MIGRACIÓN DE APLICACIONES A LA VERSIÓN 5.0.0

1. CODIFICACIÓN UTF8 DE LA APLICACIÓN

DGTIC:

a) Para empezar, lo mejor es crear una rama del proyecto a migrar partiendo de la versión (trunk o rama) correspondiente.

b) Exportamos a nuestro local (p.ej. *proy_local*)

c) Copiar en el directorio raíz del proyecto el fichero *APL2UTF8.sync*

d) Para codificar todos los ficheros a UTF8, ejecutar desde el eclipse el script *APL2UTF8.sync*. Y una vez ejecutado, veremos que se ha creado una copia del proyecto en el directorio "*proy_localUTF8_COPIA"*

e) Vacíamos en el SVN el **contenido de la rama** que hemos creado.

f) En local, desde nuestro proyecto "*proy_local*" reemplazamos por el contenido del SVN. Tendremos en local el proyecto "*proy_local*" vacío.

g) Copiar de "*proy_localUTF8_COPIA"* a "*proy_local*".

h) Hacer **commit** del proyecto.

Ya tenemos el proyecto codificado en UTF8.

- **NOTA SOBRE LA CODIFICACIÓN**

  Si en el servidor no está configurada la codificación UTF-8 por defecto en el PHP, (no es el caso de la DGTIC que no tiene definida UTF-8 por defecto en el PHP por compatibilidad con PHP anteriores), es importante incluir las directivas de forma explícita en el código.

  Incluyendo al principio del fichero la siguiente directiva:

  @ini_set('default_charset', 'UTF-8');

  Principalmente en los siguientes ficheros:

  → *apl\index.php*
  → *apl\login.php*
  → *apl\phrame.php*

Además de codificar vuestros ficheros fuentes en UTF-8 hay conectar con el servidor de BBDD fijando la misma codificación.
A veces puede ser necesario hacer uso de llamadas *utf_decode()* o *utf8_encode()*, por ejemplo, si se parsean ficheros XML.

CAMBIOS EN LAS CLASES MANEJADORAS Y DE LÓGICA

2. Cambios en los procesos de **conexión a la BD**. La conexión a la BD penaliza bastante el rendimiento, por eso se ha realizado un cambio para retardar el proceso de conexión hasta el momento necesario. Esto afectará a todo el código fuente que haga uso la clase IgepConexion.
   El paso de migración consiste en **buscar** la cadena "**new IgepConexion**" y en todas aquellas líneas **incorporar** debajo una llamada al método conectar:

   ```
   $conexion = new IgepConexion($dsn);
   $conexion->conectar();
   ```

3. **Revisión de paso de parámetros por $_GET**. Se ha cambiado el sistema de comunicación a jquery+json. La información que compone las matrices de datos se obtiene a través de una matriz de datos que se mantiene en javascript. Esto es transparente al programador excepto para **aquellas llamadas que utilizaban la url para pasar parámetros**, esto ya **NO funcionará**.

4. **Nueva matriz de datos para los fil**. Principalmente afecta a listados generados a partir de un panel de búsqueda. Hasta ahora, las acciones particulares que hacían uso de los datos de un panel de búsqueda, hacían referencia a la matriz de datos *external* para localizarlos. Para evitar confusiones en esta versión, estos datos van a estar almacenados en la matriz de "*visibles*". Esto significa que se deberán revisar estas acciones (típicamente en listados) y cambiar el *setOperation*:

   ```
   $objDatos→setOperation("visibles");
   ```

5. **Ventanas de selección y listas de clase.** El constructor pasa a tener un parámetro relativo a la conexión. De modo que tenemos que cambiar la definición a:

   ```
   public function __construct($conn=null) {
       try
       {
           $conf = ConfigFramework::getConfig();
           $this->_lang = $conf->getLanguage();
           $dsn = $conf->getDSN('g_dsn');
   ```

```
                $this->_conDB = new IgepConexion($dsn);
            }
            catch(Exception $e) {
                IgepDebug::setDebug(ERROR,'Error en clase '.__CLASS__.':\
n'.PHP_EOL.$e->getMessage());
                throw new gvHidraException('Error al conectar a la BD. Contacta con el
administrador de la aplicación');
            }
        }
```

CAMBIOS EN LAS TPL

6. TPL's. Se **eliminan** los siguientes plugins:
   - ***{CWContenedorPestanyas}*** … ***{/CWContenedorPestanyas}***
   - ***{CWPestanya}***

7. TPL's. **Ya no existe el plugin**:
   - ***{CWSelector}*** … ***{/CWSelector}***

8. En las tpl que corresponden a un maestro-detalle eliminar los tags que separaban el maestro del detalle:
   </td></tr>
   <tr><td>

9. TPL's. Repasarlas para convertir a **minúsculas todos los plugins**. Esto se debe al cambio al motor smarty3.
   Aconsejable reemplazar en bloque y en este orden para no cometer errores:
   - cwpantallaentrada
   - *cwventana*
   - *cwbarrasuppanel*
   - *cwbarrainfpanel*
   - *cwbarra*
   - *cwmenulayer*
   - *cwpanel*
   - *cwbotontooltip*
   - *cwboton*
   - *cwmarcopanel*
   - */* cwcontenedorpestanayas */ ya no debe existir en ninguna tpl*
   - *cwcontenedor*
   - */* cwpestanya */ ya no debe existir en ninguna tpl*
   - *cwfichaedicion*

- *cwficha*
- *cwtabla*
- *cwfila*
- *cwpaginador*
- *cwsolapa*
- *cwinfocontenedor*
- *cwinformation*
- *cwdetalles*
- *cwgraph*
- *cwimagen*
- *cwslider*
- *cwtree*
- *cwlabel*
- *cwmaps*
- *cwareatexto*
- *cwcampotexto*
- *cwcheckbox*
- *cwlista*
- *cwrichareatexto*
- *cwupload*

10. Parámetros del plugin ***{cwbarra}...{/cwbarra}***

    Los parámetros:

    usuario=$smty_usuario codigo=$smty_codigo customTitle=$smty_customTitle

    Se unen en uno solo, por lo que deben ser sustituidos por el siguiente parámetro:

    info=$smty_info

11. Parámetros del plugin ***{cwpanel}...{/cwpanel}***

    Los siguientes parámetros se eliminan:

    "**method**" y "**tipoComprobacion**"

12. Plugin **cwbotontooltip:**
    - Parámetro "**id**", **único** en la tpl, es **obligatorio**.
    - Eliminar el parámetro "**imagen**". (Si aún se tiene este parámetro que viene de la 4.2.x)
    - Parámetro "**funcion**" se **renombra** por "**accion**". Para unificar criterios con cwboton.
    - Parámetro "**titulo**" se **renombra** por "**title**"
    - Parámetro nuevo "**label**" que funcionará igual que en el plugin **cwboton**.

13. Plugin **cwboton:**
    - Parámetro "**id**", **único** en la tpl, es **obligatorio**.
    - Si tiene ***accion="particular"*** y queremos que sea visible sin necesidad de editar el panel, se debe añadir el parámetro ***visible="true"***.
    - Eliminar el parámetro "**imagen**". (Si aún se tiene este parámetro que viene de la 4.2.x)

- Parámetro “**texto**” se **renombra** por “**label**”

14. Plugin **cwbarrasuppanel**
    - Parámetro “**titulo**” se **renombra** por “**title**”

15. Plugin **cwsolapa**
    - Parámetro “**id**”, **único** en la tpl, es **obligatorio**.
    - Parámetro “**titulo**” se **renombra** por “**title**”

16. Plugin **cwareatexto, cwcampotexto, cwcheckbox, cwlista, cwrichareatexto, cwupload, cwimagen, cwlabel, cwslider, cwtree, cwinformation:**
    - Parámetro “**textoAsociado**” se **renombra** por “**label**”
    - Parámetro “**mostrarTextoAsociado**” se **renombra** por “**showLabel**”

17. Plugin **cwlabel:**
    - Tiene un parámetro “**tipo**” con el que le se le indicará que tipo de etiqueta es la que se quiere. Las opciones son:
        - ***icon***: icono
        - ***iconLabel***: icono + etiqueta
        - ***label***: etiqueta
        - ***link***: enlace
        - ***iconLink***: icono + enlace

18. Plugin **cwmarcopanel:**
    - El parámetro “**conPestanyas**" es opcional. Con él se indicará si se quiere que aparezcan o no los botones de los paneles (buscar, listado, edición). Por defecto valdrá true, aparecen los botones.

19. Plugin **cwmenulayer:**
    - Sustituir el parámetro:

      cadenaMenu = "$smty_cadenaMenu"

      por

      arrayMenu = $smty_arrayMenu

20. Uso de la variable **$smty_iteracionActual** en la tpl.

Por el cambio de comunicación del framework, esta variable ya no existe. Por lo tanto toda la lógica implementada con ella en las tpl hay que reprogramarla.

Se invocará a la función gvh.subscribeRewireUI(), para subscribir la clase manejadora del panel y su lógica de interfaz, y tenerla accesible al paginar para poder ejecutarla y modificar la interfaz.

gvh.subscribeRewireUI(claseManejadora,function({…}))

La función que se subscribe debe implementar la lógica que queremos aplicar en la interfaz. Para ello existe una librería, **IgepJS.js**, en la que se encuentran encapsuladas todas las funciones relacionadas con la interfaz, con las que se podrán realizar las modificaciones de interfaz que se venían haciendo utilizando la variable *$smty_iteracionActual*.

FUNCIONES incluidas en *IgepJS.js*:

```
/**
* getValue: Obtiene el valor del campo
* @access  public
* @let idCampo: Nombre del campo
* @let external: [true / false]
*/
     getValue(idCampo, external = false)

/**
* setValue: Asigna valor a un campo
* @access  public
* @let idCampo: Nombre del campo
* @let valor: Valor a asignar al componente
* @let external: [true / false]
*/
     setValue(idCampo, valor, external = false)

/**
* getClass: Obtiene el class del campo indicado
* @access  public
* @let idCampo: Nombre del campo
* @let external: [true / false]
*/
     getClass(idCampo, external = false)

/**
* addClass: Modifica el class de un campo
* @access  public
* @let idCampo: Nombre del campo
* @let external: [true / false]
* @let css: estilo css a añadir
*/
     addClass(idCampo, css, external = false)

/**
* removeClass: Elimina el css indicado de un campo
* @access  public
```

```
* @let idCampo: Nombre del campo
* @let external: [true / false]
* @let css: estilo css que se ha de eliminar
*/
     removeClass(idCampo, css, external = false)

/**
* setVisible: Mostrar/ocultar un campo
* @access  public
* @let idCampo: Nombre del campo
* @let external: [true / false]
* @let visible:  [true / false]
*/
     setVisible(idCampo, visible, external = false)

/**
* getVisible: Obtener si un campo es visible o no
* @access  public
* @let idCampo: Nombre del campo
* @let external: [true / false]
*/
     getVisible(idCampo, external = false)

/**
* setEnable: Mostrar/ocultar un campo
* @access  public
* @let idCampo: Nombre del campo
* @let external: [true / false]
* @let enable:  [true / false / nuevo]
*/
     setEnable(idCampo, enable, external = false)

/**
* getEnable: Obtener si un campo es visible o no
* @access  public
* @let idCampo: Nombre del campo
* @let external: [true / false]
*/
     getEnable(idCampo, external = false)

/**
* setEditable: Activa un campo como editable
* @access  public
```

```
* @let idCampo: Nombre del campo
* @let external: [true / false]
* @let editable: [true / false / nuevo]
*/
     setEditable(idCampo, editable, external = false)

/**
* getEditable: Saber si un campo es editable o no
* @access  public
* @let idCampo: Nombre del campo
* @let external: [true / false]
*/
      getEditable(idCampo, external = false)

/**
* setBtnDisabled: Actilet un campo como editable
* @access  public
* @let idCampo: Nombre del campo
* @let external: [true / false]
* @let disabled: [true / false]
*/
     setBtnDisabled(idCampo, disabled, external = false)

/**
* getBtnDisabled: Saber si un campo es editable o no
* @access  public
* @let idCampo: Nombre del campo
* @let external: [true / false]
*/
     getBtnDisabled(idCampo, external = false)
```

EJEMPLO:

Por el uso de smarty 3 ya no es necesario encapsular el código JavaScript entre las etiquetas *{literal}...{/literal}*

```
<script type="text/javascript" defer>
```

// La lógica de interfaz que se implementaba con el uso de la variable smty_iteracionActual, ahora la desarrollaremos con el uso de los métodos indicados arriba. Esta lógica se debe ejecutar cuando la página ya está cargada (ready).

```
$(document).ready( function() {

```

// Invocación al método subscribeRewireUI para subscribir la lógica de interfaz (función) a la clase manejadora del panel.

```
     gvh.subscribeRewireUI("ClaseManejadora",function() {
```

```
        // Creamos un objeto de la clase IgepJS pasándole la clase manejadora sobre la que se trabaja en el
        panel, y el tipo de panel ('lis','lisDetalle','edi','ediDetalle')
              var objIgepJS = new gvh.IgepJS('ClaseManejadora','idPanel');

        // El objeto tendrá los métodos detallados arriba para poder trabajar con la interfaz
        // Condición equivalente con la variable {if $smty_datosFicha.smty_.edi_componente eq "S"}
              if (objIgepJS_ClaseManejadora.getValue('edi_componente') == 'S')
              {
                    objIgepJS.setEnable('edi_func', true, false);
                    objIgepJS.setEnable('edi_capas', true, false);
              }
              else {
                    objIgepJS.setEnable('edi_func', false, false);
                    objIgepJS.setEnable('edi_capas',false, false);
              }
        });
});
</script>
```

CAMBIOS EN LAS VIEWS

21. Patrones **Maestro-Ndetalles**, cambio en los **views** correspondientes, hay que añadir el tipo de panel al array de definición:

```
$detalles = array (
      array (
            "panelActivo" =>"DocumentosTramite",
            "titDetalle" =>$tituloDoc,
            "panel" => "lis"
      ),
      array (
            "panelActivo" =>"TramiteSubsana",
            "titDetalle" =>$tituloTSubsana,
            "panel" => "lis"
      )
 );
```

CAMBIOS EN LAS DTD, ficheros xml

22. Cambios en las DTDs, se ha de eliminar la DTD incluida en los ficheros xml y añadirla de forma relativa:

    gvHidraConfig.inc.xml. Eliminar la DTD y añadir lo siguiente:
    ```
    <!DOCTYPE gvHidraConfig SYSTEM "igep/dtd/configAPP.dtd">]>
    ```

    menuModulos.xml. Eliminar la DTD y añadir lo siguiente:
    ```
    <!DOCTYPE menu SYSTEM "../igep/dtd/menu.dtd">]>
    ```

    menuHerramientas.xml. Eliminar la DTD y añadir lo siguiente:
    ```
    <!DOCTYPE menu SYSTEM "../igep/dtd/menu.dtd">]>
    ```

    menuAdministracion.xml. Eliminar la DTD y añadir lo siguiente:
    ```
    <!DOCTYPE menu SYSTEM "../igep/dtd/menu.dtd">]>
    ```

23. En la DTD existe una propiedad para configurar el log a nivel de JavaScript. Por defecto, si la propiedad no existe el debug JavaScript no está activado.
    ```
    <logJSSettings status='LOG_ALL'/>
    ```

CAMBIOS EN EL AppMainWindow.php

24. Migración de la función **emptyLogTable()**:

```
public function emptyLogTable($dias=60) {
      //Recogemos dsn de conexion
      $conf = ConfigFramework::getConfig();
      $g_dsnLog = $conf->getDSN('gvh_dsn_log');
      $usuario = IgepSession::dameUsuario();
```

```
        try {
            IgepDebug::purgeDBLog($dias, $usuario, $g_dsnLog);
        }
        catch (Exception $e) {
            error_log(FILE_CLASS_.__METHOD__."Error al vaciar la
    tabla de LOG");
        }
        return 0;
    }//emptyLogTable
```

CAMBIOS EN EL mappings.php

25. Si los **botones cancelar** no tienen definido el parámetro "**action**" en la tpl, asegurarse de que en el mappings está definida la acción "**cancelarEdicion**" las siguientes líneas:

```
$this->_AddMapping('ClaseM__cancelarEdicion', 'ClaseM');
$this->_AddForward('ClaseM__cancelarEdicion',  'gvHidraSuccess',
'index.php?view=views/p_ClaseM.php&panel=listar');
```

PROGRAMACIÓN SALTOS

26. TPL:
    En la tpl el salto es provocado a partir de un cwboton o un cwbotontooltip, necesitan el parámetro **accion="saltar"**

```
{cwbotontooltip id="VMdiputados" iconCSS="glyphicon glyphicon-plus"
accion="saltar" actuaSobre="fil_DIPUTADO" titulo=#smty_AddDiputado#}
```

    CLASE MANEJADORA:

```
public function saltoDeVentana($objDatos, $objSalto)
{
    $idSalto = $objSalto→getId(); // ID del salto
    if($idSalto == 'VMdiputados')
    {
        IgepSession::borraPanel('Vmdiputados');
```

// En el caso de que estemos en un detalle y no hay datos, visibles no devolverá nada. Por lo tanto para obtener la clave del maestro utilizar dameCampoTuplaSeleccionada

```
                $objDatos->setOperation('visibles');
                $valor = $objDatos->getValue('lis_VALOR');
                if(empty($valor)){
                  $valor=IgepSession::dameCampoTuplaSeleccionada('ClasePadre',
'lis_VALOR_PADRE');
                }

                // Clase manejadora del salto
                $objSalto→setNameTargetClass('VMdiputados');
                // Modo de entrada en la clase manejadora del salto
                $objSalto→setTargetAction('iniciarVentana');

                      // Opcional: el salto se abre en ventana modal
                      $objSalto→setModal(true);
                      $objSalto->setSizeModal(1200,600);

                // Parámetros que se pasan
                $objSalto->setParam('param1',$array);
                $objSalto->setParam('param2','valorParam2');

                // REGRESO. Dos opciones
                      // 1.- acción de interfaz
                      $objSalto→setReturnAsTriggerEvent(true);
                      // 2.- ejecutar acción particular de la clase origen
                      $objSalto->setReturnAction('funcionRegreso');
                return 0;
          }
     }
     public function accionesParticulares($accion, $objDatos)
     {
          switch($accion) {
                case “funcionRegreso”:
                      // Limpiamos el objeto salto
                      IgepSession::borraPanel("saltoIgep");

                      … // Operamos

                      // Redireccionamos
                      $fw = $objDatos->getForward('correcto');
                      return $fw;
                break;
                ...
          }
     }
```

MAPPINGS:

```
            $this->_AddMapping('claseManejadora__regresoSalto',
'claseManejadora');
            $this->_AddForward('claseManejadora__regresoSalto',      'correcto',
'index.php?view=views/p_claseManejadora.php&panel=listar');
```