



QUIPUX-COMUNIDAD

Versión 4

SISTEMA QUIPUX

Secretaría de la Administración
Pública

2014-2015

1. INTRODUCCIÓN

Quipux está desarrollado con lenguaje de programación PHP, la mayoría de la parte funcional de documentos está desarrollado con programación Orientada a Objetos la cual nos ha permitido reutilizar el código en gran parte del sistema. Además utilizamos HTML, JAVASCRIPT, AJAX y JSON.

El esquema de desarrollo está implementado de la siguiente manera:

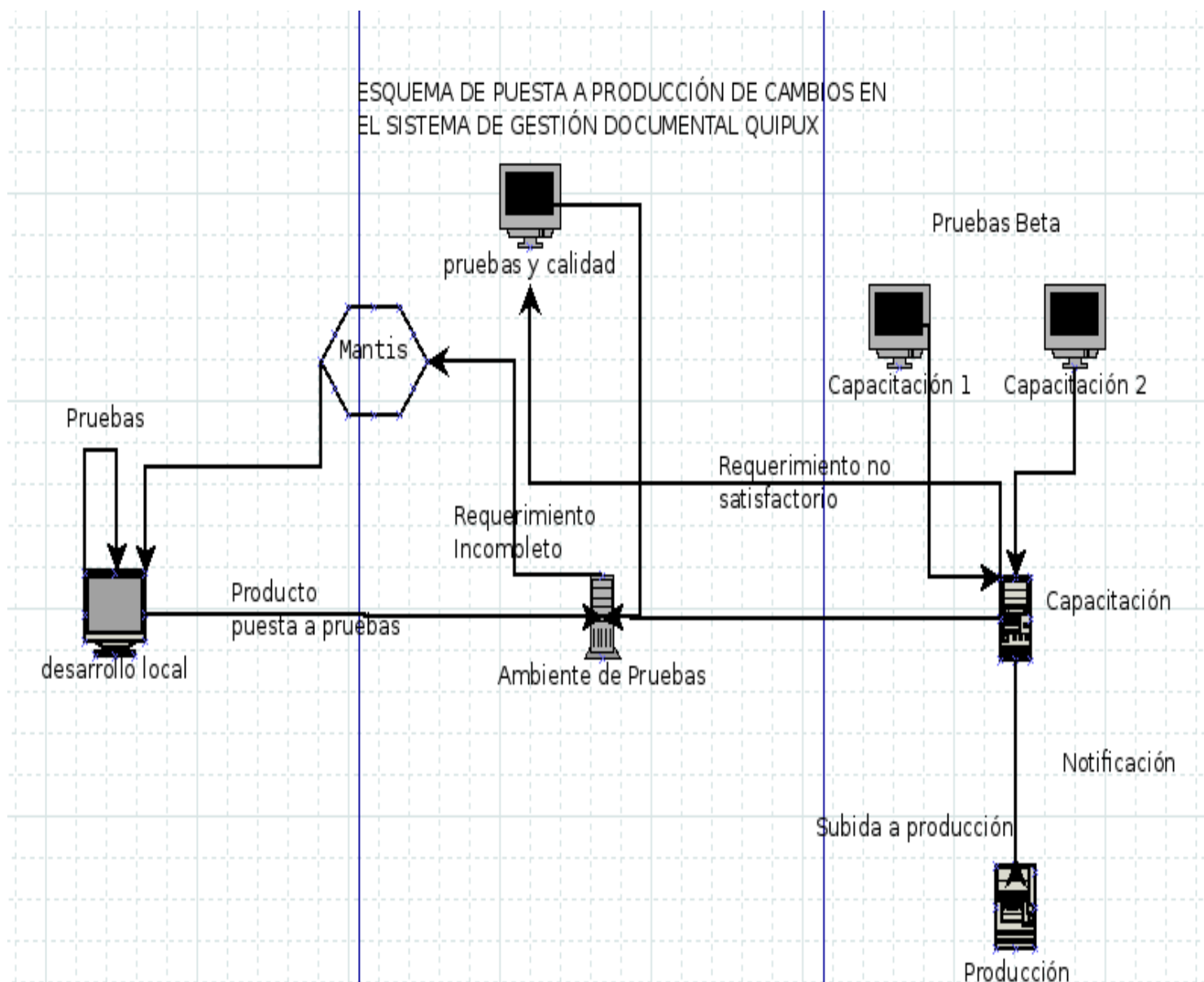


Figura 1.- Esquema de Cambios sobre el SGDQ

El esquema de programación de Quipux, 528 directorios, 3414 archivos, vamos a describir a continuación:

CARPETAS	DESCRIPCIÓN
<ul style="list-style-type: none"> Administracion <ul style="list-style-type: none"> archivos catalogos ciudadanos ciudadanos_solicitud corregir_datos_usuarios dependencias instituciones_adscritas listas mensajes_alerta migrar_bodega notificaciones subrogacion tbasicas usuarios usuarios_dependencias adodb anexos archivo asociar_documentos backup BDD bodega busqueda busquedaN cron envios estilos iconos imagenes img include <ul style="list-style-type: none"> barcode class db js local query <ul style="list-style-type: none"> administracion tx uploadFile tx upload interconexion interconexion_sistema 	<p>Administración del Sistema</p> <p>Código de archivos adjuntos al sistema</p> <p>--</p> <p>Creación, combinación de ciudadanos</p> <p>Solicitudes de firma electrónica</p> <p>--</p> <p>Creación de áreas o dependencias de Institución</p> <p>Creción de listas</p> <p>--</p> <p>--</p> <p>--</p> <p>Subrogación</p> <p>serialización de numeración de documentos</p> <p>usuarios, activación, llamada a servicio web</p> <p>asignación de usuarios y ciudadanos</p> <p>módulo de conexión de Base de datos</p> <p>vínculo de anexos a los documentos.</p> <p>administración de archivos</p> <p>asociar documentos a carpetas</p> <p>--</p> <p>--</p> <p>Bodega temporal de documentos</p> <p>Búsqueda con código like (desahabilitada actualmente)</p> <p>Búsqueda exacta (habilitada actualmente)</p> <p>Cron que ejecuta Quipux</p> <p>--</p> <p>Css del Software</p> <p>íconos</p> <p>imágenes</p> <p>img</p> <p>La carpeta include tenemos 1 carpeta principal:</p> <p>query: en donde encontramos</p> <p>Administracion: encontramos los select de las instituciones.</p> <p>Encontramos los servicios web configurados para la creación de documentos.</p>

<ul style="list-style-type: none"> — js — menu — metadatos — plantillas — radicacion — recursivas — reportes — reportes_new — seguridad — tareas — tipo_documental — tx — updates — uploadFiles 	<p>librerías javascript utilizadas en el sistema menu, menús, bandejas</p> <p>código de metadatos hoy en día llamado etiquetas. plantillas de documentos una de las carpetas más importantes del sistema, donde se maneja todas las transacionalidades de la documentación. utilizada para dependencias manejando recursividad -- Reporte del sistema Nivel de seguridad para documentos tareas relacionadas con documentos</p> <p>Tx, funcionalidades principales de transacciones</p>
---	---

Tabla 1.- Directorios de Quipux

2. ARCHIVO DE CONFIGURACIÓN

Código	Descripción
<pre>\$version_light=false; \$config_numero_meses = 6; \$numeroCaracteresTexto = 0; \$config_bloquear_acceso_ciudadano = false; // Configuración de la conexión con la BDD \$usuario = "postgres"; \$contrasena= "postgres"; \$servidor = "127.0.0.1:5432"; \$driver = "postgres"; \$db = "quipux_comunidad"; \$usuario_bodega = "postgres"; \$contrasena_bodega = "postgres"; \$servidor_bodega = "127.0.0.1:5432"; \$db_bodega = "quipux_combodega";</pre>	<p>Versión a desplegar Configuración de fechas Mínimo de número de caracteres para buscar Bloqueo a ciudadanos al sistema</p> <p>Configuración de conexión de base de datos transaccional.</p> <p>Configuración de base de datos documental</p>

Tabla 2.- Configuración Quipux

3. MANEJO DE SENTENCIAS SQL

Obtención de datos para varios registros

Código	Descripción
<pre> \$sql = "select usua_nombre, usua_apellido, usua_abr_titulo, usua_cargo , usua_institucion, usua_area, inst_codi from usuarios_radicado where radi_num_radi=\$radicado and radi_usua_tipo=\$tipo"; \$rs=\$db->conn->query(\$sql); \$cargo=""; \$usua_nombre=""; while(\$rs && !\$rs->EOF) { \$cargo.= \$rs->fields["USUA_CARGO"]; \$usua_nombre.= \$rs->fields["USUA_NOMBRE"]; \$rs->MoveNext(); } </pre>	<p>Armando el query</p> <p>Obtener datos del query Si obtenemos varios datos y varios registros, manejamos con While</p> <p>en la variable carga lo obtenido query, en un bucle alimentando cada vez cargo y usua_nombre</p>

Tabla 3.- Obtención del result de una sentencia, varios registros

Obtención de datos para varios registros

Partiendo del mismo ejemplo

Código	Descripción
<pre> \$sql = "select usua_nombre, usua_apellido, usua_abr_titulo, usua_cargo , usua_institucion, usua_area, inst_codi from usuario where usua_codi = 234"; \$rs=\$db->conn->query(\$sql); \$cargo=""; if (!\$rs or \$rs->EOF) { \$cargo = \$rs->fields["USUA_CARGO"]; \$usua_nombre = \$rs->fields["USUA_NOMBRE"]; } </pre>	<p>Armando el query</p> <p>Obtener datos del query Si obtenemos varios datos</p> <p>en cada variable se carga lo que se obtiene del query</p>

Tabla 4.- Obtención del result de una sentencia, un registro

4. MANEJO DE PERMISOS Y SESIONES

/var/www/html/quipux/session_orfeo.php

Permite el manejo de sesiones, expiración de la sesión y permisos, Si bien es cierto el manejo de las sesiones tenemos configurado en apache en php.ini (15 minutos), también debemos controlar para que el sistema caduque la sesión y se deslogue el usuario a continuación vemos el siguiente código que permite ejecutar lo antes descrito.

Conexion a la Base de datos	El archivo session_orfeo.php, es llamado en todo el sistema para controlar las sesiones y conexiones ala base de datos
Código	Descripción
<pre>Variable \$ruta_raiz include_once ("\$ruta_raiz/include/db/ConnectionHandler.php"); include_once ("\$ruta_raiz/config.php"); include_once ("\$ruta_raiz/config_replicacion.php"); error_reporting(7); \$db = new ConnectionHandler("\$ruta_raiz"); \$db->conn->SetFetchMode(ADODB_FETCH_ASSOC);</pre>	<p>Es el path relativo para dirigirse a los archivos a utilizar</p> <p>Conexion a Postgres</p> <p>Replicación de la base de datos, configurado en el config.php</p> <p>Variable de conexión para Quipux, utilizada en todo el sistema</p>

Tabla 5.- Manejo de Conexiones

Código	Descripción
<pre>\$query = "select p.nombre, count(pc.id_permiso) as permiso from permiso p left outer join permiso_usuario pc on p.id_permiso=pc.id_permiso and pc.usua_codi=\$usua_codi group by p.nombre"; \$rs = \$db->query(\$query); while(\$rs && !\$rs->EOF) { \$nom_perm = \$rs->fields["NOMBRE"]; \$_SESSION[\$nom_perm] = \$rs->fields["PERMISO"]; \$rs->MoveNext(); }</pre>	<p>Consulta a la tabla permiso y permiso_usuario, obteniendo antes el código de usuario.</p> <p>Se abre sesiones de acuerdo a los permisos configurados por el sistema.</p>

Tabla 6.- Manejo de Sesiones Quipux

Además de las permisos que se asignan a sesiones tenemos las siguientes variables de sesión.

inst_codi, Id de Institución perteneciente al usuario logeado

usua_codi, Id del usuario perteneciente al usuario logeado

depe_codi, Id de la dependencia al usuario logeado

usua_nombre, nombre de usuario logeado

usua_codi_jefe, Id del jefe del usuario logeado

Sintáxis para variables enteras

Código	Descripción
<code>\$_SESSION["usua_codi_jefe"] = (!\$rsComp->EOF) ? 0+\$rsComp->fields["USUA_CODI_JEFE"] : 0;</code>	este es un if si existe el código del jefe se asigna, caso contrario se asigna el valor de 0

Tabla 6.- Asignación de una variable a una sesion

5. CONFIGURACIÓN DESEABLE DEL APACHE

Por ejemplo podemos crear un archivo que contenga la siguiente información:

/etc/apache2/conf.d/quipux.conf, las siguientes variables tendrán acceso en el config.php de quipux, también depende de la distribución de Quipux.

SetEnv DB_USER "postgres"

SetEnv DB_PASS "postgres"

SetEnv DB_SERVER "192.168.0.2:5432"

SetEnv DB_DRIVER "postgres"

SetEnv DB_NAME "quipux"

Configuración config.php	Descripción
<pre>\$usuario = \$_SERVER['DB_USER']; \$contrasena = \$_SERVER['DB_PASS']; \$servidor = \$_SERVER['DB_SERVER']; \$driver = \$_SERVER['DB_DRIVER']; \$db = \$_SERVER['DB_NAME'];</pre>	Variables configurados en el quipux.conf

Tabla 7.- Configuración deseable

De la tabla permisos tenemos los siguientes campos:

Esquema html del sistema (**index_frame.php**)

<u>FRAME CABECERA</u> (Usuarios) f_top.php	
<u>FRAME MENU</u> (Menu) correspondencia.php	<u>FRAME CUERPO</u> (Contenido de Bandejas) cuerpo.php

Tabla 8.- Esquema de Interfáz

Interpretado en Html



Figura 2.- Esquema de interfáz

De cada archivo descrito se explicará las funciones principales para el mantenimiento del sistema.

El siguiente archivo nos permite administrar los menús a desplegar para los usuarios, dependiendo del perfil, se desplegará cada ítem.

En esta carpeta se encuentra los menús de bandejas a desplegar:

quipux/menu/

La función **crear_item_bandeja(posicion,leyenda, leyenda alterna,link)** se encuentra en

quipux/menu/correspondencia.php

Por cada ítem generamos una fila y armamos el html, al final hacemos un return de la variable tr.

Tabla 9.- Esquema de Interfáz

Configuración config.php	Descripción
<pre><body onload="recargar_estadisticas(); init_menu();"> <center> <div id="div_bloquear_menu" style="width: 100%; height: 0%; z- index: 1000; position: fixed; top: 0; left: 0;"></div>
 <table width="160px" border="0" cellpadding="0" cellspacing="3"> <?php if (\$_SESSION["tipo_usuario"]==2) { if (\$_SESSION["inst_codi"] == 1) { include "\$ruta_raiz/menu/nuevo_salida.php"; include "\$ruta_raiz/menu/ciudadano_firma.php"; } else {</pre>	<p>HTML</p> <p>TIPO USUARIO =2 CIUDADANO TIENEN LOS PRIVILEGIOS</p>

<pre> include "\$ruta_raiz/menu/ciudadano.php"; } } else { if (\$_SESSION["depe_codi"]!=0) { // Si no tiene un área no puede realizar ninguna accion porque genera errores de secuencias. include "\$ruta_raiz/menu/nuevo_salida.php"; include "\$ruta_raiz/menu/bandeja.php"; include "\$ruta_raiz/menu/radicacion.php"; } include "\$ruta_raiz/menu/menuPrimero.php"; } ?> </table>
 <div id="div_estadisticas_menu" style="width: 160px;"></div> </pre>	INCLUDES de los PHP que generan menús
---	---------------------------------------

Tabla 10.- menú de Quipux

/var/www/html/quipux/obtenerdatos.php, funciones principales

Código	Descripción
<pre> \$usua=ObtenerDatosUsuario(\$_SESSION["usua_codi"], \$db,"U"); </pre>	Funciones reutilizables, a partir de del usua_codi, enviamos como parámetro, y obtenemos los siguientes datos:
<pre> \$vector["usua_codi"] =trim(\$rs->fields["USUA_CODI"]); \$vector["cedula"] =trim(\$rs->fields["USUA_CEDULA"]); \$vector["login"] =trim(\$rs->fields["USUA_LOGIN"]); \$vector["nombre"] = \$rs->fields["USUA_NOMBRE"]; \$vector["usua_nombre"] = \$rs->fields["USUA_NOMB"]; \$vector["usua_apellido"] = \$rs->fields["USUA_APELLIDO"]; \$vector["inst_codi"] =trim(\$rs->fields["INST_CODI"]); \$vector["institucion"] = \$rs->fields["INST_NOMBRE"]; \$vector["inst_estado"] =trim(\$rs->fields["INST_ESTADO"]); \$vector["usua_estado"] =trim(\$rs->fields["USUA_ESTA"]); \$vector["cargo"] = \$rs->fields["USUA_CARGO"]; \$vector["cargo_cabecera"] = \$rs->fields["USUA_CARGO_CABECERA"]; \$vector["titulo"] = \$rs->fields["USUA_TITULO"]; \$vector["abr_titulo"] = \$rs->fields["USUA_ABR_TITULO"]; \$vector["email"] = \$rs->fields["USUA_EMAIL"]; \$vector["dependencia"] = \$rs->fields["DEPE_NOMB"]; \$vector["depe_codi"] = \$rs->fields["DEPE_CODI"]; \$vector["tipo_usuario"] = \$rs->fields["TIPO_USUARIO"]; \$vector["direccion"] = \$rs->fields["USUA_DIRECCION"]; \$vector["telefono"] = \$rs->fields["USUA_TELEFONO"]; \$vector["ciudad"] = \$rs->fields["USUA_CIUADAD"]; </pre>	Retornamos el vector: así podemos hacer uso con la llamada a la función <pre> \$usua=ObtenerDatosUsuario(\$_SESSION["usua_codi"], \$db,"U"); echo \$usua["cedula"]; </pre> //desplegará la cédula correspondiente al usuario logeado.

<pre> \$vector["usua_firma_path"] = \$rs- >fields["USUA_FIRMA_PATH"]; \$vector["tipo_certificado"] = \$rs- >fields["USUA_TIPO_CERTIFICADO"]; \$vector["cargo_tipo"] = \$rs->fields["CARGO_TIPO"]; </pre>	
---	--

Tabla 11.- obtener datos de un usuario

Código	Descripción
<pre> function ObtenerDatosRadicado(\$radicado,\$db) { //Obtiene datos del radicado en función del texto if (trim(\$radicado)=="") return array(); \$db->conn- >SetFetchMode(ADODB_FETCH_ASSOC); \$sqlTEXT = "select r.*, e.esta_desc, tr.trad_desc, radi_fech_asig::date - CURRENT_DATE as num_dias from radicado r left outer join estado e on r.esta_codi=e.esta_codi left outer join tiporad tr on r.radi_tipo = tr.trad_codigo where r.RADI_NUME_RADI='".trim(\$radicado); \$rs=\$db->query(\$sqlTEXT); \$vector["radi_ume_radi"] = \$rs- >fields["RADI_NUME_RADI"]; \$vector["radi_ume_temp"] = \$rs- >fields["RADI_NUME_TEMP"]; codigo de los campos de la tabla radicado return \$vector; </pre>	Obtener los datos de acuerdo al número de radicado

Tabla 12.- obtener datos de un documento

6. NUMERACIÓN DE DOCUMENTOS

20140000000000000001

2014	000000	000000001
año	dígitos 0	9 dígitos secuenciales

Tabla 13.- numeración de documentos

términa en **0** cuando el documento pertenece al usuario remitente, o es generado en la institución,
 termina en **1**, cuando el documento pertenece al destinatario o copia,

termina en 2 si el documento está registrado como externo.

Generación de Documentos

En la generación de documentos tenemos el archivo

quipux/radicacion/NEW.php en donde se grafica la interfaz de la página.

Código	Descripción
<pre><form action="funciones_NEW.php?<?=\$var_envio?>" ENCTYPE="multipart/form-data" method="post" name="formulario" id="formulario"> <!--onChange="document.formulario.fl_modificar1.value=1;"--> <input type="hidden" name="hidden_tipo_anterior" id="hidden_tipo_anterior" value="<?=\$raditipo?>" /> <input type="hidden" name="hidden_radi_actual" id="hidden_radi_actual" value=""/> <input type="hidden" name="hidden_actualiza_opciones" id="hidden_actualiza_opciones" value=""/> <input type="hidden" name="hidden_titulo_anterior" id="hidden_titulo_anterior" value=""/> <input type="hidden" name="bandera_cambiarop" id="bandera_cambiarop" value="1"/> <input type="hidden" name="txt_refeResponder" id="txt_refeResponder" value="<? =\$txt_refeResponder?>" /> <input type="hidden" name="hidden_lista_modificada" id="hidden_lista_modificada" value="<? =\$lista_modificada?>" /> <input type="hidden" name="txt_plugins_navegador" id="txt_plugins_navegador" value=""/></pre>	<p>Acción que registra la metadata del documento</p> <p>Si el documento es reasignado o editado, se muestran estos valores</p>

Tabla 14.- formulario de nuevo documentos

El siguiente archivo es donde se ejecuta el guardado del documento, realizando algunas validaciones:

quipux/radicacion/funciones_NEW.php

Código	Descripción
<pre>function validar_html(\$html) { \$html = preg_replace('<input (.*)type=[\'\"]?(hidden submit button image reset file)[\'\"]?.*>;i', '', \$html); \$html = preg_replace('<style.*?>.*?</style>;is', '', \$html); \$html = preg_replace('<img.*?>;is', '', \$html); \$html = preg_replace('<!--.*?-->;is', '', \$html); \$html = preg_replace('<p.*?>;is', '', \$html); //aumentado por SC para evitar que se dañen las viñetas al pegar desde OOo \$html = preg_replace('<col.*?>;is', '', \$html); //aumentado por SC para evitar que se dañen las tablas al pegar desde OOo \$html = str_replace("<p>", "
", \$html); \$origen = array("á", "é", "í", "ó", "ú", "ñ", "Á", "É", "Í", "Ó", "Ú", "Ñ"); \$destino =</pre>	<p>Función que permite eliminar html en el cuerpo del documento.</p>

<pre>array("&aacute;","&eacute;","&iacute;","&oacute;","&uacute;","&ntilde;"; ,"&Aacute;","&Eacute;","&Iacute;","&Oacute;","&Uacute;","&Ntilde;"); \$html = str_replace(\$origen, \$destino, \$html); return \$html; }</pre>	
--	--

Tabla 15.- Validación de Html

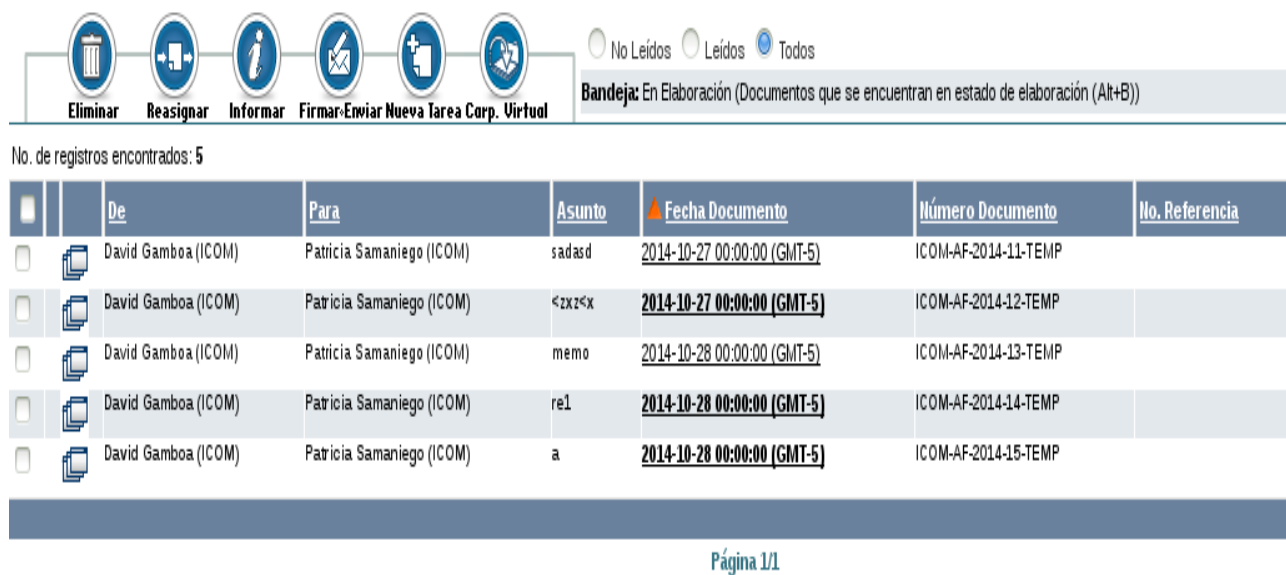
La función que obtiene el número de documento es:

Código	Descripción
newRadicado(\$tpRad, \$Dependencia, &\$noRadText)	Función
\$newRadicado = date("Y") . str_pad(\$Dependencia,6,"0", STR_PAD_LEFT) . str_pad(\$secNew,9,"0", STR_PAD_LEFT) . \$tpRad;	Cadena de texto para recuperar el número de documento correspondiente

Tabla 16.- Numeración de Documentos

7. CARPETA tx

Las acciones que tenemos en cada bandeja son ejecutadas por javascript llamando a su vez a los php que procesaran la información dependiendo de la opción realizada.



Bandeja: En Elaboración (Documentos que se encuentran en estado de elaboración (Alt+B))

No. de registros encontrados: 5

	De	Para	Asunto	Fecha Documento	Número Documento	No. Referencia
	David Gamboa (ICOM)	Patricia Samaniego (ICOM)	sadasd	2014-10-27 00:00:00 (GMT-5)	ICOM-AF-2014-11-TEMP	
	David Gamboa (ICOM)	Patricia Samaniego (ICOM)	<zxz<x	2014-10-27 00:00:00 (GMT-5)	ICOM-AF-2014-12-TEMP	
	David Gamboa (ICOM)	Patricia Samaniego (ICOM)	memo	2014-10-28 00:00:00 (GMT-5)	ICOM-AF-2014-13-TEMP	
	David Gamboa (ICOM)	Patricia Samaniego (ICOM)	re1	2014-10-28 00:00:00 (GMT-5)	ICOM-AF-2014-14-TEMP	
	David Gamboa (ICOM)	Patricia Samaniego (ICOM)	a	2014-10-28 00:00:00 (GMT-5)	ICOM-AF-2014-15-TEMP	

Página 1/1

Figura 3.- Acciones de las Bandejas

Al seleccionar una de las opciones, ejecutamos el javascript enviando un código, cada boton de la parte superior de arriba de la figura 3, tiene un código el cual es ejecutado por un solo php, el cual

describimos a continuación.

Para ejecutar una acción, es necesario seleccionar uno o varios documentos:

Los botones redondos de la Figura 3, son graficados de la siguiente manera:

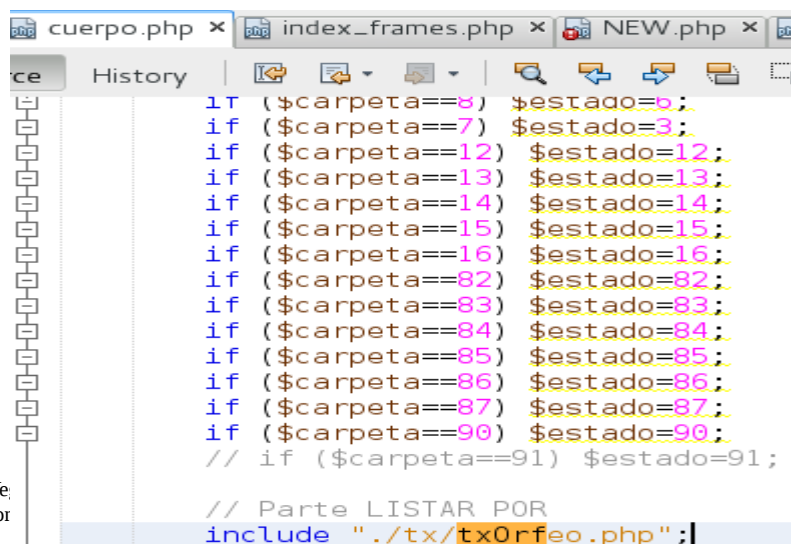
Por ejemplo: para dibujar el boton eliminar tenemos:

Código	Descripción
<pre><td valign="bottom" id="btn_noeliminar" style='display:none'> <a href="javascript:void(0);" onMouseOut="MM_swapImgRestore()" onClick="seleccionBarra = 6;changedepesel(6);" onMouseOver="MM_swapImage('Image3','<? =\$ruta_raiz?>/imagenes/internas/overNoEliminar.gif',1)" title="Shift+Ctrl+R"> <img src="<?=\$ruta_raiz? >/imagenes/internas/noEliminar.gif" name="Image3" border="0" alt="Restaurar"> </td></pre>	<p>A parte del html que grafica el boton y la imagen que despliega, tenemos la función <code>changedepesel(codigoEjecutar)</code>, esta permite ejecutar la acción enviada.</p>
<pre>function changedepesel(enviara) { document.form1.codTx.value = enviara; envioTx(); }</pre>	<p>Como podemos ver esta función se encarga de alimentar una caja de texto tipo hidden, también ejecutará la funcion <code>envioTx()</code>;</p>

Tabla 17.- Ejecución de acciones de las bandejas de Entrada y Salida

El archivo en donde se ejecutan las acciones es:

Si revisamos el código de la página `corpo.php`, podemos observar que se realiza un include a `txOrfeo.php`



```

17 ($carpeta==8) $estado=6;
18 if ($carpeta==7) $estado=3;
19 if ($carpeta==12) $estado=12;
20 if ($carpeta==13) $estado=13;
21 if ($carpeta==14) $estado=14;
22 if ($carpeta==15) $estado=15;
23 if ($carpeta==16) $estado=16;
24 if ($carpeta==82) $estado=82;
25 if ($carpeta==83) $estado=83;
26 if ($carpeta==84) $estado=84;
27 if ($carpeta==85) $estado=85;
28 if ($carpeta==86) $estado=86;
29 if ($carpeta==87) $estado=87;
30 if ($carpeta==90) $estado=90;
31 // if ($carpeta==91) $estado=91;

// Parte LISTAR POR
include "../tx/txOrfeo.php";

```

Figura 4.- dibujar botones

El php que ejecuta estas acciones dependiendo que botón presionamos, tenemos:

quipux/tx/realizarTx.php

Código	Descripción
<pre>\$chk_firma = \$_POST['chk_firma']; \$codTx = \$_POST['codTx']; \$observa = \$_POST['observa']; \$fechaAgenda= \$_POST['fechaAgenda']; \$checkValue = \$_POST['checkValue']; \$fecha_doc = \$_POST['fecha_doc'];</pre>	Recibimos por Post, de las cajas de texto hidden
Luego	
Dependiendo del código enviado por la función changedepesel(codigoEjecutar), tenemos el siguiente script	
<pre>switch (\$codTx) { case 2: //Eliminar Documentos \$nombTx = "Eliminar Documentos "; \$susCodDestino = \$tx->eliminarDocumento(\$radicadosSel, \$_SESSION["usua_codi"], \$observa); if(\$susCodDestino!="") \$MensajeTx="El/Los documento(s) est&aacute;n en la bandeja &quot;Eliminados&quot;."; else echo "
No se pudo realizar esta acci&oacute;n. Solo los documentos en estado de Elaboraci&oacute;n pueden ser eliminados.
"; break; case 6: //Sacar de eliminados sigue código</pre>	Se compone por un switch

Tabla 18.- Ejecución de acciones de las bandejas de Entrada y Salida

De la misma manera podemos mencionar a los archivos: formEnvio.php y formEnvio_ajax.php funcionan de la misma manera.

8. EJECUCIÓN DE CONSULTAS DE BANDEJAS

cuerpo.php, es el encargado de dibujar la parte frontal al usuario, en este mismo archivo ejecuta por ajax el archivo cuerpo_paginador.php.

Para llamar al cuerpo_paginador.php tenemos el siguiente script:

```
$paginador = new ADO_DB_Pager_Ajax($ruta_raiz, "div_cuerpo", "cuerpo_paginador.php",
    "txt_fecha_desde,txt_fecha_hasta,estado,busqRadicados,tiposLectura,slc_tarea_estado,radi_tipo",
    "carpeta=$carpeta");
```

Por GET se envía a cuerpo_paginador las variables txt_fecha_desde,txt_fecha_hasta, etc, estas variables son las que se toma por html de las cajas de texto.

quipux/cuerpo_paginador.php

Código	Descripción
<pre>\$txt_fecha_desde = trim(limpiar_sql(\$_GET['txt_fecha_desde'])); \$txt_fecha_hasta = trim(limpiar_sql(\$_GET['txt_fecha_hasta'])); \$estado = 0+\$_GET['estado']; \$busqRadicados = trim(limpiar_sql(\$_GET['busqRadicados']));</pre>	<p>Recibimos lo que enviamos en la función anterior</p>
<pre>include "\$ruta_raiz/include/query/queryCuerpo.php"; echo "
"; \$pager = new ADO_DB_Pager(\$db,\$sql,'adodb', true,\$orderNo, \$orderTipo,true); \$pager->checkAll = false; \$pager->checkTitulo = true; \$pager->toRefLinks = \$linkPagina; \$pager->toRefVars = \$encabezado; \$pager->descCarpetasGen=\$descCarpetasGen; \$pager->descCarpetasPer=\$descCarpetasPer; \$pager->Render(\$rows_per_page=20,\$linkPagina, \$checkbox=chkAnulados);</pre>	<p>En este archivo está el query a ejecutar la consulta.</p> <p>tenemos el componente</p> <p>algunas variables que dejamos por defecto</p> <p>Número de registros por página</p>

Tabla 19.- Ejecución de acciones de las bandejas de Entrada y Salida

9. INSERT Y UPDATES EN EL SISTEMA

En Quipux, utilizamos la misma librería de conexión la cual nos permite usar la función REPLACE que nos ayuda a ejecutar el insert o update en la base de datos del sistema.

Por ejemplo:

Código	Descripción
<pre>\$record["indi_codi"] = \$repos_codi; \$record["esta_codi"] = \$repos_estado; \$record["tamano"] = "pg_table_size(nombre_tabla)::numeric"; \$ok = \$dba->conn->Replace("indice", \$record, "indi_codi", false, false, true, false);</pre>	<p>sería para un UPDATE</p> <p>Replace("indice", \$record, "indi_codi", false, false, true, false);</p> <p>indice: es la tabla a insertar o modificar</p> <p>\$record: es el array que contiene los datos y campos a</p>

	insertar o modificar. UPDATE indi_codi es el campo where si es un update a ejecutar INSERT indi_codi no va sería para un INSERT Replace("indice", \$record, "", false, false, true, false); los demás valores dejarlos por defecto, si necesitamos ver el query que se ejecuta, el último parámetro ponerlo en true
--	---

Tabla 20.- Ejecución de acciones de las bandejas de Entrada y Salida

10. EL ARCHIVO OSCURO tx.php

En este archivo se manejan las acciones que se realizan con los documentos, por lo general todas las funciones tienen como parámetro **radicados**, el cual es un array de código de documentos para ejecutar las acciones. De esta manera controlamos el check de las bandejas de documentos:

Código	Descripción
<pre>foreach(\$radicados as \$radi_nume) { //aquí ejecutamos las acciones que vamos a realizar por cada documento seleccionado, claro los documentos seleccionados deben tener la misma acción para ejecutar. }</pre>	

Tabla 21.- foreach de un conjunto de documentos seleccionados.

Este archivo es el más importante en todo el sistema, es recomendable si se va a cambiar una de las funciones copiar y renombrarla con otro nombre para no afectar al sistema.

Por poner un ejemplo tenemos la función informar(radicados,usua_codi,usua_dest,observa)

radicados: id de radicados a modificar

usua_codi: usuario quien realiza la acción

usua_dest: usuario a quien va dirigida la acción

observa: observaciones que ingresa el usuario quien realiza la acción

Código	Descripción
--------	-------------

<pre>function informar(\$radicados, \$susua_codi, \$susua_dest, \$observa) { include_once "../obtenerdatos.php"; //Consulta de datos de los usuarios y radicados \$usr_dest = ObtenerDatosUsuario(\$susua_dest,\$this->db); //\$observa = "A: " . \$usr_dest["login"] . " - \$observa"; \$mail_param["num_docs"] = 0; foreach(\$radicados as \$radi_num) { # Asignar el valor de los campos en el registro \$record["RADI_NUME_RADI"] = \$radi_num; \$record["INFO_DESC"] = \$this->db->conn->qstr(\$observa); \$record["INFO_FECH"] = \$this->db->conn->sysTimeStamp; \$record["USUA_CODI"] = \$susua_dest; \$record["USUA_INFO"] = \$susua_codi; //Insertamos los datos \$informaSql = \$this->db->conn->Replace("INFORMADOS", \$record,array('RADI_NUME_RADI','USUA_INFO','USUA_CODI'),false,false,true,false) ; \$this->insertarHistorico(\$radi_num, \$susua_codi, \$susua_dest, \$observa, 8); ++\$mail_param["num_docs"]; } \$mail_param["enviado_por"] = "Informado por:"; \$mail_param["bandeja"] = "Informados"; if (\$mail_param["num_docs"] == 1) \$this->enviarMail(\$susua_codi, \$susua_dest, \$radi_num, "Documento Informado", "0", \$mail_param); elseif (\$mail_param["num_docs"] > 1) \$this->enviarMail(\$susua_codi, \$susua_dest, \$radi_num, "Documento Informado", "9", \$mail_param); return \$usr_dest["nombre"]; }</pre>	<p>Ejemplo</p> <p>funcion que tenemos para reutilizarla</p> <p>empieza el for para ejecutar el proceso a todos los radicados en el array dado.</p> <p>Insertamos en la tabla INFORMADOS</p> <p>Insertamos en Histórico</p> <p>Si necesitamos enviar mail</p>
--	--

Tabla 22.- foreach de un conjunto de documentos seleccionados.

11. CARPETA INTERCONEXIÓN

En la carpeta interconexión tenemos los servicios web configurados, para explicar de manera didáctica vamos a utilizar el servicio web que genera documentos:

Invocamos al archivo que tiene el servicio web a consumir

```
require_once $this->ruta_raiz."/interconexion/generar_pdf.php";
```

Llamamos al servicio web

```
$pdf = ws_generar_pdf_base64($this->documento_html, $this->plantilla_documento, $servidor_pdf, $this->registro_padre["estado"], $this->numero_documento, $this->fecha_documento, $this->registro_padre["ajust_texto"], $this->formato_pdf);
```

Código	Descripción
<pre>function ws_generar_pdf_base64(\$html, \$plantilla, \$servidor, \$estado="", \$numDocu="", \$fechDocu = "", \$numPag = "", \$orientPag="V") { try { \$wsdl = "\$servidor/html_a_pdf.php?wsdl"; if(!@file_get_contents(\$wsdl)) { throw new SoapFault('Server', 'No WSDL found at ' . \$wsdl); return "0"; } //Llamado a la clase SOAP PHP para instanciar clienteSOAP ini_set('soap.wsdl_cache_enabled', '0'); \$archivo = ""; if (trim(\$plantilla) != "") { if (is_file(\$plantilla)) \$archivo = base64_encode(file_get_contents(\$plantilla)); } \$oSoap = new SoapClient("\$wsdl",array("trace" => 1, "exceptions" => 0)); \$envioDatos=\$oSoap->__soapcall('html_a_pdf', array(new SoapParam(base64_encode(\$html), "set_html"), new SoapParam(\$archivo, "set_pdf"), new SoapParam(\$estado, "set_estado"), new SoapParam(\$numDocu, "set_num_docu"), new SoapParam(\$fechDocu, "set_fech_docu"), new SoapParam(\$numPag, "set_num_pag"), new SoapParam(\$orientPag, "set_orient_pag"))); //Comentar /* var_dump(\$envioDatos); // Display the request and response print "<pre>\n"; print "Request :\n".htmlspecialchars(\$oSoap- >__getLastRequest())."\n"; print "Response:\n".htmlspecialchars(\$oSoap- >__getLastResponse())."\n"; print "</pre>"; /**/ //Hasta aqui if (strtoupper(substr(trim(\$envioDatos),0,4)) == "SOAP" or strlen(\$envioDatos)<1000) { throw new SoapFault('Server', 'Error SOAP: ' . \$envioDatos); return "0"; } return \$envioDatos; } catch (SoapFault \$e) { //Captura los errores var_dump(\$e); // printf("No se generó correctamente el archivo PDFs."); return "0"; } }</pre>	<p>//consumo de servicio web, hacemos referencia al servidor donde se encuentra publicado el servicio</p> <p>Verificamos el documento</p> <p>llamamos a la funcion html_a_pdf</p> <p>lo que recibimos es el pdf</p>

<pre> } } } </pre>	
--------------------------------	--

Tabla 23.- función de consumo de servicio web

12. TIPOS DE DOCUMENTOS

quipux/plantillas/generar_documento.php

La generación de documentos está realizado con configuraciones en la base de datos exclusivamente de la tabla tiporad:

trad_codigo	trad_descr	trad_tipo	trad_inst_c	trad_estado	trad_abrevi	trad_opc_ir	trad_texto	trad_formato	trad_forma
[PK] numer	character v	character v	integer	smallint	character v	character v	character v	character varying	smallint
1	Oficio	S	0	1	0	opc_imp_o	De mi cons Con sentir	**QUIPUX_DATOS_DOC_ASUNTO** **QUIPUX_DATOS_DOC_DESTINATARIO** **QUIPUX_DATOS_DOC_LUGAR_DESTINATARIO** **QUIPUX_DATOS_DOC_CUERPO** **QUIPUX_DATOS_DOC_JUSTIFICAR_FIRMA_INICIO** **QUIPUX_DATOS_DOC_DESPEDIDA** **QUIPUX_DATOS_DOC_FRASE_REMITENTE** **QUIPUX_DATOS_DOC_FIRMA_DIGITAL** **QUIPUX_DATOS_DOC_REMITENTE** **QUIPUX_DATOS_DOC_JUSTIFICAR_FIRMA_FIN** **QUIPUX_DATOS_DOC_DESTINATARIO_AL_PIE** **QUIPUX_DATOS_DOC_LINEAS_ESPECIALES**	1

Figura 5.- configuración de oficio

Tal como tenemos la configuración del oficio, tenemos para todos los documentos, así reemplazamos en php los valores del campo **trad_formato**.

Código	Descripción
<pre> \$this->documento_html = str_replace("**QUIPUX_DATOS_DOC_DESTINATARIO**", \$this->datos_doc_destinatario, \$this->documento_html); </pre>	Los campos entre **, serán reemplazados por

<pre> \$this->documento_html = str_replace("***QUIPUX_DATOS_DOC_DESTINATARIO_AL_PIE**", \$this->datos_doc_destinatario_al_pie, \$this->documento_html); \$this->documento_html = str_replace("***QUIPUX_DATOS_DOC_LUGAR_DESTINATARIO**", \$this->datos_doc_lugar_destinatario, \$this->documento_html); \$this->documento_html = str_replace("***QUIPUX_DATOS_DOC_REMITENTE**", \$this- >datos_doc_remitente, \$this->documento_html); \$this->documento_html = str_replace("***QUIPUX_DATOS_DOC_ASUNTO**", \$this- >datos_doc_asunto, \$this->documento_html); \$this->documento_html = str_replace("***QUIPUX_DATOS_DOC_CUERPO**", \$this- >datos_doc_cuerpo, \$this->documento_html); \$this->documento_html = str_replace("***QUIPUX_DATOS_DOC_DESPEDIDA**", \$this- >datos_doc_despedida, \$this->documento_html); \$this->documento_html = str_replace("***QUIPUX_DATOS_DOC_FRASE_REMITENTE**", \$this->datos_doc_frase_remitente, \$this->documento_html); \$this->documento_html = str_replace("***QUIPUX_DATOS_DOC_FIRMA_DIGITAL**", \$this- >datos_doc_firma_digital, \$this->documento_html); \$this->documento_html = str_replace("***QUIPUX_DATOS_DOC_LINEAS_ESPECIALES**", \$this->datos_doc_lineas_especiales, \$this->documento_html); \$this->documento_html = str_replace("***QUIPUX_DATOS_DOC_JUSTIFICAR_FIRMA_INICIO**", "&nbsp;
", \$this- >documento_html); \$this->documento_html = str_replace("***QUIPUX_DATOS_DOC_JUSTIFICAR_FIRMA_FIN**", "&nbsp;
", \$this->documento_html); \$this->documento_html = str_replace("***QUIPUX_DATOS_DOC_DADO_EN**", \$this- >datos_doc_dado_en, \$this->documento_html); </pre>	los datos obtenidos de la clase.
--	----------------------------------

Generación de clave de usuario:

Utilizamos md5, además a este realizamos un script:

Por ejemplo:

Para un usuario que tenga clave: 123

el md5('123')= "202cb962ac59075b964b07152d234b70"

la clave para quipux es: 02cb962ac59075b964b07152d2