Normas para el Desarrollo

|  |  |
| --- | --- |
| **Fecha** | 31/01/2015 |
| **Modificado por** | Alex Jiménez |
| **Versión** | 2.5 |

## Introducción

En este documento se detallan tanto normas técnicas de obligado cumplimiento como convenciones de conceptos, nomenclatura y comentado del código fuente.

Las normas y buenas prácticas descritas en este documento deberán cumplirse para la aprobación y puesta en producción de un nuevo desarrollo en la plataforma descrita en la tabla inicial.

En el caso de producirse una actualización del documento se facilitará de forma inmediata a todos aquellos involucrados en el desarrollo de nuevas funcionalidades y modificaciones sobre la plataforma mencionada. A partir de ese momento cualquier versión previa del documento quedará invalidada y los desarrollos deberán ajustarse al documento más reciente.

## Objetivos

El objetivo de éste documento es asegurar los puntos siguientes:

* 1. Intentar alterar los estándares de las plataformas lo mínimo posible asegurando las funcionalidades que se requieran.
  2. Cuando no haya otra opción que realizar un desarrollo, cumplir Best-Practices de la plataforma en concreto además de las normas adicionales pertinentes que aseguren un impacto mínimo sobre el sistema y no compliquen ni imposibiliten futuras ampliaciones y modificaciones.
  3. Mantener la mayor coherencia posible a lo largo del sistema, tanto a nivel técnico como a nivel de interfaz intentando ajustarse siempre a las guías de diseño de la plataforma.
  4. Establecer convenciones de codificación y nomenclatura que se extiendan a lo largo de todas las plataformas involucradas para evitar al máximo las posibles confusiones derivadas.

## Entornos de plataforma

En caso de que la plataforma disponga de varios entornos, se deberá programar cualquier desarrollo en el entorno más alejado de Producción posible, es decir si disponemos por ejemplo de tres entornos que respondan a las siguientes funciones:

1. **Desarrollo**
2. **Test**
3. **Producción**

Cualquier nueva funcionalidad deberá ser desarrollada siempre y sin excepción en el entorno de **Desarrollo** ya que es el de menor repercusión funcional en caso de error, así como también deberán crearse en éste entorno las etiquetas y complementos necesarios para la traducción de los nuevos elementos creados y traspasarse después a los entornos superiores, validando las pruebas propuestas para el desarrollo en cada uno de los entornos.

## Desarrollo en capas y modelos

Si la aplicación dispone de capas en las que estructurar los desarrollos, se acordará la capa en la que desarrollar con el arquitecto de la plataforma antes de iniciar cualquier tipo de desarrollo. Los desarrollos a realizar se deberán programar siempre en la capa de aplicación acordada. Actualizaciones de la plataforma provenientes del proveedor habitual o un tercero, hotfixes y parches que no provengan de desarrollo propio se tratarán como casos excepcionales y deberán ser estudiados y analizados de forma independiente.

Si existiera uno o más niveles de clasificación del código por debajo del inicial (capas) como pudieran ser modelos de desarrollo, se establecerá un modelo de desarrollo por empresa desarrolladora, por ejemplo la empresa ZZZ desarrollaría en la capa VAR incluyendo todos sus desarrollos en el modelo VAR\_ZZZ.

Los modelos desarrollados únicamente se incorporarán en el entorno de Test y Producción bajo conformidad del arquitecto de la plataforma. Con el objetivo de dar su conformidad el arquitecto deberá recibir el modelo a importar en formato de archivo de importación para su correcta catalogación/versionado y una copia del log de compilación final con el control de Best-Practices de la plataforma activado si existiera.

## Nomenclatura

Un objeto nuevo en el sistema deberá ser nombrado con el siguiente formato:

**SYP + NombreCompuestoInicialesEnMayusculas**

En caso de que el objeto contenga a su vez otros objetos, como por ejemplo una clase nueva conteniendo métodos o una tabla nueva conteniendo campos, los elementos contenidos no deberán incluir el prefijo **SYP**.

En el caso especial de los proyectos, si se refieren a un número de GAP concreto y documentado, deberán nombrarse de la forma siguiente:

**SYP\_ + [Año en cuatro dígitos] + [Mes en dos dígitos] + [Día en dos dígitos] + \_ + [Id. de GAP]**

Si no se refieren a un documento de GAP concreto, la nomenclatura será la siguiente:

**SYP\_ + [Año en cuatro dígitos] + [Mes en dos dígitos] + [Día en dos dígitos] + \_ + [NombreCompuestoInicialesEnMayusculas]**

Cuando se extienda de un objeto previamente existente se deberán seguir las normas anteriores añadiendo el sufijo **Ext** al objeto en cuestión:

**SYP + [NombreCompuestoInicialesEnMayusculas] + Ext**

En el caso de los formularios que utilicen un origen de datos específico y claramente definido, su nombre deberá ser exactamente el mismo que el de la tabla utilizada en el origen de datos principal.

Siempre se intentará que los nombres utilizados sean lo más explícitos y descriptivos posibles y únicamente se utilizará el idioma inglés para nombrarlos.

En el caso de que la plataforma disponga de secuencias numéricas que configurar, éstas seguirán el formato siguiente:

**[Código de compañía en 3 dígitos] + [Descripción de la secuencia en máximo de 10 dígitos]**

**017 + DIMODCUS (Diario de modificaciones masivas de clientes)**

A partir de este punto y a lo largo del documento se hará referencia a dos conceptos diferenciados, Consulta e Informe.

Se entenderá como **Consulta** una de las dos opciones siguientes, según contexto:

1. ***Un origen de datos destinado a alimentar la información de una vista o formulario de la plataforma.***
2. ***Una representación de datos visual dentro de un formulario propio de la plataforma, permitiendo o no la modificación de los mismos.***

A su vez, se entenderá como **Informe**:

1. **Una representación de datos visual en formato imprimible / exportable a otra aplicación que mayoritariamente tendrá como objetivo estar adecuadamente formateado para su impresión en un formato de documento estático (pdf) o físico (papel).**

## Creación/Modificación de objetos

### Seguridad

Se debe asegurar que cualquier objeto de nueva creación disponga de los permisos adecuados para poder asignarlos a un rol de seguridad específico, para este objetivo se repasará siempre que cualquier elemento que se cree esté incluido en un privilegio, ya sea de nueva creación o estándar del sistema. Se debe asegurar que la descripción del privilegio es lo suficientemente explícita para sobreentender que incluye la funcionalidad adaptada.

### Proyectos

Cualquier desarrollo que se realice en el sistema deberá estar incluido en un proyecto / solución según la plataforma. No se permitirá migrar nunca objetos de forma independiente entre entornos.

### Tablas

Se dará especial importancia a agrupar todos los campos de una tabla en agrupaciones que puedan ser utilizadas tanto en formularios como informes y procurar que ningún campo quede aislado sin pertenecer a ninguna agrupación.

Las propiedades de una tabla que afecten a la visualización de sus datos representados en informes, vistas, formularios, etc… no se dejarán nunca en blanco para mantener la coherencia en la presentación de los datos al usuario final.

Todos los campos de nueva creación de una tabla utilizarán un tipo de datos extendido en caso de estar presente en la plataforma, incluso los de tipo enumerado. En caso de no disponer del tipo de datos extendido requerido en el estándar, los crearemos siguiendo las especificaciones de cada uno.

Si se requiere añadir una relación de navegación a una tabla (con el objetivo de aprovechar el sistema de *auto-join* de la plataforma, si aplica) Se requerirá que la relación no se tenga en cuenta en la estructura de la BBDD interna mediante la propiedad adecuada de la plataforma.

Si se crean tablas nuevas con estructura de cabecera/líneas, se deberá asegurar que la tabla de líneas está correctamente relacionada con la tabla de cabecera y que ésta, a su vez, contiene las acciones necesarias de borrado en cascada para asegurar la integridad de los datos almacenados.

Las tablas de sistema no se modificarán excepto necesidad excepcional. Si se hace necesario añadir campos o métodos a una tabla estándar se creará una nueva tabla vinculada a la original mediante el identificador de registro único pertinente de la plataforma y con las mismas propiedades de auditoría y guardado de registros por compañía que la original. A éstas tablas nuevas las llamaremos **Tablas Extendidas** y como mínimo les crearemos los campos, relaciones y métodos siguientes.

|  |  |  |
| --- | --- | --- |
| **Tipo de objeto** | **Nombre** | **Objetivo** |
| Campo | RefRecId | Vincular la tabla extendida con su original (Corresponde al RecId de la tabla original) |
| Campo | RefTableId | Vincular la tabla extendida con su original (Corresponde al TableId de la tabla original) |
| Relación | Master | Vincular la tabla extendida con su original (Utiliza RefTableId + RefRecId) |
| Método | FindOrig | Devolverá el registro de la tabla original al que hace referencia. |
| Método | Find (static) | Dado un RecId de la tabla original, devolverá el registro vinculado de la tabla extendida |

### Tipos de datos extendidos

Los tipos de datos extendidos del sistema no se modificarán excepto necesidad excepcional, en su defecto se heredará un nuevo tipo de datos y se modificará para adecuarse a la necesidad.

En caso de que el nuevo tipo de datos extendido actúe como índice de una tabla de nueva creación se requerirá crear la relación interna del tipo de datos extendido con la tabla.

### Enumerados

Los enumerados del sistema no se modificarán excepto necesidad excepcional, en su defecto se copiará el enumerado requerido y se modificarán sus valores en la copia realizada.

Como norma general, no se eliminarán valores de un enumerado aunque se dejen de utilizar, se guardarán como valores de histórico y se aplicarán las transformaciones necesarias a los registros que tengan asignado tales valores.

### Vistas

Las vistas del sistema no se modificarán excepto necesidad excepcional, en su defecto se puede copiar la vista requerida y realizar modificaciones sobre la copia.

Las propiedades de una tabla que afecten a la visualización de sus datos representados en informes, vistas, formularios, etc… no se dejarán nunca en blanco para mantener la coherencia en la presentación de los datos al usuario final.

### Consultas

Las consultas del sistema no se modificarán excepto necesidad excepcional, en su defecto se copiará la consulta requerida y se modificará la copia realizada.

### Formularios

Si está definido como buena práctica en la plataforma, Los formularios que muestren sus datos en forma de tabla deberán contener un mínimo de dos páginas organizadas en pestañas, siendo la segunda una visualización de la tabla en forma de grupos de campos con el mismo contenido.

Se deberá dedicar un esfuerzo especial a que los diseños, orígenes de datos, agrupaciones de campos y campos de un formulario mantengan sus propiedades por omisión siempre que sea posible.

Si se deben añadir botones al formulario que ejecuten una clase o abran otro objeto de la plataforma que pueda ser accesible a través de un Elemento de Menú se utilizará (o creará si no existe) el Elemento de Menú necesario para gestionar la llamada al objeto desde el formulario.

La prioridad de controles a utilizar en un formulario será la siguiente:

1. **Grupos de campos de tabla con la propiedad AutoDataGroup establecida en Yes.**
2. **Grupos de campos de tabla con la propiedad AutoDataGroup establecida en No.**
3. **Campos de tabla.**
4. **Métodos Display o Edit del origen de datos creados en la tabla.**
5. **Métodos Display o Edit del origen de datos creados en el formulario.**
6. **Controles basados en tipos de datos extendidos.**

Todos los formularios deben tener la habilidad de ser redimensionados sin perder la coherencia en su contenido.

### Informes

Los informes desarrollados en la plataforma, si se crean mediante herramientas externas o AddIns de terceros, deberán tener como mínimo el mismo rendimiento que tendrían en caso de haberse realizado con la herramienta nativa, además de asegurar un impacto mínimo de la plataforma y una extensibilidad adecuada para sus futuras ampliaciones y/o modificaciones.

### Clases

Las clases del sistema no se modificarán excepto necesidad excepcional, en su defecto se heredará una nueva clase y se modificará para adecuarse a la necesidad.

Cualquier clase desarrollada en la plataforma deberá adecuarse a uno de los tres modelos especificados a continuación:

1. **Una clase que representa un objeto.**
2. **Una clase de acciones.**
3. **Una clase de soporte.**

### Elementos de menú

Los Elementos de Menú del sistema no se modificarán excepto necesidad excepcional, en su defecto se copiará el Elemento de Menú requerido y se modificará la copia realizada.

### Trabajos (Scripts)

Un Trabajo o Script estático nunca deberá ser parte necesaria para el circuito de un proyecto, sólo se aceptarán Trabajos que sirvan a modo de procesos de prueba, creación de registros para una configuración inicial, etc… pero cualquier código necesario para el funcionamiento del proyecto deberá estar incluido o bien en una Clase o en un objeto nativo de la plataforma como método.

### Secuencias Numéricas

Si la plataforma dispone de diferentes tipos de secuencia numérica, los usaremos con los objetivos siguientes:

##### 4.13.1. Secuencias manuales

Configuraremos como manuales todas aquellas secuencias que no requieran ser indicativas de un orden concreto.

##### 4.13.2. Secuencias continuas sin recuperación de valores

Configuraremos como secuencias continuas sin recuperación de valores aquellas secuencias que requieran ser indicativas de un orden concreto pero en las cuales, si se produce un vacío entre sus valores, no supone un problema para los requerimientos del negocio y/o la plataforma.

##### 4.13.3. Secuencias continuas con recuperación de valores

Configuraremos como secuencias continuas con recuperación de valores aquellas secuencias que requieran ser indicativas de un orden concreto y en las cuales no puede existir un vacío entre sus valores.

### Etiquetas

##### Archivo de etiquetas

Si la plataforma dispone de catalogación de etiquetas mediante archivos, todas las etiquetas de nueva creación se crearán en el fichero de etiquetas propio de Soler&Palau **@SYPxxxxx**. Nunca se utilizará otro archivo de etiquetas.

##### Uso de las etiquetas

Nunca se utilizarán literales escritos directamente en una propiedad de objeto, en el código fuente o en cualquier otro punto del entorno de desarrollo de la plataforma.

##### Ubicación de las etiquetas

Las etiquetas de campos de datos se asignarán a los objetos con el orden de prioridad siguiente:

1. **Campos de tabla.**
2. **Campos de origen de datos de un formulario / Informe.**
3. **Campos de datos del propio formulario.**

## Ubicación del código fuente

La ubicación del código fuente se presenta a continuación en orden de prioridad. Sólo bajaremos el nivel de prioridad si, por el objetivo del código a desarrollar, no es posible programarlo en un nivel superior.

**Hay que tener presente que sólo estará permitido incorporar código con lógica de negocio en tablas y clases, dejando el objetivo del código en otros objetos para la lógica de interfaz.**

### Tabla

El código referente única y exclusivamente a una tabla, se creará en los métodos de la tabla en cuestión.

### Mapa

El código referente a agrupaciones de campos limitadas, como por ejemplo los campos de dirección lo incluiremos en el mapa de la agrupación.

### Vista

En un objeto de tipo vista únicamente incluiremos métodos de tipo Display (Visualización de campos calculados).

### Clase

El código referente a varias tablas, acciones externas, procesos por lotes, utilidades de uso general, extensiones de clases estándar… Se incluirá en clases, organizadas por módulos y/o proceso y siguiendo las convenciones de nomenclatura.

Ejemplos

**SYPInventUtils** – Utilidades generales para procesos de inventario.

**SYPUserLogBatchDelete** – Proceso para borrado de registros de usuario en lote.

**SYPInventDimTrackingExt** – Clase que extiende (hereda) de InventDimTracking añadiendo y/o modificando parte de su lógica.

### Formulario

El código que se permite desarrollar en un formulario se estructura en cinco secciones diferenciadas:

##### Raíz del formulario

Cualquier método que por su naturaleza no se pueda incluir en una clase, ya que su ámbito de ejecución va a ser un formulario concreto y no se refiera a un registro de alguno de los orígenes de datos, o los métodos que guardan la lógica que se produce en un formulario al modificar alguno de los valores de los campos que presenta, se escribirán en la raíz del formulario.

##### Raíz de un origen de datos

En la raíz del origen de datos del formulario permitiremos código que afecte al registro seleccionado del origen de datos en el formulario, código que afecte a cualquiera de sus registros en general debería estar incluido en su tabla o tabla extendida.

##### Campos de un origen de datos

En los campos de un origen de datos no permitiremos introducir ningún método de creación propia, es decir que no se haya generado mediante la herramienta **Anular método…** Estos métodos deberán incluir el menor código posible, actuando como disparadores para métodos permitidos en otros puntos del formulario u otros objetos.

##### Campos de datos del formulario

En los controles de formulario, cajas de texto, botones de opción, etc… no permitiremos introducir ningún método de creación propia, es decir que no se haya generado mediante la herramienta **Anular método…** Estos métodos deberán incluir el menor código posible, actuando como disparadores para métodos permitidos en otros puntos del formulario u otros objetos.

##### Botones de formulario

En los botones de los formularios sólo permitiremos incluir código que llame a métodos que se permitan en otros puntos del formulario u otros objetos. No permitiremos introducir ningún método de creación propia, es decir que no se haya generado mediante la herramienta **Anular método…**

## Valores hard-coded

Bajo ningún concepto se permitirá poner en producción cualquier desarrollo que incluya un valor de configuración escrito en forma de literal en el código fuente. Para cualquier valor que se requiera y no tenga un campo especificado para su almacenaje en el sistema estándar de la plataforma se creará el parámetro pertinente, asignándolo al módulo y sección que le corresponda por contexto.

## Comentado del código fuente

Cualquier código fuente que se modifique o cree nuevo en la plataforma deberá ser firmado por el desarrollador que lo haya incorporado, aunque no requeriremos un formato de firma en concreto si exigiremos como mínimo, los datos siguientes:

1. **Id. del proyecto para el que se ha implementado el código firmado.**
2. **Fecha de la implementación del código.**
3. **Empresa para la cual trabaja el desarrollador.**
4. **Nombre y apellidos del desarrollador.**
5. **Marca de inicio y fin de la modificación realizada.**