecto.<Nombre_Subsistema>.adaptador.[Nombre_Paquete]*

- Implementaciones:

es.trafico.Acrónimo_Proyecto.<Nombre_Subsistema>.adaptador.[Nombre_Paquete]*.impl

Paquetes de la capa de presentación:

es.trafico.Acrónimo_Proyecto.<Nombre_Subsistema>.*pres*.[Nombre_Paquete]*

Paquetes clases dominio (Dominio DGT):

es.trafico.Area_de_negocio.[Topico]*.*dominio*.[Nombre_Paquete]*

Paquetes clases transversales:

- **Utilidades:**

  es.trafico.Acrónimo_Proyecto.<Nombre_Subsistema>.*util*.[Nombre_Paquete]*

- **Dominio interno:**

  es.trafico.Acrónimo_Proyecto.<Nombre_Subsistema>.modelo.[Nombre_Paquete]*

Componentes Comunes de infraestructura: Los componentes comunes de infraestructura no tendrá que ajustar la paquetería a las capas lógicas de las aplicaciones. Su nomenclatura tendrá el siguiente patrón: *es.trafico.framework.topico.[nombrePaquete]**

topico: será obligatorio y describirá el propósito general del componente y será único y global para todos los paquetes del componente

Paquetes de la capa batch:

es.trafico.Acrónimo_Proyecto.<Nombre_Subsistema>.*batch*.[Nombre_Paquete]*

Paquetes clases clientes (nomenclatura de paquetes aplicable a los clientes de servicio web o similar):

es.trafico.cliente.Area_de_negocio.[Topico]*

es.trafico.cliente.Area_de_negocio.[Topico]*.*dominio*.[Nombre_Paquete]*

es.trafico.cliente.Area_de_negocio.[Topico]*.*excepciones*.[Nombre_Paquete]*

es.trafico.cliente.Area_de_negocio.[Topico]*.*webservices*.[Nombre_Paquete]*

Siendo:

< >: Paquetes de existencia opcional.

[]*: Múltiples paquetes de existencia opcional

Acrónimo_Proyecto: Son las 4 letras que describen de cara a la organización la aplicación en forma unívoca.

<Nombre_Subsistema>: Paquete opcional existente para una partición en módulos de aplicaciones de gran tamaño. No suele ser utilizado. No tiene por qué tener correspondencia con el concepto de Subsistema que maneja esta guía en los patrones de gestión de la configuración.

Tópico: Este nombre de paquete no debe tener referencias tecnológicas (Ejemplo: dao, neg, util, etc.). En él se realizan categorizaciones de conceptos de negocio del área, si se considera necesario. Para más detalle al respecto consultar las especificaciones referidas anteriormente donde se describe el significado concreto que se debe aplicar a "Area_de_negocio" y "Topico".

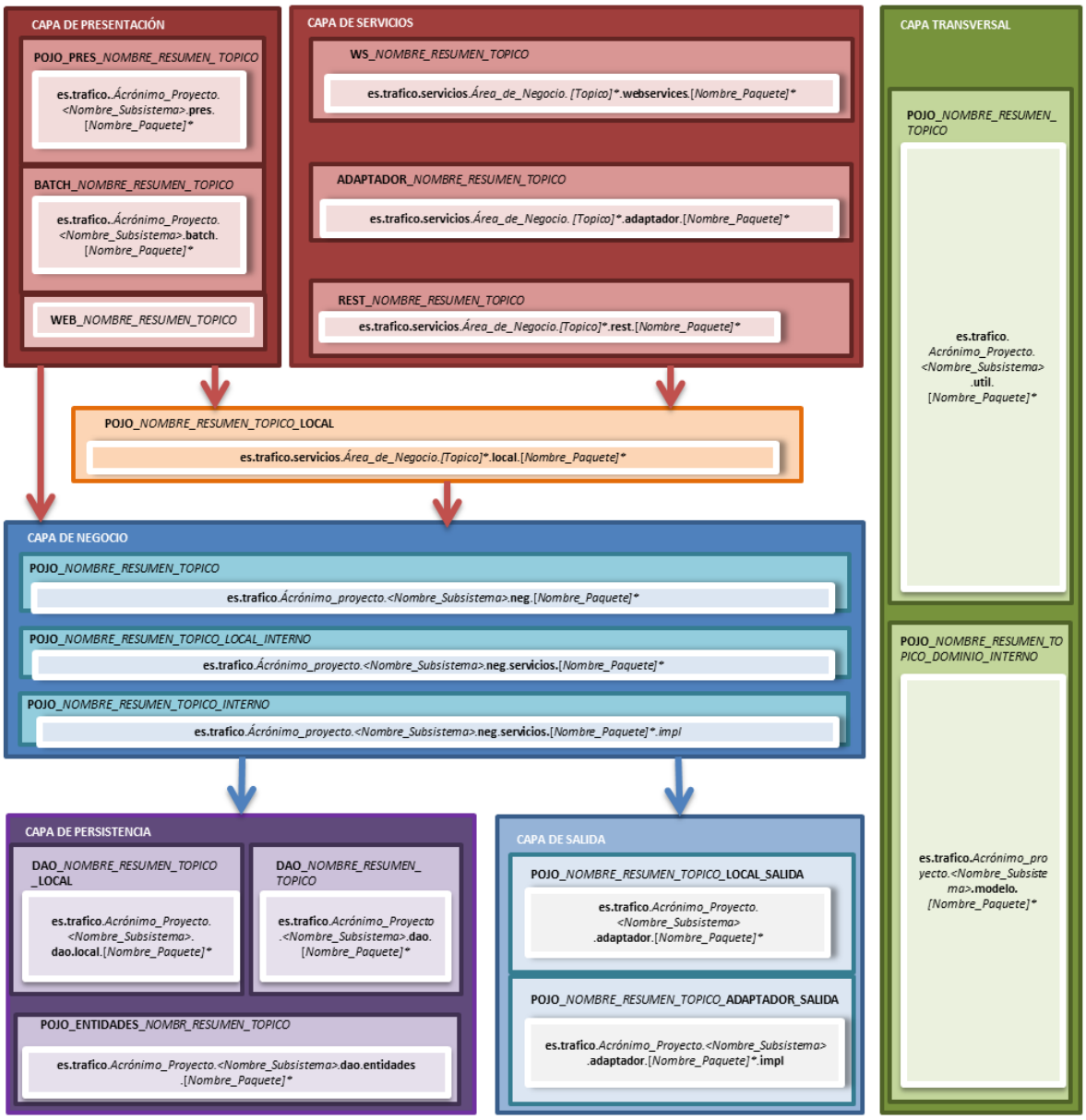

Ilustración 4. Resumen Nomenclatura de Módulos y Paquetes

4.3 Framework DGT

4.3.1 Componentes Comunes

La DGT ha establecido una serie de componentes comunes que deberán utilizar todos los proyectos. Los componentes comunes proporcionados pueden ser consultados en el siguiente directorio *\\dgt.red\sc\Gi\Calidad\SOA\Componentes*. Se recomienda el uso de la última versión publicada de los componentes comunes.

La lista de componentes comunes de uso obligatorio es la siguiente:

CHST. Sólo si la aplicación tiene accesos al Host.

CLOG. Componente de trazas

CAUDIT. Componente de auditoría.

CCSG. Módulo de login.

CXCP. Jerarquía base de excepciones.

4.3.2 Servicios de Infraestructura

La DGT ha establecido una serie servicios que deberán utilizar todos los proyectos que requieran de la funcionalidad que proveen. Al igual que los componentes comunes, los servicios de infraestructura proporcionados pueden ser consultados en el siguiente directorio *\\dgt.red\sc\Gi\Calidad\SOA\Componentes*.

A continuación, describimos los servicios más significativos:

4.3.2.1 SUSR. Servicio de Información de Usuarios y Unidades

Este servicio ofrece toda la información de los usuarios existente en el repositorio de usuarios, con excepción de la información protegida.

4.3.2.2 SADD. Servicio de diccionarios de datos comunes

Servicio de obtención de diccionarios de datos comunes. Ofrece como servicio datos comunes como provincias, códigos postales, comunidades autónomas, municipios, etc. Asimismo ofrece el directorio común de organismos de la administración (DIR3).

4.3.2.3 SFIR. Servicio de Firma Digital

El Servicio de Firma Digital valida certificados y realiza firmas de servidor con el sello de DGT (con y sin sello de tiempo).

4.3.2.4 FIRe. Firma Electrónica certificado cliente

Las aplicaciones de la DGT tienen la posibilidad de integrarse con el servicio de infraestructura FIRe para la realización de la firma electrónica con certificado cliente. Este servicio, distribuido por la Administración General del Estado, contiene la lógica necesaria para realizar la firma bien con certificados locales, bien con certificados en la nube (custodiados por la Dirección General de la Policía y accesibles a través de la pasarela habilitada por GISS).

Las instrucciones de desarrollo están disponibles en el *"Manual del Integrador"*, disponible en el Portal de Administración Electrónica (PAe). Todas las librerías necesarias para completar el desarrollo, están disponibles en el repositorio de artefactos de DGT.

4.3.2.4.1 FIRe. Solicitud de alta de aplicaciones en el Sistema FIRe

Las aplicaciones que requieran integrarse con este servicio, deben realizar una solicitud al departamento de arquitectura (dep.arquitectura@dgt.es), aportando la siguiente información:

- ASUNTO: [Acronimo_Proyecto] Alta aplicación Sistema FIRe
- Datos de la solicitud:
  - Nombre de la aplicación: (Acrónimo y descripción de la misma)
  - Responsable/s
    - Correo y teléfono de contacto de cada uno de los responsables
  - [OPCIONAL] Identificadores de usuario de pruebas: en el entorno de desarrollo el servicio de firma en nube (cl@veFirma) es un *mock*, en el que se simula la firma con un certificado ficticio. Se creará un certificado ficticio por cada uno de los identificadores de usuarios facilitados y se utilizarán en el mecanismo de firma simulada en nube.

Como respuesta a la solicitud, se facilitará la siguiente información, necesaria para incluir en la configuración de la aplicación que se va a integrar con FIRe:

- Identificador de la aplicación para el entorno de desarrollo.
- Certificado cliente de autenticación para el entorno de desarrollo.

- [OPCIONAL] *Password* de los usuarios de prueba facilitados para completar la autenticación en el proceso simulado de firma en nube del entorno de desarrollo.

4.3.2.4.2 Configuración en Preproducción y Producción

Las aplicaciones deben incluir la configuración de integración con FIRe en ficheros de propiedades dependientes de entorno.

El procedimiento contempla que la obtención del identificador de aplicación y el certificado para los entornos de Preproducción y Producción, serán obtenidos por Operaciones en el momento de despliegue, si así se informa en las instrucciones de despliegue y serán incluidos en los ficheros de configuración correspondientes.

5 Estándar de Construcción

5.1 Configuración de despliegue

Se establecerán las siguientes rutas en los archivos de configuración para el despliegue:

ELEMENTO	LOCALIZACIÓN	LOCALIZACIÓN EN EL SERVIDOR	ARCHIVOS
Módulo conf	Para proyectos gradle, se localizará en: conf/src/conf/dist conf/src/rec/dist (ver anexo 17.02 para más detalle)	/opt/appdata/conf/XXXX /opt/appdata/rec/XXXX /opt/appdata/lib/XXXX	En los módulos de configuración indicados en la columna LOCALIZACIÓN, se incluirán los ficheros de configuracion del proyecto (parámetros que puedan cambiar sin que afecte a la generación del EAR). En la carpeta **rec** se incluirán los ficheros de recursos del proyecto incluyendo aquí los ficheros para la implementación de multiidioma. En la carpeta **lib** se incluirán librerías adicionales que son necearias instalar en el servidor.

Tabla 6. Estructura general de carpetas de archivos de configuración para el despliegue

Nota: La variable CLASSPATH del servidor de aplicaciones apuntará a nivel **/opt/appdata/conf** y **/opt/appdata/rec** en el servidor de aplicaciones.

5.2 Scripts de base de datos

Se establecerán las siguientes rutas para los scripts de base de datos:

ELEMENTO	LOCALIZACIÓN	ARCHIVOS
Scripts de base de datos	dev/scripts/bbdd/carga	entencias SQL de carga de datos, o de actualización de los mismos
Scripts de instalación	Instalador del producto dev/scripts/instalador Creación del esquema en la base de datos: dev/scripts/bbdd/instalacion	En dev/scripts/bbdd/instalacion se localizaran los scripts de creación del esquema de datos o de actualización del mismo.
Scripts de carga en el Almacén de datos	dev/scripts/bbdd/almacen	Scripts o procesos batch para las cargas en el almacén de datos
Scripts de procesos *batch*	dev/scripts/batch	En esta carpeta estarán los procesos batch de la aplicación. Nota1:Esta ubicación queda a efectos históricos cuando se publique la especificación batch. Se promueve el uso de "ear's" para despliegue de los procesos batch. Se debe hablar con el departamento de arquitectura hasta su publicación sobre las directrices a seguir. Nota2: Para los proyectos gradle, los scripts de shell deberán incluirse en el subproyecto conf para generar el zip de scripts correspondiente, en la ruta src/scriptsshell/dist. Se puede consultar el anexo 17.2 Normativa de proyectos Gradle para más información sobre cómo configurar en gradle la generación de estos empaquetados

Tabla 7. Estructura general de carpetas de archivos de base de datos

5.3 Fuentes del Host

ELEMENTO	LOCALIZACIÓN	ARCHIVOS
Fuentes Host de aplicación	dev/Host/src/	Fuentes pertenecientes a la aplicación Host .
Includes de Host	dev/Host/includes/	Contendrá todos los includes nuevos o modificados por el propio desarrollo.
JCLs y SYSIN	dev/Host/JCL/ dev/Host/SYSIN/	En JCL estarán todas las cadenas nuevas o modificadas, y en SYSIN los miembros necesarios para la ejecución de estas cadenas.

Tabla 8. Estructura general de carpetas de archivos de Host

5.4 Implementación de Usuario de Control/ Página de Test

El equipo de desarrollo implementará una página web de testeo, que permita controlar el correcto funcionamiento de los sistemas a los que accede la aplicación. Esta debe chequear aquellos elementos de infraestructura que va a necesitar para llevar a cabo un correcto funcionamiento así como el chequeo de los interfaces entre servicios adyacentes. Estos interfaces deben de estar incluidos en el documento de implantación.

5.4.1 Diseño

El equipo de desarrollo implementará una página web de testeo, que permitirá controlar el correcto funcionamiento de los sistemas a los que accede la aplicación. Ésta deberá chequear aquellos elementos de infraestructura necesarios para un correcto funcionamiento, así como, el chequeo de los interfaces entre servicios adyacentes; estos interfaces deben de estar incluidos en el documento de implantación.

Cualquier aplicación de tipo Web Navegable, de tipo Web-Service o de tipo mixta que incluyan ambos módulos necesitará obligatoriamente esta página de test.

La implementación de esta página de test va a requerir seguridad declarativa empleando el LDAP Corporativo. Cada aplicación deberá de implementar un ROL o "perfil de java" que permitirá la ejecución de la página de test, añadiendo un perfil de usuario de tipo Test en el documento de "Perfiles de Usuario" y nombrado en el documento de "Implantación". Cada perfil corresponde a un grupo de LDAP. Dicho Rol, será mapeado en el servidor de aplicaciones WebSphere, durante el proceso de instalación de las aplicaciones.

El tipo de autenticación será el mismo que el resto de la aplicación, mediante <Usuario/Password> o bien, mediante empleo de "Certificado de Cliente". El usuario y password son previamente definidos en el LDAP Corporativo.

En el entorno de Producción el usuario de test no va a ser conocido por el área de Desarrollo. Dicho usuario de test dispone de un "Certificado de Cliente" válido para la DGT.

En el entorno de Preproducción, el Área de Desarrollo deberá de proveer un usuario de test y certificado de cliente, este último, en los casos en que proceda.

5.4.2 Implementación

El funcionamiento de la página de test será el siguiente:

- Se accederá a la funcionalidad especial de test en la url <URL_Aplicación>/test
- Se realiza el proceso de autenticación.
- Como resultado de la ejecución de la página de test, mostrará una cadena de texto devolviendo el estado de los elementos de infraestructura e interfaces definidos, siguiendo la siguiente nomenclatura:
    - <Acrónimo de proyecto>: Asignado por el Departamento de Calidad.
    - <Versión>: La que se entrega a calidad y obtiene su correspondiente etiqueta.
    - <Elemento de Infraestructura>: OK/ERROR
        - Ejemplo, BBDD: OK/ERROR , LDAP: OK/ERROR
    - <Código proyecto interfaz aplicación> : OK/ERROR
        - Ejemplo, GRGT: OK/ERROR, RELE: OK/ERROR
    - HOST <tipo Host> -> OK/ERROR

- Ejemplos:
  - HOST Conductores: OK/ERROR
  - HOST Vehículos: OK/ERROR
  - HOST Personas: OK/ERROR
- <Sistemas Externos>: OK/ERROR
  - Ejemplo, ATEX https://hostname/WS_ATEX/services/ATEX: OK/ERROR

5.4.3 Monitorización

La monitorización de los servicios de la DGT se lleva a cabo exclusivamente por el departamento de Sistemas.

Para ello contamos con un sistema de monitorización que permite ejecutar páginas de test, o bien, navegaciones programadas, en las aplicaciones. En función de los resultados obtenidos en la ejecución se envían vía mail las incidencias producidas a las diferentes listas de correo designadas para cada área de la DGT o proyecto "incidencias.<área/proyecto>@dgt.es" .

En caso de que una aplicación requiera implementar un sistema de notificaciones de eventos particular estos deben de ser justificados en la fase de diseño.

5.5 Contenidos estáticos

El contenido estático es aquel que no necesita ser procesado para ser servido al cliente web, como por ejemplo son las imágenes (jpg, gif,…), páginas web (html, css, js,…) y documentos (doc, pdf's, xls,…).

Toda aplicación que necesite trabajar con contenidos estáticos deberán de ser publicados en la capa de frontales web. Estos contenidos deberán de ser servidos por los servidores de tipo WebServer IHS y ubicados en el "Docroot", especificado en el fichero de configuración http.conf.

Las aplicaciones deben de organizar los contenidos estáticos dentro de un directorio llamado con el acrónimo de proyecto designado por Calidad (escrito en mayúsculas).

Ejemplo: /SEDE/…. o /COND/….

Los servidores WebServer disponen de un “único” servidor virtual definido, por cada dominio WEB, para cada uno de los entornos. No se permite realizar configuración adicional de los servidores virtuales definidos. Esto quiere decir que las aplicaciones deben de tener en cuenta este hecho en la fase de implementación de las aplicaciones.

5.6 Ficheros de logs

Todos los servicios deberán de integrar el componente común de log de la DGT (CLOG) creado para estandarizar el tratamiento de ficheros de traza.

Los ficheros empleados para la monitorización de la aplicación son tres: eventos.log, actividad.log y debug.log. El primero de los mencionados es el empleado por sistemas para la explotación directa y automatizada de las aplicaciones. De su correcto uso se derivará una monitorización exitosa del servicio.

Para mayor detalle vea el anexo “22.2 Clasificación de eventos de explotación.doc” y el manual de uso del componente común de log.

5.6.1 Eventos.log

En el fichero de “Eventos.log” solo deben de aparecer códigos de errores de aplicación. Estos códigos de errores serán los que se detallen en el documento de explotación. Estos errores serán los lanzados como ERROR o FATAL según la especificación log4j empleada en la DGT.

Están prohibidos los volcados de pila de las excepciones en dicho fichero de log.

Este fichero es empleado exclusivamente para el sistema de Monitorización de Servicios.

El uso del componente de log de la DGT garantiza que el sistema podrá ser monitorizado en tiempo de explotación de forma adecuada ya que garantiza una estructura fija normalizada.

Cuando se hace uso del componente debe indicar el código de error y una descripción del mismo. La parte descriptiva del error se recomienda que siga el formato “Origen ## Descripción”. Tenga en cuenta los separadores b##b para su correcta monitorización. No obstante esta parte descriptiva es libre aunque si la sigue podrá obtener si se requiere un grano mayor en la monitorización.

Ejemplo:

[15-03-2011] [12:54:40,438] [ERROR] [PAGO2/CORE] [PAGO2000] [HOST ## Falta de conectividad]

5.6.2 Actividad.log

Este fichero no es explotado por sistemas y se puede disponer de él si el equipo de desarrollo necesita sacar datos adicionales de la aplicación. En este fichero de acuerdo con la especificación de log4j se obtendrán todos los mensajes lanzados INFO Y WARN además de los ERROR y FATAL.

Se recomienda incluir en este fichero todos los controles de negocio que necesite. Por ello es recomendable cuando trate los errores de aplicación discernir si son de negocio o son de infraestructura.

Ejemplo:

[15-03-2011] [12:54:40,438] [INFO] [PAGO2/CORE] [] [Se orquestan bien los subsistemas]

5.6.3 Debug.log

Este fichero de log está deshabilitado en el entorno de producción. Se puede solicitar la activación de este fichero de traza bajo demanda en casos puntuales de resolución de incidencias y a petición del Jefe de Proyecto de la DGT.

Ejemplo:

[15-03-2011] [12:54:40,438] [DEBUG] [PAGO2/CORE] [] [Se anota el pago ref 12121312]

5.7 Colas JMS

En vista a las múltiples opciones que presenta la configuración de Colas JMS en WebSphere, se ha elaborado un documento guía como un nuevo estándar para la organización DGT. Todos los grupos de desarrollo pueden consultar el documento anexo para proceder a realizar las configuraciones de entorno en las zonas de Desarrollo.

Dicho documento contiene todos los aspectos de configuración de colas JMS para soportar alta disponibilidad.

Ver “Anexo 22.01. Configuración Colas JMS”.

Para cualquier duda o aclaración se puede dirigir la petición buzón de sistemas.

5.8 Ficheros de Propiedades

5.8.1 Estructura directorios

Cada aplicación va a disponer de una estructura de directorios en los servidores de aplicaciones para ubicar los diferentes ficheros entregables. La ubicación en el servidor de estos ficheros siempre será /opt/appdata/<conf, rec>/<XXXX>. La cadena XXXX corresponde al Acrónimo de Proyecto escrito en "mayúsculas", "conf" directorio de ficheros de configuración, "rec" directorio de ficheros de recursos.

Los ficheros de configuración y de recursos se empaquetarán en zips independientes durante el proceso de compilación (consultar el anexo 17.2 Normativa de proyectos Gradle para más información sobre cómo configurar en gradle la generación de estos empaquetados).

5.8.2 Ficheros de recursos

En el subdirectorio de registro "**rec**" podemos encontrar plantillas, menús parametrizados, ficheros xml, fuentes necesarias por la aplicación y en general recursos que necesite la aplicación y no tengan otro sitio donde colocarse. Los certificados sin embargo deben de ir dentro del servidor de aplicaciones y no deben ir nunca dentro del rec.

5.8.3 Ficheros de configuración

En el directorio de configuración "**conf**" se indicarán los dos ficheros de parámetros de configuración. Se contemplan 3 tipos de ficheros de configuración:

5.8.3.1 Tipos de ficheros de configuración

5.8.3.1.1 Ficheros de configuración no dependientes de entorno

En el fichero con nombre *"<nombre_fichero>.properties*", se indicarán los parámetros no dependientes del entorno que corresponden a la configuración interna de aplicación.

5.8.3.1.2 Ficheros de configuración dependientes de entorno de aplicación

En el fichero con nombre *"<nombre_fichero>**_DE_app**.properties*", se indicarán los parámetros de configuración dependientes del entorno cuyos valores son conocidos por la aplicación y son diferentes por cada entorno de ejecución.

5.8.3.1.3 Ficheros de configuración dependientes de entorno de Sistemas

El segundo fichero, con nombre *"<nombre_fichero>**_DE_sistemas**.properties*", se indicarán los parámetros de configuración dependientes del entorno cuyos valores son conocidos y custodiados por el departamento de Sistemas, como por ejemplo: *URLs*, usuarios, contraseñas,…

5.9 Procesos Batch

5.9.1 Tipologías

Actualmente la DGT cuenta con procesos batch en los formatos que a continuación se describen. Para otras formas de procesos batch se debe consultar con el área de Arquitectura para validar su uso.

Independientemente del formato, es de uso obligatorio el lanzamiento de los procesos con el planificador de IBM TWS.

5.9.1.1 Shell Script

El arranque de batchs desde el planificador puede acometerse mediante la ejecución de Shell scripts de tipo Linux. Para ello desde el proyecto se diseñará el entregable del subsistema Batch con los siguientes recursos:

- Desarrollo de un Shell script que invoca al ejecutable Java standalone (JAR), que empaqueta el código del subsistema Batch.
- En cuanto a la ruta para la invocación Java, tanto el Shell script como el ejecutable Java estarán ubicados en la misma máquina de la instalación de WebSphere.
- Shell script y ejecutable de java deberán empaquetarse en un zip durante el proceso de compilación.

- Las librerías de proveedores necesarias para el runtime del ejecutable Batch serán localizables con la carga del classpath, ya sea a través del MANIFEST incluido en el JAR o definiendo variable desde el Shell script.
- Las rutas generadas para el Classpath tendrán en cuenta la configuración del entorno destino de ejecución. Para entornos gestionados por Sistemas se tendrá en cuenta la configuración indicada en la sección **5.9.2 Estructura de los procesos batch**.
- Shell script devolverá código de salida OK, o código de error para las acciones del sistema asociadas a tareas de Operaciones.

Se debe consultar al departamento de Sistemas las versiones de Java disponibles en los entornos de ejecución. En caso de necesitar una versión de Java no disponible, ésta se debe solicitar a dicho departamento.

Este tipo de scripts Shell-Java deberán de ser planificados por el Planificador TWS. El entorno de ejecución, aun estando en la zona Intranet, no es accesible por los servicios expuestos en Datapower.

Se permite el uso de JDBC para acceso directo a BBDD.

5.9.1.2 JEE

Se dispone también de una instalación de WebSphere 7.1 en cada una de las máquinas destinadas a la ejecución de procesos batch. Está permitido instalar procesos batch sobre esta instalación del servidor de aplicaciones. Estos deberán ser empaquetados en formato ".ear".

5.9.2 Estructura de los procesos batch

Los procesos batch deben seguir la siguiente estructura de directorios, donde XXXX es el acrónimo del proyecto:

- /opt/appdata/conf/<XXXX>/ Ficheros de configuración necesarios para la ejecución del proceso batch
- /opt/appdata/rec/<XXXX>/ Ficheros de recursos del proceso batch.
- /opt/appdata/lib/<XXXX>/ Directorio donde se encuentran las librerías necesarias para la ejecución del proceso batch.

- /opt/appdata/bin/<XXXX>/ Directorio donde se encuentran los ejecutables que serán lanzados y planificados por TWS. El nombre de los scripts en este directorio seguirá la nomenclatura PB_SH_<id>.sh
- /var/log/aplicaciones/<XXXX> Directorio donde se almacenan los logs actividad.log y debug.log. El fichero de eventos.log se escribirá en /var/log/aplicaciones

La propiedad y ejecución de estos ficheros será realizada por un único usuario estándar del sistema que no tendrá privilegios ni de super-usuario ni de administración.

5.10 Servicio corporativo de directorio LDAP

5.10.1 Entornos y configuración de clientes / conexiones

El servicio corporativo de LDAP de la DGT consta de diferentes instancias en función del entorno de ejecución. Se deben solicitar al departamento de Sistemas los datos de configuración para la conexión a cada una de dichas instancias.

5.10.2 Políticas aplicadas en los LDAP sobre los usuarios

- NO SE PERMITE crear usuarios de prueba o genéricos en el LDAP de producción.
- NO SE PERMITE crear usuarios personales en los LDAP de preproducción y desarrollo.
- NO SE PERMITE definir atributos en LDAP cuyo uso exclusivo sea para realizar autorización programática.
- La ubicación y nombre de los usuarios genéricos y de prueba en el LDAP de preproducción y desarrollo será decidida por sistemas.
- NO SE PERMITEN consultas que devuelvan más de 1000 resultados.

5.10.3 Políticas aplicadas en los LDAP sobre proyectos y perfiles

Cada proyecto consta de un código de proyecto, normalmente de cuatro letras y un atributo de presencia, que deben estar documentados en el documento de perfiles.

La entrada en el LDAP del proyecto será el código de proyecto, colgando del área del proyecto.

Los perfiles del proyecto se obtienen también del documento de perfiles, cada perfil se corresponde a un grupo del LDAP y un valor para el atributo de presencia. Así el acceso a un perfil supone la inclusión del usuario en el grupo de correspondiente y la asignación del atributo de presencia con el valor correspondiente para ese perfil.

La ubicación y el nombre del grupo de LDAP será inmediatamente debajo de la entrada de proyecto y comenzando el nombre por el código de proyecto y continuando con algo asociable intuitivamente al perfil. Esta información debe constar en el documento de perfiles.

Todo proyecto constará de un perfil de test, necesario para el testeo de la aplicación, donde se incluirá el usuario de testeo de la aplicación de sistemas, así como otros si fuera necesario.

Ejemplo:

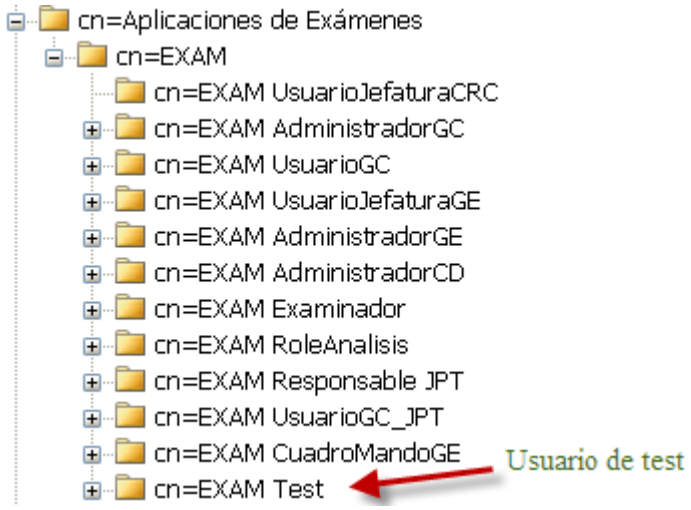

Ilustración 5. Ejemplo de definición de grupos LDAP

El alta y baja de los perfiles debe hacerse con la aplicación GEUS (gestión de usuarios de la D.G.T.).

Cualquier excepción de estas políticas debe justificarse y será necesaria la aprobación expresa de la DGT.

6 Cambios y Novedades

6.1 Guía de desarrollo 2019. Estándar de Arquitectura y Diseño

6.1.1 Documentación de Diseño de Arquitectura:

Se da la posibilidad s los proyectos de documentar el Diseño de Arquitectura mediante un documento ofimático. Se permitirá el uso de la herramienta *Rational Software Architect* (RSA) durante la vigencia de esta nueva Guía de Desarrollo.

6.1.2 Capas lógicas de Arquitectura lógica de aplicaciones

- **Adaptadores de Entrada. Capa de Servicios:**
  - **EJB remotos**. Se elimina la posibilidad de definir interfaces a los servicios mediante EJB remotos. Las aplicaciones que provean este tipo de interfaces, deben proporcionar interfaces web (SOAP/REST) alternativas.
  - **Diseño API:** se impulsa la definición de interfaces REST en la capa de servicios y se establece las directrices de diseño de APIs en la nueva especificación: "*DGTE-008 Diseño APIs*".
- **Capa de Persistencia:** se elimina la obligatoriedad de usar el servicio común de acceso a datos (DAAS) para acceder a los modelos de datos propios.
- **Dominio DGT:** se elimina la necesidad de definir el "Dominio DGT". Se mantiene por retrocompatibilidad con proyectos que mantienen EJB remotos y/o clientes de servicio web (a extinguir).
- **Adaptadores de Salida:** se da libertad al proyecto en la estrategia de modularización de esta capa. Se permite agrupar en un único módulo todos los adaptadores de salida que usa la capa de negocio de la aplicación.

6.1.3 Nomenclatura de módulos.

Se modifica la nomenclatura de módulos para adecuarla a la Normativa de Proyectos Maven (Anexo 17.1) y Proyectos Gradle (Anexo 17.2)

6.1.4 Configuración de aplicaciones

Se establece una distinción entre los ficheros de configuración dependientes de entorno. Se deben de generar ficheros diferentes para diferenciar los parámetros que conoce Sistemas de los que gestiona directamente la aplicación como elementos de configuración funcional.

6.1.5 Componentes y servicios comunes:

- Se elimina la obligatoriedad de usar la **última versión** de los componentes comunes. Esta directriz de obligado cumplimiento, pasa a ser recomendación.
- **Componente ESAD**: se elimina la obligatoriedad de uso de este componente común.
- Se describe el procedimiento de solicitud para permitir la integración de aplicaciones con el Sistema FIRe.