|  |
| --- |
| **NORMATIVIDAD** |
| **Buenas prácticas para el desarrollo de software** |
| **SUB DIRECCIÓN DE DESARROLLO DE PROYECTOS - OSCE** |

Revisión Histórica

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Autoría** | **Descripción** |
| 01/04/2014 | 1.0 | Milton García Llamoca. | Creación del documento. |
| 29/05/2015 | 1.1 | Milton García Llamoca | Inclusión practicas varias. |

CONTENIDO

[**1.** **Marco de Referencia** 4](#_Toc423954859)

[**1.1.** **Propósito** 4](#_Toc423954860)

[**1.2.** **Alcance** 4](#_Toc423954861)

[**1.3.** **Definiciones, Acrónimos y Abreviaciones** 4](#_Toc423954862)

[**2.** **Aplicaciones empresariales basadas en Java** 4](#_Toc423954863)

[**2.1.** **Generales** 4](#_Toc423954864)

[**2.2.** **Liberación de aplicaciones** 4](#_Toc423954865)

[**2.3.** **Implementación en capa de utilitarios** 5](#_Toc423954866)

[**2.4.** **Implementación en capa de datos** 5](#_Toc423954867)

[**2.5.** **Implementación en capa de negocio** 6](#_Toc423954868)

[**2.6.** **Implementación en capa de presentación** 6](#_Toc423954869)

[**3.** **Aplicaciones compuestas** 8](#_Toc423954870)

[**3.1.** **Generales** 8](#_Toc423954871)

[**3.2.** **Liberación de aplicaciones** 8](#_Toc423954872)

[**3.3.** **Implementación en capa de datos** 8](#_Toc423954873)

[**3.4.** **Implementación en capa de actividades de negocio** 8](#_Toc423954874)

[**3.5.** **Implementación en capa de procesos de negocio** 8](#_Toc423954875)

[**3.6.** **Implementación en capa de presentación** 8](#_Toc423954876)

**BUENAS PRÁCTICAS PARA EL DESARROLLO DE SOFTWARE**

1. **Marco de Referencia**
   1. **Propósito**

Este documento describe las prácticas recomendadas que deben aplicar durante el diseño y construcción de aplicaciones software.

* 1. **Alcance**

Este documento es de interés a los equipos del OSCE y empresas externas que ejecuten proyectos de nuevos desarrollos de software y/o mantenimiento de software.

* 1. **Definiciones, Acrónimos y Abreviaciones**

Las definiciones y acrónimos deben referenciarse del documento de arquitectura de software.

1. **Aplicaciones empresariales basadas en Java** 
   1. **Generales**

* La dependencia de clases y/o recursos debe ser lineal de manera que no se generen dependencias circulares dentro del mismo binario y entre binarios.
* El desarrollador para realizar seguimiento de su implementación podrá hacer uso de los utilitarios LOG en nivel DEBUG los mensajes a mostrar en los entornos de producción deben ser configurados de WARN hacia mayor criticidad.
* El uso de log debe incluir el criterio de cuando un evento es conocido se escribe una frase especifica sin traza y cuando el evento es desconocido y/o tiene múltiples causas raíz debe incluir la traza.
* Las implementaciones “Singleton” deben incluir un bloque de sincronización durante la creación de la instancia y no del método por el cual se accede a la instancia.
* Las implementaciones de acceso lectura/escritura a base de datos pueden usar sintaxis JPQL y/o SQL Nativos y/o Procedimientos almacenados considerando los criterios de mantenibilidad y desempeño esperado.
* Las consultas hacia base de datos deben ser evaluadas para optimizar los objetos de base de datos que sean necesarios, esto incluye indexación, particionamiento, etc.
* Las funcionalidades de procesamiento masivo de datos en base de datos puedan hacer uso de procedimientos almacenados para evitar la carga de datos en las aplicaciones.
* El código que haga uso de la técnica try/catch debe implementar el bloque finally para codificar la liberación de los objetos y/o variables para minimizar el procesamiento que realiza el recolector de basura para liberar memoria.
* Hacer uso adecuado de técnica de herencia de manera que conceptos similares puedan ser mantenible con facilidad, ejemplo: “Persona” sería una clase general y las clases derivadas serian “PersonaNatural”, “PersonaJuridica”, etc.
* En el mantenimiento de la aplicación se debe incluir comentarios que identifique los bloques de código con respecto a la implementación de un requerimiento y/o levantamiento de incidencia.
  1. **Liberación de aplicaciones**
* Los recursos de aplicación como reportes Jasper deberán ser ubicados en sistema de ficheros externos al empaquetado de la aplicación.
* Los recursos web estáticos (javascript, imágenes, css, etc.) de las aplicaciones web dinámicas deben ser externalizados en proyectos web estáticos.
* Las opciones de configuración, valores, parámetros de las aplicaciones empresariales basadas en java deben ser escritas/leídas desde archivos de propiedades o archivos XML y ubicadas externamente al empaquetado de la aplicación.
* Cada aplicación empresarial desplegada debe tener su propio registro de eventos ubicadas externamente al empaquetado de la aplicación.
* Los binarios que representen clientes deben minimizar la dependencia hacia librería de terceros de manera que sean utilizados sin adicionar estas dependencias en las aplicaciones consumidoras.
* Lo recursos estáticos javascript y css de las aplicaciones web deben ser optimizados para minimizar su tamaño haciendo uso de herramientas de compresión y/u ofuscamiento de contenido.
  1. **Implementación en capa de utilitarios**
* Las clases de la capa utilitarios deberá contener aquellas funcionalidades que es de uso genérico en distintos proyectos.
  1. **Implementación en capa de datos**
* Las clases de entidades persistentes debe tener todos las configuraciones y/o anotaciones y/o valores necesarios que permita establecer un mapeo absoluto hacia el objeto de base de datos que le corresponda.
* Para las consultas de base de datos donde intervengan múltiples entidades persistentes deben realizarse con sentencias JPQL, SQL nativo o procedimientos almacenados para obtener solo los valores de atributos/columnas requeridos.
* Los atributos de las clases del modelo de entidades persistentes (clases JPA) deben corresponder en tipo a los atributos equivalentes en las clases del modelo de entidades de negocio (clases DTO).
* Los atributos mapeados a columnas de base de datos en las clases de entidades persistentes deben ser declaradas usando las clases envoltorios que corresponda a los de tipo de datos primitivos, ejemplo usar Integer en lugar int, Boolean en lugar de boolean, etc.
* Los valores de uso recurrente en las aplicaciones deben ser centralizadas en clases que contendrán las constantes y/o enumerados, el uso de uno o más clases/enumerados dependerán del contexto de información usado para agrupar estos valores.
* El modelo de entidades persistentes debe tener configurada la pertenencia de esquema y/o catalogo dentro de las notaciones y/o configuraciones de las clases de entidades persistentes.
* El modelo de entidades persistentes debe tener configurada la pertenencia de esquema y/o catalogo dentro de las notaciones y/o configuraciones de las clases de entidades persistentes.
* El modelo de entidades persistentes los atributos mapeados a columnas de base de datos en las clases de entidades persistentes deben ser declaradas usando las clases envoltorios que corresponda a los de tipo de datos primitivos, es decir usar Integer en lugar int, usar Long en lugar long, etc.
* Para el uso de JPA, las relaciones entre clases deben estar definidas con “Lazy” en lugar de “Eager”, esto para evitar carga de datos innecesaria.
* Para el uso de JPA, los atributos numéricos de las clases de entidades persistentes deben ser declarados como Double, BigDecimal, Long, Integer, etc. según el valor de negocio que almacenara tomando como criterio el rango de valores que soportara y el uso de decimales, ejemplo no deberá usarse Double para un atributo de propósito cantidadPersonas.
* Los atributos que representan montos e importes deben ser declarados con “BigDecimal” en lugar de “Double”, considerar que cualquier cálculo que se pudiera realizar se da con mayor precisión usando la clase BigDecimal.
* Para el uso de JPA, los atributos de tipo fecha en las clases de entidades persistentes deben ser declaradas con “Date” en lugar de “Timestamp” y en los casos donde se requiera leer/grabar el valor con el formato fecha/hora usar la notación @Temporal(TemporalType.TIMESTAMP).
* En los formularios con tipo de búsqueda el acceso a los datos debe implementarse con sentencias JPQL adecuadas para minimizar la lectura de columnas en tablas de base datos.
* Para el uso de JPA, cuando se haga uso de secuencias se debe establecer claramente el número de valores que se tendrá en cache de memoria el cual debe ser proporcional al número de transacciones de inserción esperado.
* Cuando se haga uso de secuencias el valor máximo usado por la aplicación debe ser el valor máximo de la secuencia en base de datos, es decir el incremento del secuencial en base de datos debe corresponder a los valores usados dentro de la aplicación.
* Las consultas hacia base de datos deben incluir solo los atributos/columnas necesarios, evitar el uso las consultas donde el criterio es traer todas las columnas posibles bajo la perspectiva de reutilización.
* Las consultas de búsqueda deben evitar la complejidad y priorizar los criterios de búsqueda por los parámetros más utilizados antes que abundar en parámetros de escaso uso que por lo regular usan campos no indexados.
  1. **Implementación en capa de negocio**
* Las implementaciones que usen patrón de diseño "service façade" deben estar orientados a atender un proceso/subproceso
* Las implementaciones que usen patrón de diseño "session façade" orientados a atender actividad y tareas del negocio del proyecto.
* Las realizaciones del patrón de diseño "session façade" deben implementarse a través de interfaces locales y deben reusadas a través de otros componentes o "service façade".
* Las implementaciones que usen el patrón de diseño "data access object" donde intervienen múltiples entidades persistentes deben ser agrupadas según la actividad y/o contexto de información.
* Las transacciones negocio debe establecerse un número máximo de reintento de recuperación ante la no disponibilidad de un servicio y/o recurso del cual depende.
* Las clases de los modelo de entidades persistentes deben ser independientes a las clases del modelo de entidades de negocio de manera que se pueda realizar la separación de binarios en los casos que se requiera sin que se cree una dependencia circular entre binarios.
* Las interfaces locales de las implementaciones del patrón Service Facade deben ser heredadas de las interfaces remotas.
* Las firmas de los métodos definidas en las interfaces de las clases que implementen el patrón “Service Facade” y “Session Facade” deben hacer uso de las clases que implementen el patrón DTO, estos métodos no deben hacer uso de las clases que implementen JPA.
* Las implementaciones de procesamiento asíncrono deben realizarse a través de técnicas de colas de mensajes (JMS).

* 1. **Implementación en capa de presentación**
* En las implementaciones de formularios web debe maximizarse el uso de código html, javascript y/o librerías/frameworks javascript haciendo un uso adecuado de los componentes visuales JSF.
* En las aplicaciones web las validaciones de tipos de datos debe realizarse en el lado cliente a través de utilitarios librerias y/o framework javascript.
* Las implementaciones realizadas en los JSF Managed Bean deben estar relacionados proporcionar datos y/o controlar el comportamiento de los formularios según la interacción de la aplicación.
* Los recursos de tipo imagen deben corresponder en tamaño de bytes al tamaño de presentación de la misma, ejemplo no debe realizarse ajustes de una imagen de 48x48 pixeles para ser mostrados como 16x16 pixeles, para estos casos crear la misma imagen en formato 16x16 pixeles.
* Las imágenes deben usar la resolución mínima que permita ser vista con buena calidad y maximizar el uso de formatos ligeros como gif y jpeg, en los casos que corresponda se deberá usar el formato png.
* Los recursos del tipo javascript y hoja de estilos deben ser desarrollados y/o iterados para que estos tengan la menor capacidad de bytes posibles.
* Las utilidades javascript comunes a los formularios de aplicaciones deben ser centralizadas como recursos estáticos en archivos \*.js y referenciados por las páginas que lo requieran, está permitido utilitarios javascript embebidos en los formularios web cuando su utilidad está limitado al formulario.
* Las técnicas de paginación debe realizarse en los clientes web a partir de un conjunto de datos que pueden ser proporcionados por los servicios en formato JSON o XML.
* En los casos que corresponda se debe manejar el resultado de los formularios buscadores con técnicas de paginación implementadas en la capa de negocio y/o capa de datos.
* Los textos a visualizar en las páginas web (etiquetas, ayudas, etc.) deben ser obtenidas de archivos de propiedades considerando los caracteres especiales sean escritos con el código Unicode que le corresponda.
* En implementaciones JSF, los Managed Bean para el formularios buscadores deben ser de alcance "@RequestScope" y en excepciones y por paginación de datos se requiera extender a "@ViewScope".
* En implementaciones JSF, se debe maximizar el uso de Managed Bean de alcance “@RequestScope” de requerir el alcance de “@SessionScope” se debe implementar los mecanismos para eliminarlos cuando ya no sean requeridos.
* Evitar la duplicidad de código en las vistas o formularios web, para eso maximizar el uso de tags que permitan la reusabilidad en las diferentes vistas.
* Las implementaciones para uso de JSF composite deben ser diseñados con el patrón delegate, de manera que la clase consumidora del JSF composite implemente la interfaz que requiera el componte.
* Los formularios web de tipo buscadores de uso público deben implementar técnicas para diferenciar computadoras de humanos (ejemplo captcha) para evitar el uso automatizado de estos formularios.
* Los controles de los formularios web deben tener definido su "id" y "nombre", estos valores no deben ser generados de manera aleatoria por el uso de algún framework.
* Las validaciones de tipos de datos, obligatoriedad (requerido), rangos de valores deben realizarse en los navegadores clientes de manera que se minimice la comunicación con los servidores web y/o servidores de aplicaciones.

1. **Aplicaciones compuestas**
   1. **Generales**

* Los servicios a reutilizar en más de un proyecto de sistemas de información deben serán implementados y desplegados en una plataforma SOA.
* Los servicios de alcance local a un sistema de información no debe ser expuesto a través de una infraestructura SOA y deben ser implementados dentro de la plataforma tecnología que este utilice.
* Los indicadores de desempeño del servicio debe ser los mínimos requeridos y estructurados a medida de los reportes que se quieran generar con ellos de manera que no se genere sobrecarga de trabajo en la escritura y lectura de su fuente de datos.
* Implementar rutinas automatizadas que permita que los valores de desempeño de servicios puedan ser purgado de la fuente de datos donde son almacenados.
* Los registros de eventos de infraestructura se debe mantener separado a los registros de eventos de aplicación compuesta.
* Las aplicaciones compuestas deben ser utilizadas cuando se quiere crear un nuevo servicio en base a los servicios y/o funcionalidades de sistemas ya disponibles.
* La orquestación de servicios debe estar orientada a implementar requerimientos técnicos más que requerimientos de negocio.
* El procesamiento masivo de datos debe ser externalizada a la plataforma SOA pudiendo para ello usar la plataforma de servidores de aplicación.
  1. **Liberación de aplicaciones**
* Los servicios deben ser versionados y redesplegados mediante mecanismos que permita que la versión anterior termine de procesar las instancias creadas con ellas.
* Todos servicio en una plataforma SOA debe ser como mínimo virtualizado en el ESB de la plataforma.
  1. **Implementación en capa de datos**
* Se debe maximizar el uso del ESB de la plataforma SOA para la transformación de datos previos al consumo de un servicio que no requieran lógica de negocio y no requiera un entorno transaccional.
* Las aplicaciones compuestas deben usar recursos XSLT para realizar transformaciones de datos complejos y/o reiterados en múltiples aplicaciones compuestas.
  1. **Implementación en capa de actividades de negocio**
* Las reglas de negocio deben ser centralizadas y claramente definidas a partir del modelo conceptual de datos de los procesos de negocio.
  1. **Implementación en capa de procesos de negocio**
* Usar la aplicación compuesta para orquestar múltiples servicios internos o externos potencialmente asíncronos con periodo extenso de ejecución y/o en un entorno transaccional.
  1. **Implementación en capa de presentación**
* En los casos de que la funcionalidad del servicio sea expuesta de manera asíncrona se debe controlar la unicidad del mensaje de manera que un mismo “request” no sea atendido muchas veces.