CASO DE ESTUDIO:
PAGINACIÓN SIN LÍMITE DE TUPLAS EN PANEL EDI DE LA RAMA 5.X DE GVHIDRA

En GV-MUSEIA se ha necesitado poder saltar a un panel EDI que muestre todas las piezas tras un aplicar un filtro de búsqueda (esto es, sin límite de número de tuplas resultantes) de modo que permita cambiar de pieza en pieza desde el propio panel EDI, pudiéndose paginar desde la primera tupla hasta la última sin necesidad de volver atrás a redefinir el filtro de búsqueda.

La clase manejadora donde el usuario define el filtro de búsqueda con un panel FIL es '`Piezas`', mientras que la clase destino del salto donde se muestra el panel EDI que se desea que tenga paginación sin límites es '`PiezaRegistro`'.

La solución consiste en que el panel FIL de la clase '`Piezas`' realice una búsqueda sin límite de tuplas, después obtenga los ids de todas las tuplas en el método '`postBuscar`' y los pase como parámetro de un salto (creado a mano) a la clase '`PiezaRegistro`'. En el constructor de ésta última clase se obtendrá el conjunto de identificadores y solo filtrará por el primero de ellos (para evitar cargar todos en memoria y desbordar la memoria). La paginación entre piezas en '`PiezaRegistro`' se realizará con una acción particular que desplazará el índice sobre el conjunto de piezas a la pieza deseada (ya sea la siguiente pieza a la actual, ya sea a la anterior pieza, a la primera, a la última, o directamente a la página elegida).

1. BOTÓN 'BUSCAR' QUE ACTUARÁ COMO UN SALTO FICTICIO.

En la plantilla Smarty de la clase '`Piezas`' se añade un botón con `accion="buscar"` para el que se definirá un parámetro `external` con el valor "`verPiezas=true`" :

```
{cwboton iconCSS="glyphicon glyphicon-eye-open" class="button" accion="buscar"
external="verPiezas=true" mostrarEspera="true" title=#smty_TitleEditarPiezas#
label=#smty_LabelEditarPiezas#}
```

2. MÉTODO PREBUSCAR EN LA CLASE FILTRO 'Piezas'.

En el método `preBuscar` se debe configurar el límite que se aplicará en la búsqueda, de modo que no haya límite para el caso deseado.

Téngase en cuenta que los resultados de la búsqueda deben tener la mínima cantidad de campos posible, para no desbordar la memoria con la consulta y para no tardar un tiempo considerable en

obtener los resultados. La recomendación es obtener solamente el id de la pieza, es decir, solamente la clave primaria de las tuplas.

En nuestro caso, además distinguimos entre búsquedas con límite y búsquedas sin límite, de modo que se puedan realizar [1] búsquedas con límite cuyo resultado se muestre en un panel LIS y [2] búsquedas sin límite cuyo resultado se muestre en un panel EDI tras un salto. Para ello usamos como discriminante el parámetro `$_REQUEST['verPiezas']`.

```
public function preBuscar ($objDatos)
{
	if (isset ($_REQUEST['verPiezas'])) {
		$verPiezas = filter_var ($_REQUEST['verPiezas'], FILTER_VALIDATE_BOOLEAN);
		$this->setLimit ($verPiezas ? -1 : 100);
		$this->modoBusqueda = ($verPiezas ? 'pieza' : 'listado');
	}
	// ...
}
```

3. MAPPING FORWARD DE LA CLASE FILTRO ‘Piezas’.

Tras ejecutar la acción `buscar` realizaremos un salto ficticio a la clase ‘`PiezaRegistro`’, de modo que el resultado de la acción ‘`Piezas__buscar`‘ se redirigirá mediante un *mapping forward* a ‘`PiezasRegistro__buscar`‘:

```
$this->AddMapping ('Piezas__buscar', 'Piezas');
// ...
$this->AddForward ('Piezas__buscar', 'verPiezas', 'phrame.php?action=PiezaRegistro__buscar');
```

4. MÉTODO POSTBUSCAR EN LA CLASE FILTRO ‘Piezas’.

En el método `postBuscar` realizamos un salto ficticio a la clase ‘`PiezaRegistro`’, pasando como parámetro del salto el conjunto de ids de las piezas. Además, reestableceremos el límite de tuplas a consultar de la clase `Piezas` para evitar que posibles consultas tras el retorno causen un desbordamiento de memoria y garantizando una reducido tiempo de ejecución de esas posibles consultas.

```
public function postBuscar ($objDatos)
{
	$verPiezas = false;
	if (isset ($_REQUEST['verPiezas'])) {
		$verPiezas = filter_var ($_REQUEST['verPiezas'], FILTER_VALIDATE_BOOLEAN);
	}
	if ($verPiezas) {
		// Saltamos a PiezaRegistro pasándole los ids de las piezas a visualizar.
		$objSalto = new IgepSalto ('Piezas', 'verPiezas');
		$objSalto->setBtnId ('verPiezas');
		$objSalto->setNameSourcePanel ('lis');

		$mDatos = $objDatos->getAllTuplas ();
```

```
                $ids = gvhPHPExtension::array_column ($mDatos, 'lis_ID');

                if (empty ($ids) || count ($ids) == 0) {
                        $this->showMessage ('APL-2');
                        return -1;
                }

                $this->setLimit (100);
                IgepSession::borraPanel ('PiezaRegistro');
                IgepSession::borraSalto ();

                $objSalto->setClase ('PiezaRegistro');
                $objSalto->setAccion ('buscar');
                $objSalto->setReturnAction ('regresoSalto');
                $objSalto->setParam ('identificador', $ids);
                IgepSession::guardaSalto ($objSalto);

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

5. MÉTODO CONSTRUCTOR EN LA CLASE DESTINO ‘PiezaRegistro’.

En el método constructor de la clase ‘`PiezaRegistro`’ obtendremos el salto y guardamos los ids de las piezas a mostrar, sin embargo solo utilizaremos el id de la primera pieza a la hora de realizar la consulta del panel.

```
class PiezaRegistro extends gvHidraForm_DB {

        /** @var [] $piezas - Contiene los ids de las piezas a editar
                              y el índice de la pieza actual */
        public $piezas = array (
                'indice'        =>      0 ,
                'ids'           =>      array (-1)
        );


        public function __construct () {

                // ...

                $objSalto = IgepSession::dameSalto ();
                if (is_object ($objSalto)
                    && $objSalto->getClaseDestino() == get_class($this)
                ) {
                        $identificadores = $objSalto->getParam ('identificador');

                        $this->piezas = array (
                                'indice'        =>      null ,
                                'ids'           =>      array ()
                        );
                        if (!empty ($identificadores)) {
                                if (!is_array ($identificadores)) {
                                        $identificadores = array ($identificadores);
```

```
                        }
                        if (count ($identificadores) > 0) {
                              $this->piezas['indice'] = 0;
                              $this->piezas['ids'] = $identificadores;
                        }
                  }

            } // is_object $objSalto

            // ...

            if ( !empty ($this->piezas['ids'])) {
                  // Where del modo de trabajo EDI
                  $idPieza = $this->piezas['ids'][ $this->piezas['indice'] ];
                  $this->setWhereForSearchQuery( <<< setWhereForSearchQuery
                        MUS_PIEZA.ID = {$idPieza}
setWhereForSearchQuery
                  );
            }

            // ...

      } // Fin del construct
```

6. ACCIÓN PARTICULAR PARA AVANZAR LA PIEZA POR PAGINACIÓN SIN LÍMITE.

Se añade una acción particular ‘`avanzarPieza`’ que permitirá avanzar la pieza actual mediante la nueva paginación al uso del panel. Para ello haremos uso del parámetro ‘`paginaPieza`’, el cual indicará el número de página al que avanzar y, por tanto, se utilizará como índice sobre el conjunto de ids para fijar el filtro del panel EDI en base al id de la pieza asociada a dicho índice.

```
public function accionesParticulares ($str_accion, $objDatos) {

      switch ($str_accion) {
            case 'avanzarPieza' :
                  $pagina = filter_var ($_REQUEST['paginaPieza'],
                                        FILTER_SANITIZE_NUMBER_INT);
                  if (!isset ($pagina) || $pagina != intval ($pagina)) {
                        return new ActionForward ('gvHidraNoAction');
                  }

                  $pagina = intval ($pagina);
                  if ($pagina < 1 || $pagina > count ($this->piezas['ids'])) {
                        $this->showMessage ('APL-666');
                        return new ActionForward ('gvHidraNoAction');
                  }

                  // Avanzamos el índice de la pieza actual
                  $this->piezas['indice'] = $pagina - 1;
                  $idPieza = $this->piezas['ids'][ $this->piezas['indice'] ];

                  $this->setWhereForSearchQuery ( <<< setWhereForSearchQuery
                        MUS_PIEZA.ID = {$idPieza}
```

```
setWhereForSearchQuery
                    );

                    $this->setFilterForSearch ( <<< setFilterForSearch
                          WHERE MUS_PIEZA.ID = {$idPieza}
setFilterForSearch
                    );

                    $this->refreshSearch (true);
                    return $objDatos->getForward ('gvHidraSuccess');
             break;

             // ...
       } // switch $str_accion

       // ...

} // Fin de accionesParticulares
```

7. MAPPING FORWARD DE LA CLASE DESTINO ‘PiezaRegistro’ QUE PERMITIRÁ LA PAGINACIÓN SIN LÍMITE.

En la clase ‘`PiezaRegistro`’ añadiremos un mapping ‘`avanzarPieza`‘ que permita ejecutar la acción asociada a la paginación sin límite de registros:

```
    $this->AddMapping ('PiezaRegistro__avanzarPieza', 'PiezaRegistro');
    $this->AddForward ('PiezaRegistro__avanzarPieza', 'gvHidraSuccess',
                       'index.php?view=views/GestionPiezas/p_PiezaRegistro.php&panel=listar');
    $this->AddForward ('PiezaRegistro__avanzarPieza', 'gvHidraError',
                       'index.php?view=views/GestionPiezas/p_PiezaRegistro.php');
```

8. PAGINADOR SIN LÍMITE DE PIEZAS EN PLANTILLA ‘PiezaRegistro’.

En la plantilla Smarty de la clase ‘`PiezaRegistro`’ se añade un paginador en la barra superior del panel `cwbarrasuppanel` , así como al final de la `cwfichaedicion` , la cual permitirá moverse entre las piezas:

```
{cwpanel id="edi" esMaestro="true" claseManejadora="PiezaRegistro" action="operarBD"
estado="$estado_edi" itemSeleccionado=$smty_filaSeleccionada}
   {* ... *}
   {cwbarrasuppanel title=#smty_tituloPanelPiezaRegistro#|cat:
$smty_datosFicha.0.edi_MUSEO_NOMBRE|cat:$smty_badge}
      {if $smty_totalPiezas > 1}
         {capture "cwbarrasuppanel_content_post"}
            {assign var="id_paginador" value=1|mt_rand:999999999}
            {$idPagination = "PiezaRegistroPaginadorSuperior_"|cat:$id_paginador}
            {$idGoToPage = "IrAPaginaSuperior_"|cat:$id_paginador}
            <span class="pull-right">
               <span class="pull-right" style="margin-right: 2.5em; margin-top: 2px;">
                  <span class="infoPag">Pg. </span>
                  <input id="{$idGoToPage}" class="goToPage" type="text" size="2">
               </span>
               <ul id="{$idPagination}" class="pagination pagination-sm pull-right"
style="margin: 0 1.5em 0 0;"></ul>
```

```
            </span>

            {include "GestionPiezas/paginador.js.tpl" idPagination=$idPagination
idGoToPage=$idGoToPage piezaActual=$smty_indicePiezaActual
totalPiezas=$smty_totalPiezas}
         {/capture}
      {/if}

      {* ... *}
   {/cwbarrasuppanel}
   {cwcontenedor}
      {cwfichaedicion id="FichaEdicion" datos=$smty_datosFicha}
         {cwficha}
            {* ... *}
         {/cwficha}

         {if $smty_totalPiezas > 1}
            {assign var="id_paginador" value=1|mt_rand:999999999}
            {$idPagination = "PiezaRegistroPaginadorInferior_"|cat:$id_paginador}
            {$idGoToPage = "IrAPaginaInferior_"|cat:$id_paginador}
            <div style="padding-top: 1em;">
               <ul id="{$idPagination}" class="pagination pagination-sm"
style="display: inline;"></ul>
               <span style="margin-left: 2.5em;">
                  <span class="infoPag">Pg. </span>
                  <input id="{$idGoToPage}" class="goToPage" type="text" size="2">
               </span>
            </div>

            {include "GestionPiezas/paginador.js.tpl" idPagination=$idPagination
idGoToPage=$idGoToPage piezaActual=$smty_indicePiezaActual
totalPiezas=$smty_totalPiezas}
            {/if}
         {/cwfichaedicion}

         {* ... *}

{/cwpanel}
```

Contenido del fichero ‘`GestionPiezas/paginador.js.tpl`’:

```
<script>
   $( '#{$idPagination}' ).twbsPagination ( {
      totalPages:   {$totalPiezas} ,
      startPage:    {$piezaActual + 1} ,
      visiblePages: 1 ,
      page:         '{literal}{{number}}{/literal} de
{literal}{{total_pages}}{/literal}' ,
      paginationClass: 'pagination pagination-nohover pagination-pageNoActive' ,
      initiateStartPageClick: false ,
      onPageClick:  function (event, page)
      {
         // Si la página pulsada es la misma que la actual, entonces no hace falta
         // paginar (habitualmente porque se acaba se cargar la página y el plugin
         // dispara éste evento).
         var currentPage = {$piezaActual + 1};
         if (page == currentPage) {
            return;
         }
```

```
         gvh.semaphoreEnqueue ( {
            id:          'click.particular-custom' ,
            keepLoading: true ,
            callback:   function() {
               $(this).controllerJS ( {
                  id:        'bttlParticular_edi' ,
                  claseM:    'PiezaRegistro' ,
                  panel:     'edi' ,
                  destino:   'phrame.php?
action=PiezaRegistro__avanzarPieza&paginaPieza='+page+'&checkForm=1' ,
                  newWindow: undefined ,
                  confirm:   '' ,
                  message:   'cargando' ,
                  esModal:   false ,
                  idVentana: undefined ,
                  lastPath:  ''
               } );
            } // callback
         } ); // gvh.semaphoreEnqueue
      } // onPageClick
   } ); // twbsPagination


   {*
    * Evento para el campo que permite ir directamente a la página deseada del
paginador.
    *}
   $( '#{$idGoToPage}' ).off ('keyup blur').on ('keyup blur', function (e)
   {
      if (e.type != 'blur') {
         var code = e.which; // normalizar códigos de teclas
         if (code !== 13) {
            return;
         }
       }

      e.preventDefault();

      var page = $('#{$idGoToPage}').val ();

      // Comprobamos que el valor introducido no es vacío.
      if (typeof page === 'undefined' || page === '' || isNaN (page)) {
         // TODO : Mostrar un aviso al usuario, alertándole que debe introducir un
valor y que debe un valor numérico válido.
         return;
      }

      // Si el valor numérico está fuera del intervalo, corregimos el valor al más
cercano al límite válido.
      var totalPages = $('#{$idPagination}').twbsPagination ('getTotalPages');
      page = parseInt (page);
      page = (page < 1 ? 1 : page);
      page = (page > totalPages ? totalPages : page);

      // Mostramos la página deseada en el paginador.
      gvh.semaphoreEnqueue ( {
         id:          'click.particular-custom' ,
         keepLoading: true ,
         callback:   function() {
            $(this).controllerJS ( {
               id:        'bttlParticular_edi' ,
```

```
                claseM:     'PiezaRegistro' ,
                panel:      'edi' ,
                destino:    'phrame.php?
action=PiezaRegistro__avanzarPieza&paginaPieza='+page+'&checkForm=1' ,
                newWindow: undefined ,
                confirm:    '' ,
                message:    'cargando' ,
                esModal:    false ,
                idVentana: undefined ,
                lastPath:   ''
            } );
          }
      } ); // gvh.semaphoreEnqueue
   } ); // #{$idGoToPage}.keyup

</script>
```