<LA EMPRESA>

Reglas de diseño de Arquitectura de Aplicaciones(GENERAL)

Aprobación del Documento

Nombre	Cargo	Fecha

Control de Versión de la Plantilla

Control de Versiones

Historial de Cambios

Versión	Autor	Revisado/Aprobado por:	Fecha

Cambios

Versión	Cambios

Referencias

Ref. No.	Nombre Documento	Ubicación

Fuentes de Información

Descripción	Entidad

Definiciones, Acrónimos y Abreviaturas

Abreviatura / Acrónimo	Definición
MVC	Model view Controller
ORM	Object Relational Mapping

ÍNDICE DE CONTENIDOS

1 Motivo del documento

El presente documento describe las reglas generales de diseño que debe seguir una Aplicación de Software creada para <LA EMPRESA> por un proveedor externo. <LA EMPRESA> no aceptará la entrega de una aplicación si no sigue estrictamente las reglas presentadas en este documento.

2 Introducción

En este documento vamos a exponer las diferentes reglas que **deben cumplirse** en un desarrollo y diseño de arquitectura de una aplicación creada por un proveedor para <LA EMPRESA>. Estas directrices van orientadas a valorar herramientas, patrones, etc. que podremos utilizar para una unificación de conceptos y una mejora en la calidad de entrega final del código fuente y en el mantenimiento de las aplicaciones.

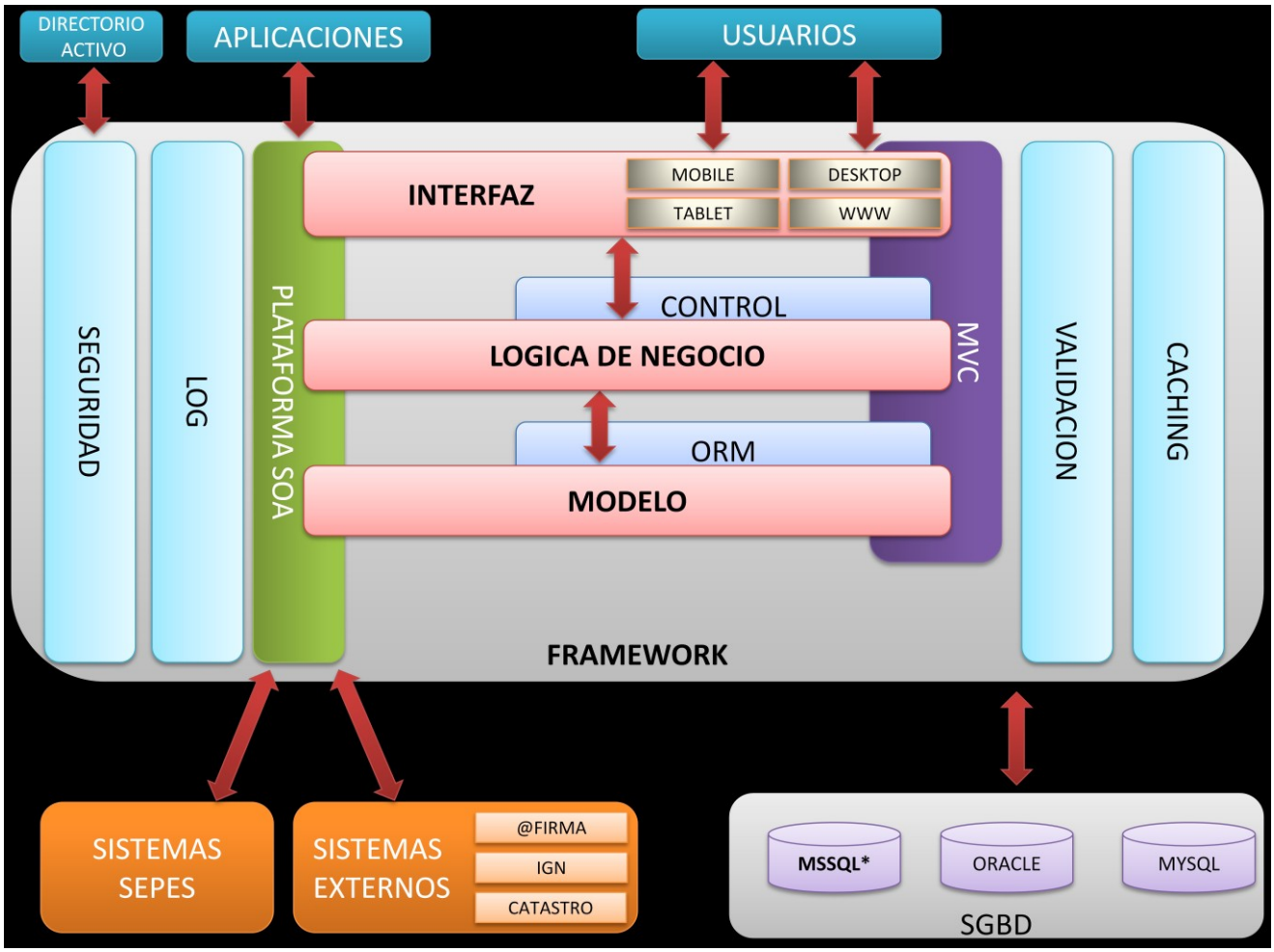

Figura 1

A continuación en las siguientes secciones se describirán todos los componentes representados en la Figura 1.

3 PRINCIPIO DE SEPARACIÓN EN TRES CAPAS:

Siguiendo los principios del patrón de software MVC la codificación de una aplicación en <LA EMPRESA> se realizará separando la capa de INTERFAZ, LÓGICA DE NEGOCIO y MODELO. Todo ello apoyado bajo un framework que de soporte a este patrón de diseño de software.

- Las **interfaces** pueden dar soporte tanto a usuarios desde interfaces móviles, desktop o www. Por otro lado también ofrecen la posibilidad de interoperar con otras aplicaciones creando interfaces de conexión en la plataforma SOA de <LA EMPRESA>.
- La **lógica de negocio** debe contener las clases que implementen las reglas de negocio de <LA EMPRESA> para la aplicación determinada. Son el núcleo sobre el cual se realizarán la mayoría de las pruebas unitarias para la garantía calidad.
- La capa **modelo** debe contener las entidades que principalmente serán utilizaras para la gestión de su almacenamiento persistente.

Se debe tener en consideración que en el desarrollo se debe parametrizar todo lo que sea posible y debe ser indicado en un fichero de configuración y/o BBDD.

También se debe utilizar los ficheros de recursos y localizarlos (por cultura: gl-ES, ca-ES, eu-ES, en-GB..) si es necesario.

3.1 CAPA INTERFAZ

- El diseño de las interfaces gráficas (Desktop, www, Tablet, Informes, etc..) deben seguir el manual de identidad corporativa y los principios de diseño de interfaces de <LA EMPRESA>.
- Toda interfaz web desarrollada debe cumplir las normas de accesibilidad W3C AA y debe adaptarse a la “localización” del usuario (múltiples lenguajes).
- Todo Layout que se realice en cualquier aplicación debe poder adaptarse a la resolución de la pantalla.
- Cualquier interfaz de escritorio para aplicaciones internas de la intranet de <LA EMPRESA> debe cumplir normas básicas de accesibilidad (acordadas previamente con la unidad de Desarrollo e Integración) para los empleados que utilicen las aplicaciones relativas a las habilidades: visión, movilidad, audición, aprendizaje, lenguaje y edad. **
- Para interfaces de autómatas (Servicios web, u otro tipo de servicios) se deberán seguir la normativa de notación de codificación e inclusión dentro de la plataforma SOA.
- Debe realizarse la validación de cualquier dato existente en la interfaz antes de ser procesado por la capa lógica de negocio. Las notificaciones de error deben ser claras y directas para su posterior comprensión y corrección.
- Todos los errores mostrados deben contener mensajes legibles y fácilmente entendibles para el usuario al que van destinados.
- Todas las clases de interfaz deberán almacenarse en:
    - <LA EMPRESA>.[Proyecto].[Módulo].UI

3.2 CAPA LÓGICA DE NEGOCIO

- Gracias a la separación lógica en tres capas pueden implementarse varias interfaces (móvil, www, desktop) que utilicen la misma lógica de negocio.
- En la documentación de análisis y diseño debe quedar bien clara la lógica implementada y la trazabilidad de la misma, desde el método → clase → diseño → análisis → caso de uso.
- Debe contemplarse el control de acceso y log sobre las clases que contienen estos métodos.
- Se recomienda el control de instancias mediante el uso del patrón Singleton. Así como el uso en general de otros patrones descritos en GoF.
- En entornos cliente-servidor debe de realizarse la comprobación de la validación de los datos de entrada que provienen de la interfaz. Esta comprobación también ha de contemplar posibles ataques XSS o SQL Injection.
- Todas las clases deben almacenarse en el espacio de nombres:
  - <LA EMPRESA>.[Proyecto].[Módulo].BL

3.3 CAPA MODELO

- En esta capa se deben desarrollar las clases de entidad que correspondan con el modelo de gestión interno de datos tanto para BBDD, Ficheros como envío de información a servicios alojados en la plataforma de interoperabilidad SOA de <LA EMPRESA>.
- Se deja en función del tipo de proyecto la decisión de uso o no de un ORM, siempre consensuada con la unidad de Desarrollo e Integración.
- La lógica de negocio consume las clases definidas en la capa Modelo. Estas clases deben quedar almacenadas en dos paquetes fundamentales del proyecto:
  - DataLayer: <LA EMPRESA>.[Proyecto].[Módulo].Model.DL
  - Entities: <LA EMPRESA>.[Proyecto].[Módulo].Model.Entities
- En principio cualquier trabajo con BBDD debe realizase:
  - A través de las clases del espacio de nombres DataLayer.
    - Estas clases invocarán utilizando el framework que de soporte a trabajo con multiples sistemas BBDD (ej:EntityFramework con implementación de patrones de diseño tipo Abstract Factory).
  - Utilizando procedimientos almacenados para Guardar, Eliminar, Actualizar o Recuperar cualquier dato. Estos procedimientos han de seguir la notación definida en el documento “Normativa de Notación de Código Fuente”.
  - Cualquier acción sobre la BBDD debe quedar registrada en el LOG.
  - En el DataLayer debe contemplarse la posibilidad de realizar cache sobre determinados métodos que lo requieran dado su peso / frecuencia / prioridad, etc.
  - Para evitar SQL Injection se recomienda el uso de procedimientos almacenados y SQL parametrizable que realice la validación de los parámetros enviados a los comandos del sistema SGBD.
- Las entidades deben validar los datos que contienen a la hora de ser asignados.

4 CAPAS DE SOPORTE VERTICAL

Como soporte a todas las capas descritas anteriormente, se deberán contemplar los siguientes pilares:

4.1 REGISTRO DE ACCIONES - LOG

- Se debe registrar los accesos que realizan los usuarios a las aplicaciones.
- Las operaciones de relevancia en la aplicación se debe registrar en el modulo de log.
- Los procesos de base de datos, es decir procedimientos almacenados, procesos planificados, etc. se deben quedar almacenados en la base de datos por medio del log.
- Todas las entradas que se hagan mediante el modulo de log se debe quedar registrado en un sistema gestor de base de datos.
- En el tratamiento de excepciones que se producen en la aplicación debe quedarse registrado en el modulo de log.
- Debe poderse extraer del LOG de forma legible trazas de aplicación útiles, separadas por usuario (si procede) en diferentes niveles (DEBUG, WARN, INFO, ERROR, FATAL), módulos de aplicación y componentes (interfaces, clases de negocio, datalayer, …).
- Sistema de alertas: conectado con el LOG, la aplicación debe notificar aquellos sucesos críticos que requieran de intervención de un administrador de aplicación.

4.2 SEGURIDAD

- Todos los usuarios de la aplicación deben tener una relación con LDAP, la autenticación debe realizarse de forma transparente (sin ventanas de usuario/clave) a través del directorio activo de <LA EMPRESA>. La asignación de permisos debe poderse agrupar en ROLES que posteriormente pueden ser asignados a los usuarios. También debe existir la posibilidad de agrupar los usuarios por grupos y dar permisos y roles a esos grupos. (Salvo.. excepciones ver pfto. Final).
- Caducidad de sesión y autorizaciones temporales: Las sesiones que se adjudican a los usuarios para acceder a las aplicaciones deben tener un periodo de caducidad establecido. Por otro lado las autorizaciones temporales a usuarios que acceden a la aplicación tendrán periodos de renovación en el tiempo de vida del uso de la aplicación en la sesión que le ha sido autorizada.
- Control de acceso y registro: Se debe regular el control de acceso a varios niveles:
  - Desde la interfaz restringiendo en base a roles lo que el usuario se le debe mostrar en la pantalla.
  - Desde la lógica de negocio, en base a rol que posea el usuario deberemos controlar la ejecución de ciertos métodos de la aplicación.
  - A nivel de base de datos en base al uso de esquemas para controlar el acceso a determinadas tablas, procedimientos almacenados, vistas, etc.
- Seguridad comunicaciones: siempre se requerirá si el canal de comunicación lo desea. Toda información sensible deberá ir protegida según los principios básicos de integridad, confidencialidad y no repudio. (Cifrado y firma electrónica).
- Seguridad en BBDD: La empresa de desarrollo nunca tendrá acceso a la BBDD de producción. Las claves de aplicación deberán cambiarse de forma periódica. Los usuarios de desarrollo nunca podrán acceder a la aplicación que han creado en producción, salvo para intervenciones que se requieran con

autorización requerida (según protocolo). El contenido sensible de la información de BBDD que no debe ser visto por ningún usuario administrador deberá ir codificada (por funciones HASH SHA).

- Para evitar SQL INJECTION se deberá utilizar SQL parametrizable proporcionado por el framework. Este apartado deberá apoyarse junto con la capa vertical de VALIDACIÓN.
- XSS: En aplicaciones web el framework deberá contemplar los principios de seguridad para evitar ataques XSS. Este apartado deberá apoyarse junto con la capa vertical de VALIDACIÓN.
- Para aplicaciones que requieran la implementación de un formulario de acceso (fuera del control de dominio): Para evitar BOTs automáticos de ataque por fuerza bruta, en los formularios de acceso se deberá incluir tras tres intentos fallidos un control CAPTCHA.
- Salvo previa autorización del administrador de aplicación, debe incluirse la posibilidad de bloquear por IP aquellos intentos de acceso fallido realizados de forma repetitiva.

4.3 VALIDACIÓN

- Se debe comprobar que una aplicación funciona correctamente validando cada componente de entrada de la interfaz de usuario. (Cajas de texto, datepickers, listas, grids, combos, etc..)
- Toda información recibida en la capa de negocio debe ser validada previamente por la interfaz que la consume.
- En entornos cliente servidor, la información ha de ser validada en ambos extremos (tanto en cliente como en servidor).
- Según se ha indicado en las capas principales, para la integridad y seguridad del sistema se debe validar en las clases de entidad las posibles reglas de validación que garanticen la integridad de la información albergada en las propiedades.

4.4 USO DE CACHE

- Utilizar para mejorar el rendimiento de la aplicación.
- La recomendación es almacenar los datos los más cerca posible del punto de aplicación: en la capa de presentación si se emplean para controles, en la capa de lógica si afectan a objetos de negocio y en la capa de datos sin afectan a los parámetros de almacenados.
- Se deberán guardar los datos que tienen que estar disponibles para todos los usuarios y que además no cambien frecuentemente, también los datos cuya obtención sea muy costosa así podremos evitar llamadas constantes a las bases de datos y la transmisión de información de red masiva.
- En caché, solo se admite almacenar información sensible si se encuentra protegida (cifrada).
- Los datos cuyo uso es frecuente se almacenarán al iniciar la aplicación. Los datos de uso más esporádico se almacenará la primera vez que sean solicitados.

5 SISTEMA DE INTELIGENCIA DE NEGOCIOS

Salvo excepciones acordadas, las aplicaciones nuevas desarrolladas pueden albergar los informes en la plataforma de inteligencia de negocios de <LA EMPRESA>.

Por otro lado cualquier información aportada por una aplicación nueva ha de ser susceptible de ser trasladada al STAGING AREA, si esta es requerida por cualquier informe y/o cuadro de mando alojado en **la plataforma de BI de <LA EMPRESA>**.

La plataforma sigue la arquitectura mostrada en la figura 2. El sistema de BI posee un DataWareHouse cuya fuente de información proviene de los procesos ETL (Extracción / Transformación y Carga) implementados desde el STAGING AREA. Todos los informes de la plataforma BI son implementados sobre la plataforma SAP-BO.

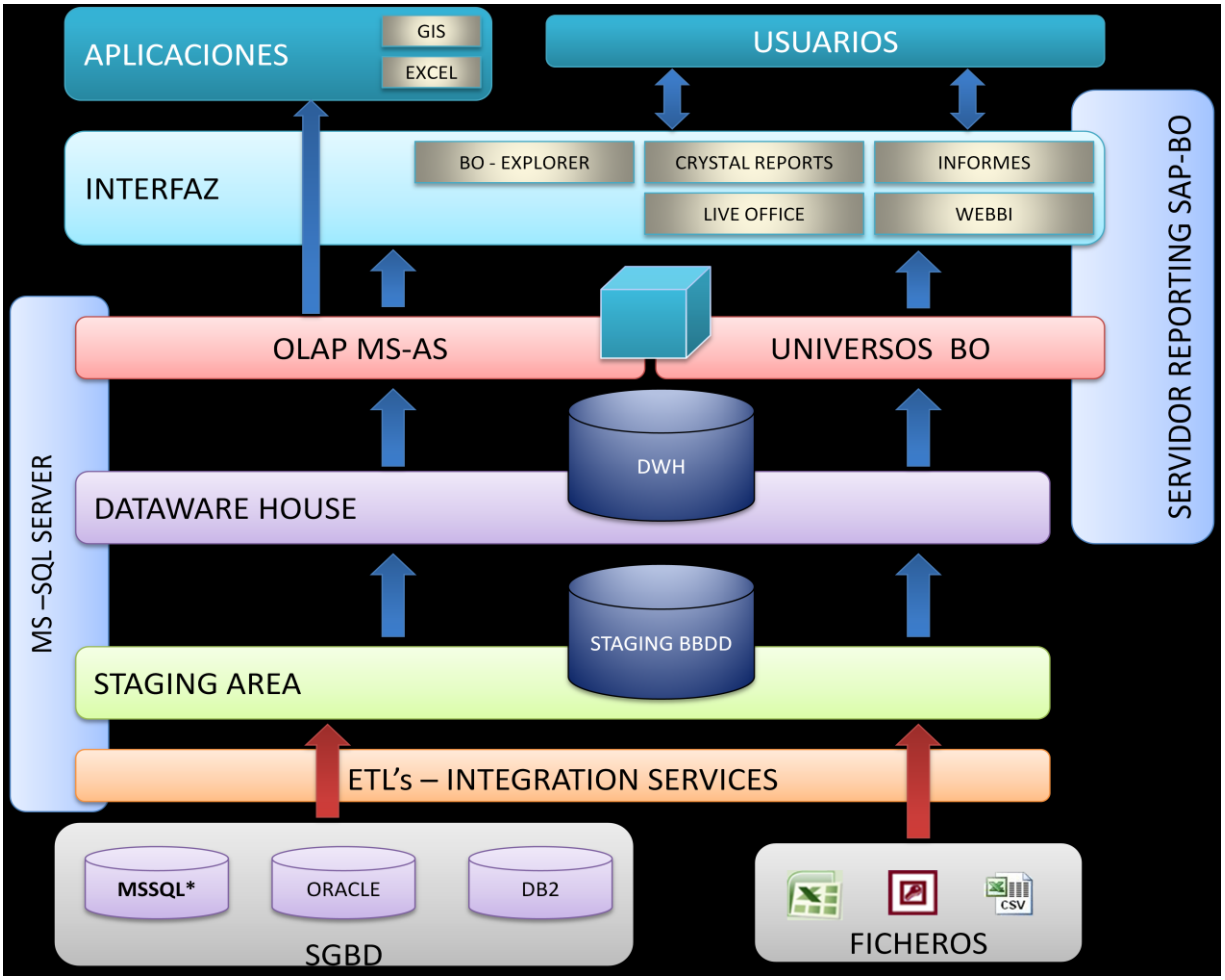

Figura 2

Por otro lado, cualquier nueva aplicación puede consumir los datos del DATAWAREHOUSE de <LA EMPRESA> utilizando los CUBOS OLAP almacenados en MS-Analysis Services.

6 <LA EMPRESA> NO PERMITE

Sin restricción en alcance, de cualquier otra falta o defecto descrito en los documentos y normativas de la metodología de <LA EMPRESA>:

1- Qué no se siga la metodología de gestión de proyectos, desarrollo y mantenimiento de aplicaciones de <LA EMPRESA>.
2- Qué no se siga la normativa de codificación y la de diseño de aplicaciones de <LA EMPRESA> planteada en este documento.
3- Que no se siga la normativa de desarrollo interno para aquellos proyectos que deben seguirla.
4- La entrega de un proyecto sin una documentación conforme a las plantillas exigidas por la Metodología de <LA EMPRESA>.
5- Entrega de aplicaciones no accesibles que no sigan en el diseño de su interfaz el manual de identidad corporativa.
6- Qué existan fallos de seguridad. Plantear arquitecturas con fallos de seguridad, inclusión ad-hoc de contraseñas, ficheros sin proteger, claves y datos sensibles accesibles desde bases de datos, acceso a métodos protegidos, XSS, SQL Injection, etc..
7- No entregar código sin comentarios: siempre se debe presentar el código documentado, con comentarios sobre las clases y métodos que describan la utilidad y uso.
8- Qué el código fuente no se encuentre bajo revisión del servidor TFS de <LA EMPRESA>. Siempre con la última versión actualizada. Se han de seguir todos los procedimientos descritos en el “Manual de procedimientos TFS”
9- Qué las cadenas de conexión se encuentren “ad-hoc” en código: Qué todo aquello susceptible de ser parametrizado se encuentre almacenado en un fichero de configuración y/o tabla de base de datos: (Rutas de archivos, Nombres de servidor, Urls, Etc)
10- Cuando se genere una excepción no realizar un tratamiento en su justa medida de los motivos y posibles soluciones que se debe llevar a cabo.
11- Utilizar librerías o componentes propios de una empresa, no reconocidos, ni evaluados, ni consentidos previamente por <LA EMPRESA>.
12- Que no se desarrolle siguiendo los procedimientos generales de transferencia entre entornos para promoción y corrección del proyecto en diferentes estadios.
13- Que para cualquier cambio que deba ser publicado en producción no siga el “manual de procedimiento de despliegue de nuevas actualizaciones en el entorno de producción”.
14- <LA EMPRESA> no admite que el proveedor no disponga de recursos con una disponibilidad adecuada y capacidad para la resolución de cualquier tarea relacionada con el proyecto en horario de oficina de <LA EMPRESA> (en especial en caso de empresas que externalicen su desarrollo a países con diferentes husos horarios).