Guía de Usuario de XGAP

Índice general

Índice de figuras

Lista de ejemplos

nota
Este documento es un trabajo en progreso.

Capítulo 1

Introducción

XGAP significa ***X*ml — *G*enerador de *AP*licaciones**.

XGAP nació en el año 2003 en la Universidad UNICEN, como una herramienta para crear aplicaciones web para su intranet.

Al principio la plataforma era PHP y base de datos PostgreSQL. Actualmente, las aplicaciones generadas con XGAP funcionan con bases de datos PostgreSQL u Oracle, pero sería posible utilizar cualquiera de las bases de datos soportadas por ADOdb.

El objetivo principal de XGAP es asistir al programador en las tareas de codificación repetitivas y comunes; es por eso que XGAP se concentra más en especificar una aplicación que en programarla. Una aplicación web se programa con XGAP creando un conjunto de archivos XML que la describen. El generador de XGAP toma esos archivos XML y crea archivos PHP por medio de plantillas XSLT. Cuando el proceso termina, la aplicación se puede empezar a usar inmediatamente.

XGAP resuelve estas tareas:

- Creación de listados, formularios de alta/modificación/borrado para las tablas de la base de datos, formularios maestro/detalle, gráficos
- Listados de selección emergentes
- Paginación automática
- Filtros de búsqueda básicos o avanzados
- Generadores automáticos de reportes (ODT, PDF, RTF, CSV)
- Detección automatica de tipos desde la base de datos
- Detección automática de campos requeridos en la base de datos
- Validación del lado del cliente y/o del servidor para la entrada de usuario
- Se incluye un editor de texto con formato
- Seguridad a nivel aplicación (usuarios, roles, permisos por rol)
- Puntos de extensión de código, útiles para resolver necesidades específicas de la aplicación y reglas del negocio
- Y muchas más...

aviso
Actualmente XGAP *no* está internacionalizado ni localizado. Todos sus mensajes están escritos en español y asume regionalización argentina.

Capítulo 2

Requerimientos

- PHP 5.2 o 5.3. No está comprobada la compatibilidad con 5.4+ (ver Ticket #80, Ticket #112, Ticket #117).
- Extensiones PHP
  - DOM
  - Fileinfo, si se usa `FormFileUpload::tipo()`
  - Filter
  - GD, si se usa la clase `Imagen`
  - Hash, si se usa la implementación por defecto de firmas digitales, o la clase `XgapCsrf`
  - JSON
  - Mail, si se envían correos electrónicos con las aplicaciones generadas
  - Mcrypt, si no está disponible la extensión OpenSSL
  - Multibyte String
  - OpenSSL, si se usa la implementación por defecto de firmas digitales, o las clases `XgapRand` o `XgapCsrf`
  - PCRE
  - Sessions
  - SPL
  - XML Parser
  - XMLWriter
- Extensiones PHP sólo para generación
  - XSL
- Oracle o PostgreSQL durante la ejecución de las aplicaciones generadas; no es necesaria una base de datos durante la generación

A partir de la versión 2.16.0.643, XGAP provee una página que permite comprobar la compatibilidad del entorno con estos requerimientos: `/test_reqs.php`. En el pie de la página de generación se incluyen automáticamente links que llevan a esta página, uno por cada versión de motor detectada.

Parte I

CÓMOs

Capítulo 3

Creación de una nueva aplicación

1. Crear un directorio con el nombre de la nueva aplicación dentro del directorio de aplicaciones (`APPS_DIR` en `xgap.php`).
2. Crear un directorio con nombre 'extras' dentro del nuevo directorio. *Importante:* el servidor web debe poder crear y modificar archivos dentro de estos dos directorios; caso contrario, se producirá un error al realizar la generación.
3. Crear la base de datos que se va a usar en la aplicación.
4. Copiar a la aplicación los scripts SQL provistos por XGAP para crear la estructura inicial requerida en la base de datos. Éstos se encuentran en la distribución de XGAP dentro del directorio `XGAP_DIR/MOTORES_SUBDIR/{motor}/bd/`, con nombre `/^\d\d-.+\.sql$/`.
5. El script `02-seguridad_datos_iniciales.sql` contiene secciones marcadas con "TODO:" que se deben adecuar a la aplicación. Realizar las modificaciones necesarias y ejecutar los scripts dentro de la base de datos recién creada, en orden ascendente por nombre.
6. Opcionalmente, copiar a la aplicación el menú inicial provisto por el motor: copiar `XGAP_DIR/MOTORES_SUBDIR/{motor}/xml/menu/main_menu.xml` a `APPS_DIR/{app}/extras/menu/`.
7. Abrir la página inicial del generador. Ingresar el nombre de la nueva aplicación en el campo "Aplicación" y presionar el botón "Configurar".
8. Dado que la nueva aplicación aún no tiene un archivo de configuración, el generador ofrece crear uno. En la nueva página, seleccionar en la lista desplegable "Versión" la versión del motor a utilizar y presionar el botón "Aceptar".
9. El generador muestra la página de configuración de la aplicación. Comprobar los valores existentes y completar los faltantes; presionar "Aceptar" para guardar los cambios.
10. El generador muestra la configuración que fue guardada. Hacer click en "Volver al generador".
11. En la página principal del generador, asegurarse que esté marcado "Generar aplicación" y presionar el botón "Generar".
12. Al completar el proceso, la aplicación inicial queda lista para usar. La primera generación copia algunos archivos a los directorios raíz y extras de la aplicación, que contienen configuraciones y definiciones de páginas básicas. Opcionalmente se pueden editar estos nuevos archivos para adaptarlos a las necesidades de la aplicación; al menos es conveniente revisar el archivo `extras/aplicacion.xml`, ya que se crea con valores por defecto.

Ejemplo 3.1 Creación de la aplicación "prueba1"

Este ejemplo asume que se está trabajando en Linux; será necesario adaptar algunos comandos y rutas para aplicarlo en otros sistemas operativos.
Comencemos por convenir algunos valores que se usarán durante el ejemplo:

- `XGAP_DIR ='/home/user/Projects/xgap/dist'`

- `MOTORES_SUBDIR ='motores'`
- `APPS_DIR ='/home/user/Projects/xgap/apps'`
- `XGAP_CONF_VERSION_XGAP ='ultimo'`
- Nombre de la aplicación: `prueba1`
- Base de datos de la aplicación: `prueba1_dev`
- Grupo al que pertenece el servidor web: `www-data`
- Directorio donde se guardan los scripts SQL para la aplicación: `/home/user/Projects/xgap/apps/prueba1/bd`

Entonces:

1. Crear el directorio `/home/user/Projects/xgap/apps/prueba1`:

```
cd /home/user/Projects/xgap/apps
mkdir prueba1
```

2. Crear el directorio `/home/user/Projects/xgap/apps/prueba1/extras`. Asegurarse que el servidor web tenga permiso de escritura sobre estos nuevos directorios. Una forma de hacerlo en Linux consiste en cambiar el grupo de los directorios al mismo que tiene el servidor web y darles permiso de escritura para el grupo:

```
mkdir prueba1/extras
chgrp -R www-data prueba1
chmod -R g+sw prueba1
```

3. Crear la base de datos `prueba1_dev`; por ejemplo, para PostgreSQL:

```
createdb -U postgres prueba1_dev 'Base de datos para la aplicación XGAP prueba1'
```

4. Copiar a la aplicación los scripts SQL de inicialización:

```
cd /home/user/Projects/xgap/apps/prueba1
mkdir bd
cd bd
cp /home/user/Projects/xgap/dist/motores/ultimo/bd/??-*.sql .
```

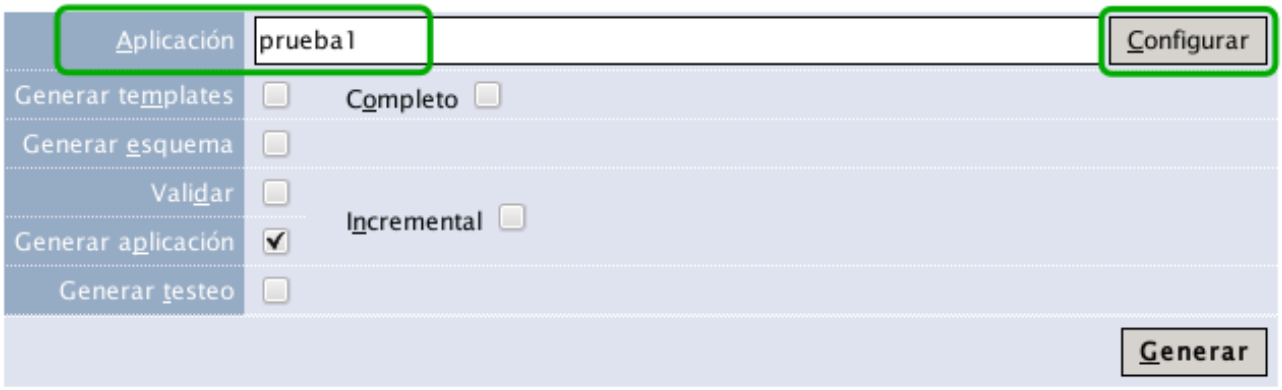
5. Modificar `02-seguridad_datos_iniciales.sql` y cargar la estructura inicial de la base de datos:

```
cd /home/user/Projects/xgap/apps/prueba1/bd
vi 02-seguridad_datos_iniciales.sql
psql -U postgres -f 00-superuser_init.sql prueba1_dev
psql -U postgres -f 01-seguridad.sql prueba1_dev
psql -U postgres -f 02-seguridad_datos_iniciales.sql prueba1_dev
psql -U postgres -f 03-firma_digital.sql prueba1_dev
```

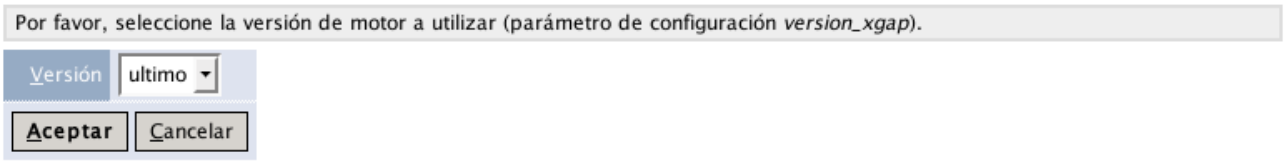
6. Copiar el menú inicial a la aplicación:

```
cd /home/user/Projects/xgap/apps/prueba1/extras
mkdir menu
cp /home/user/Projects/xgap/dist/motores/ultimo/xml/menu/main_menu.xml menu/
```

7. En la página inicial del generador, ingresar “prueba1” en el campo “Aplicación” y presionar “Configurar”.

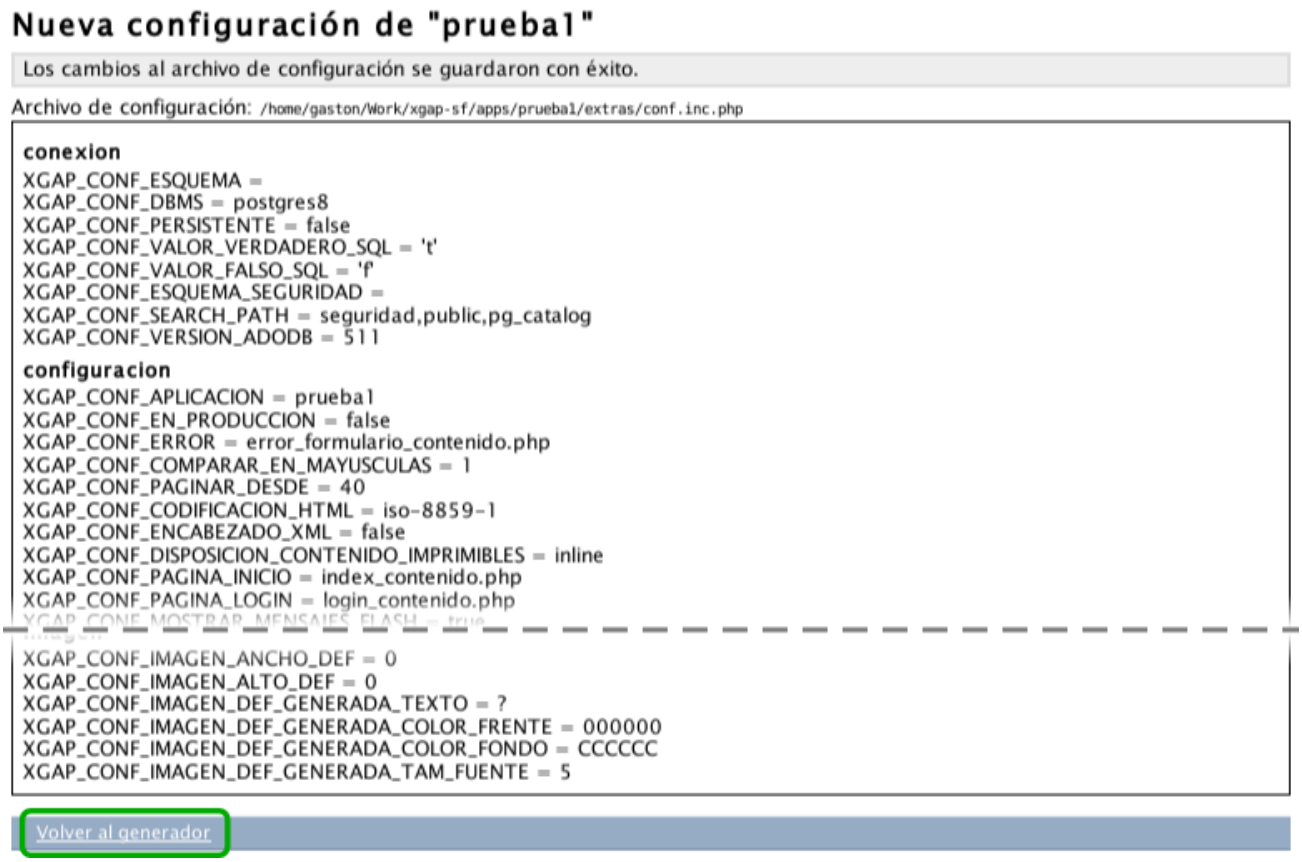


8. Seleccionar el motor a utilizar y presionar “Aceptar”.

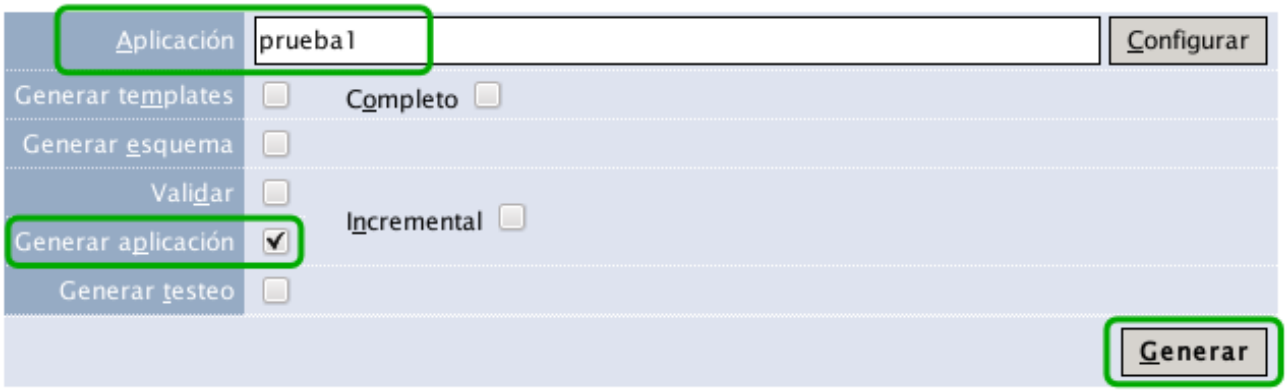


9. Realizar los cambios necesarios en la página de configuración y presionar “Aceptar”.

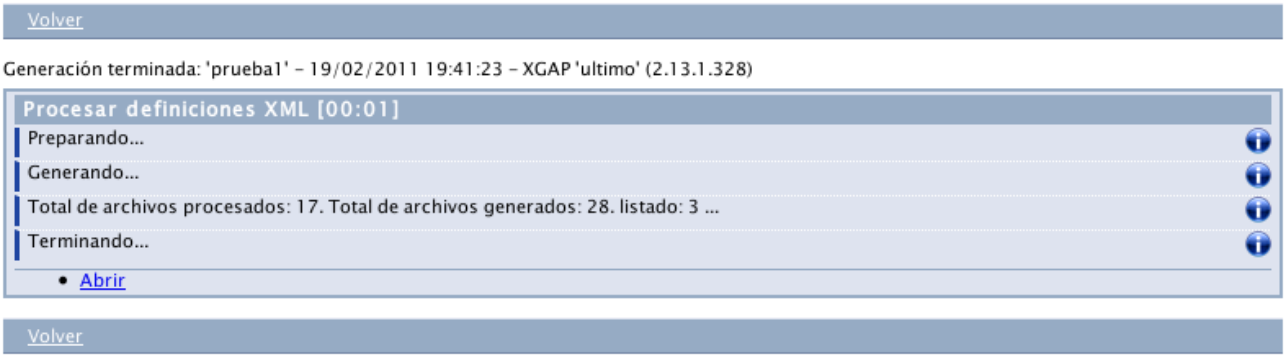
10. Se muestra el contenido del archivo de configuración de la aplicación. Volver al generador.



11. Generar la aplicación.



12. La aplicación inicial queda generada.



Parte II

Administración

Capítulo 4

Modo de mantenimiento

El *modo de mantenimiento* se puede establecer en las aplicaciones generadas para evitar que los usuarios accedan a sus páginas. Cuando los usuarios intentan acceder a las páginas de la aplicación mientras ésta se encuentra en modo de mantenimiento, en su lugar obtienen una página o mensaje que les informa la situación.

Es posible configurar una lista de roles funcionales que deben tener acceso irrestricto y una lista de páginas que deben estar disponibles para todos los usuarios aún en este modo.

Establecer modo de mantenimiento

Existen dos formas de poner una aplicación en modo de mantenimiento:

- **Crear en la raíz de la aplicación un archivo con nombre `maintenance.html`.** Cuando este archivo existe y es legible, se retorna su contenido en vez de la página solicitada. Los parámetros de configuración que afectan al modo de mantenimiento no se tienen en cuenta si se usa este archivo.
- **Establecer el parámetro de configuración `en_mantenimiento` con valor `true`.** En este caso, en vez de la página solicitada, se retorna la indicada por el parámetro de configuración `pagina_mantenimiento`.

La primera forma tiene prioridad sobre la segunda.

Más específicamente, cuando la aplicación está en modo de mantenimiento, la respuesta sigue las siguientes reglas:

- Si la página solicitada o el rol funcional actual tienen acceso libre, de acuerdo a los parámetros de configuración (`paginas_accesibles_en_mantenimiento` o `rolfs_acceso_en_mantenimiento`, respectivamente), se agrega un mensaje de aviso y se continúa el procesamiento normal sin más interferencia.
- Si no, se responde con la primera de las siguientes opciones que cumpla las condiciones indicadas:
  1. Redirección a `maintenance.html`, si existe y la página solicitada produciría respuesta HTML.
  2. Redirección a la página de mantenimiento configurada (parámetro de configuración `pagina_mantenimiento`), si existe y la página solicitada produciría respuesta HTML.
  3. Estado HTTP 503 (Service Unavailable), con el valor del parámetro de configuración `mensaje_mantenimiento` en el cuerpo de la respuesta, en cualquier otro caso.

Configuración

Los siguientes parámetros de configuración afectan al modo de mantenimiento:

en_mantenimiento
Indica si la aplicación está o no en modo de mantenimiento.

pagina_mantenimiento
Nombre de la página que se debe retornar al usuario cuando la aplicación está en modo de mantenimiento.

mensaje_mantenimiento
Mensaje a mostrar cuando la aplicación está en modo mantenimiento. Se usa en la página de mantenimiento por defecto provista por XGAP o cuando se retorna con código HTTP 503.

paginas_accesibles_en_mantenimiento
Lista de páginas accesibles por cualquier usuario cuando la aplicación está en modo de mantenimiento.

rolfs_acceso_en_mantenimiento
Lista de roles funcionales que deben poder acceder a todas las páginas de la aplicación cuando está en modo de mantenimiento.

Parte III

Referencia

Capítulo 5

Tipos de página/documento

Fuentes

En esta sección se describen los distintos tipos de archivos fuente XML que soporta XGAP.

Tipo	Sufijo	Definición [1]	Genera	Descripción
Listado	_listado.xml	listado.xsd	Listado normal, Listado para selección, Listado para selección múltiple, Listado PDF, Listado ODS, Listado RTF, Listado Excel, Listado CSV, Gráfico simple, Listado para master	Listado de entidades.
Formulario	_formulario.xml	formulario.xsd	Formulario normal, Reporte de formulario	Formulario de alta/-baja/modificación de una entidad.
Master/Detalle	_master.xml	formulario.xsd	Master/Detalle	Detalle de una entidad.
Contenido	_contenido.xml	contenido.xsd	Contenido	Página con contenido personalizado.
Menú	_menu.xml	menu.xsd	Menú	Menú de la aplicación.
Gráfico	_grafico.xml	grafico.xsd	Gráfico	Gráfico personalizado a página completa.
Reporte ODT	_reporte_odt.xml	reporte_odt.xsd	Reporte ODT	Reporte personalizado en formato ODT.
Reporte PDF	_reporte_pdf.xml	reporte_pdf.xsd	Reporte PDF	Reporte personalizado en formato PDF.
Reporte RTF	_reporte_rtf.xml	—	Reporte RTF	Reporte personalizado en formato RTF.

[1] *Esquema XML* que define la estructura del archivo fuente

Tipos generados

En esta sección se describen los distintos tipos de páginas o documentos que puede generar XGAP a partir de los fuentes XML detallados en la sección anterior.

Tipo	Sufijo	Descripción
Listado normal	_listado.php	Página HTML que contiene un listado de entidades en forma de tabla.
Listado para selección	_listado_seleccion.php	Página HTML que contiene un listado de entidades en forma de tabla, usado para seleccionar una de ellas.
Listado para selección múltiple	_listado_seleccion_m.php	Página HTML que contiene un listado de entidades en forma de tabla, usado para seleccionar una o más de ellas.
Listado PDF	_listado_pdf.php	Exportación en formato PDF de un listado de entidades en forma de tabla.
Listado ODS	_listado_ods.php	Exportación en formato ODS de un listado de entidades en forma de tabla.
Listado RTF	_listado_rtf.php	Exportación en formato RTF de un listado de entidades en forma de tabla.
Listado Excel	_listado_xls.php	Exportación en formato Excel de un listado de entidades en forma de tabla.
Listado CSV	_listado_csv.php	Exportación en formato CSV de un listado de entidades en forma de tabla.
Gráfico simple	_listado_grafico_simple.php	Imagen de un gráfico generado a partir de los datos de un listado.
Listado para master	_master_inline.php	Fragmento de página HTML que contiene un listado de entidades en forma de tabla, destinado a ser embebido en un master.
Formulario normal	_formulario.php	Página HTML que contiene un formulario destinado a realizar la alta, baja o modificación de una entidad.
Reporte de formulario	_formulario_reporte.php	Página HTML que contiene un formulario de una entidad, presentado en formato de reporte (sólo lectura).
Master/Detalle	_master.php	Página HTML que contiene el detalle de una entidad.
Contenido	_contenido.php	Página HTML con contenido personalizado.
Menú	_menu.php	Definición XML de un menú de la aplicación, usado como entrada para un componente javascript.
Gráfico	_grafico.php	Página HTML que contiene un gráfico personalizado.
Reporte ODT	_reporte_odt.php	Reporte personalizado en formato ODT.
Reporte PDF	_reporte_pdf.php	Reporte personalizado en formato PDF.
Reporte RTF	_reporte_rtf.php	Reporte personalizado en formato RTF.

Reportes ODT

Emisión en formato PDF

Los reportes ODT se pueden emitir en formato PDF, convirtiendo el ODT generado a PDF mediante un comando externo y enviando al cliente el documento resultante. Para ello debe estar definida la constante PHP `XGAP_COMANDO_GENERACION_PDF` y su valor debe ser el comando a ejecutar.

Cuando la constante está definida, se verifican otros dos valores para determinar, en cada solicitud, si la conversión se debe realizar o no:

- Una variable PHP global con nombre `$emitir_pdf`. Si está definida, su valor booleano determina si la conversión se efectúa (`true`) o no (`false`).
- Un parámetro de request con nombre `xgap_conv`. La conversión se efectúa si el parámetro *no* está presente o tiene valor `'pdf'`, y no si tiene cualquier otro valor. Este parámetro sólo se toma en consideración cuando `$emitir_pdf` *no* está definida.

Nótese que el comportamiento predeterminado cuando está definida la constante `XGAP_COMANDO_GENERACION_PDF`, si no se utiliza la variable `$emitir_pdf` ni el parámetro `xgap_conv`, es realizar la conversión.

El proceso se detalla en la figura siguiente.

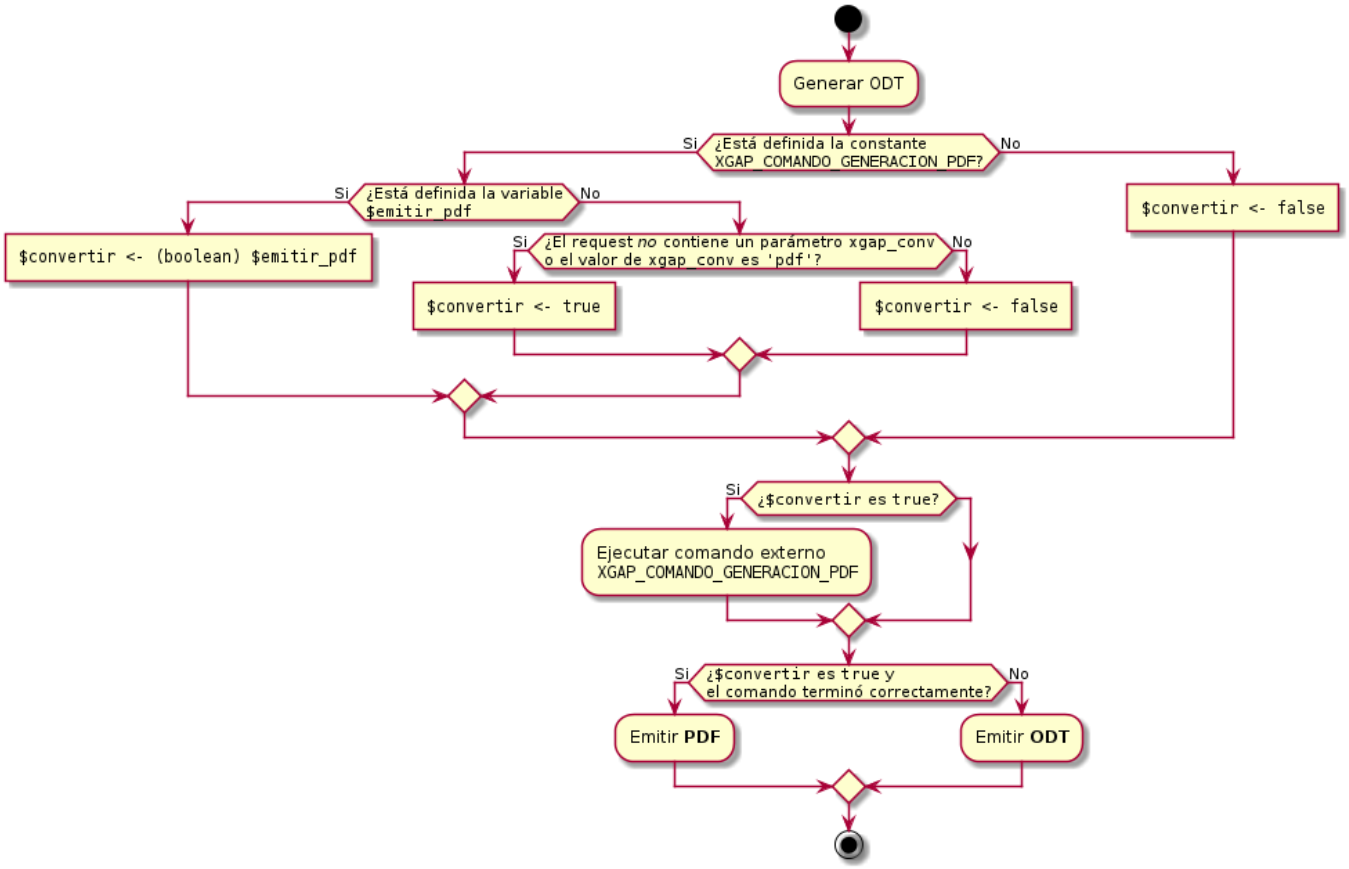


Figura 5.1: Proceso de emisión de un reporte ODT

---

nota
La funcionalidad descripta está definida en el código de salida predeterminado, es decir que no se aplica en reportes que usan salida personalizada (`/odt/configuracion/salida`).

El comando de conversión

El valor de la constante `XGAP_COMANDO_GENERACION_PDF` debe ser un string que se ejecuta como un comando externo. Este string puede incluir, opcionalmente, los marcadores `{in}` y `{out}` para indicar las posiciones donde se deben insertar los nombres de los archivos de entrada (ODT) y de salida (PDF), respectivamente. Si no se incluye el marcador `{in}`, el nombre del archivo de entrada se agrega al final del comando. Si no se incluye el marcador `{out}`, el nombre del archivo de salida no se incorpora al comando y se asume que éste produce un archivo con el mismo nombre que el de entrada, en el mismo directorio, con la extensión `.odt` reemplazada por `.pdf`.

Notas

- Tener en cuenta que el comando es ejecutado por el usuario de sistema con el que corre el servidor web; se debe asegurar que este usuario cuente con los permisos necesarios para poder completar correctamente la ejecución.
- El comando se ejecuta como está definido, sin aplicar filtros ni escapar metacaracteres del shell, para que sea posible usar comandos arbitrarios. Es responsabilidad del programador o administrador (si la constante se especifica en la configuración de la aplicación) garantizar que el comando sea seguro y no contenga datos ingresados sin filtrar por el usuario usuario.

Ejemplo 5.1 Ejemplos de definición del comando de conversión ODT → PDF

- **Opción 1**: unoconv.

```
<?php define('XGAP_COMANDO_GENERACION_PDF',
        'env HOME=/tmp /usr/bin/unoconv --timeout=10 -f pdf -e SelectPdfVersion=1 -e  ↩
            ExportBookmarks=true -e InitialView=0 -e PageLayout=0');
```

Si unoconv ya está corriendo en modo listener, lo utiliza. Si no, inicia una nueva instancia de soffice en cada invocación.

Para que el comando unoconv ejecutado sea capaz de comunicarse con una instancia corriendo en modo servidor, es posible que esta última deba estar iniciada por el mismo usuario que corre el servidor web; por ejemplo:

```
sudo -u www-data HOME=/tmp unoconv --listener &
```

- **Opción 2**: PyODConverter y soffice en modo servidor.

```
<?php define('XGAP_COMANDO_GENERACION_PDF',
        'python "' . OUT_DIR . DIR_SEP . XGAP_CONF_APLICACION . DIR_SEP . 'PyODConverter. ↩
            py" {in} {out}');
```

Requiere:

- El paquete Python PyODConverter.
- Un script Python que invoque a `PyODConverter.DocumentConverter.convert()` usando los nombres de archivo que recibe como parámetro (llamado `PyODConverter.py` en este ejemplo).
- Una instancia de soffice aceptando pedidos o de unoconv en modo listener (mencionado anteriormente). Una forma de iniciar soffice para que acepte pedidos es la siguiente:

```
sudo -u www-data HOME=/tmp soffice --headless --accept="socket,host=127.0.0.1,port=2002; ↩
    urp;" &
```

- **Opción 3**: soffice con una instancia iniciada por cada invocación

```
<?php define('XGAP_COMANDO_GENERACION_PDF',
        'env HOME=/tmp /usr/bin/libreoffice -env:HOME=/tmp --headless --invisible -- ↩
            convert-to pdf');
```

o:

```
<?php define('XGAP_COMANDO_GENERACION_PDF',
        'export HOME=/tmp; /usr/bin/libreoffice --headless --invisible --convert-to pdf');
```

Capítulo 6

Código extra en las páginas generadas

El elemento `codigoExtra/codigo` permite definir código a incluir en las páginas generadas. El atributo `@tipo` indica el tipo de código; puede ser `HTML`, `PHP` o `JAVASCRIPT`. El atributo `@ubicacion` indica la ubicación dentro de la página donde se debe emitir el código; más adelante se encuentran listadas todas las ubicaciones disponibles para cada tipo de página. En el caso de `@tipo="PHP"`, el código se emite dentro de una función que se define al comienzo de la página y se invoca en la ubicación indicada; esta función recibe un único parámetro `$params`, el cual es un array que contiene los pares `clave=>valor` indicados en la columna "Parámetros" de los listados de ubicaciones. Dado que el código extra no tiene alcance global, no es necesario preocuparse por modificar inadvertidamente el valor de alguna variable usada por XGAP.

Ejemplo 6.1 Ejemplo de uso de código extra

```
<codigoExtra>
    <codigo tipo="HTML" ubicacion="inicio_body">
        <![CDATA[
        <div class="mensaje">Esto va al inicio de la pagina.</div>
        ]]>
    </codigo>
    <codigo tipo="JAVASCRIPT" ubicacion="fin_body">
        <![CDATA[
        alert("Mensaje javascript.");
        ]]>
    </codigo>
    <codigo tipo="PHP" ubicacion="despues_inicializacion">
        <![CDATA[
        $conexion = $params['conexion'];
        $idseccion = $_REQUEST['idseccion'];
        $sql = "SELECT dvalortraducido FROM traduccion"
             . " WHERE cnombre = 'SECCION' AND cvalor = '$idseccion'";
        $descseccion = $conexion->obtenerPrimero($sql);
        // $params['titulo'] es un parámetro provisto para esta ubicación
        $params['titulo'] .= ' en ' . XGAP_CONF_APLICACION . " | $descseccion";
        ]]>
    </codigo>
    <!-- Inválido: ya existe un bloque de código en esta ubicación
        y con este tipo -->
    <codigo tipo="PHP" ubicacion="despues_inicializacion">
        <![CDATA[
        $otra_mas = $_REQUEST['otra_mas'];
        ]]>
    </codigo>
</codigoExtra>
```

Ubicaciones

Posibles valores para `codigoExtra/codigo/@ubicacion`.

En todos los tipos de página

@ubicacion	**Ubicación en la página**	**Parámetros (@tipo="PHP")**
`inicio_pagina`	Al comienzo de la página, antes de inicializar XGAP. (XGAP $\geq$ 2.11.2)	-
`antes_inicializacion`	Antes de inicializar las variables de la página	-
`xgap_cargado`	Luego de cargar el entorno de XGAP, pero antes de definir las variables de la página. (XGAP $\geq$ r2.12.10.245)	`Conexion 'conexion' iSeguridad 'seguridad' HistorialNavegacion 'historial' Configuracion 'configuracion'`

nota
El caso particular de `@ubicacion="inicio_pagina"` sólo se aplica cuando `@tipo="PHP"`. El código en esta ubicación no se emite dentro de una función, sino a nivel global. Esto permite definir elementos globales, como ser constantes, funciones o clases, lo que no sería posible o no tendría sentido dentro de una función.

En `listado`, `listado_seleccion`, `listado_seleccion_m` y `listado_master_inline`

Referencias para la columna "En":

l	`listado`
s	`listado_seleccion`
sm	`listado_seleccion_m`
mi	`listado_master_inline`

@ubicacion	**Ubicación en la página**	**Parámetros (@tipo='PHP')**	**En**
`despues_inicializacion`	Después de inicializar las variables de la página; justo antes de comenzar a emitir a la salida	`Conexion 'conexion' ADORecordSet 'recordset' HistorialNavegacion 'historial' Flash 'flash' string &'titulo'`	l, s, sm, mi

@ubicacion	**Ubicación en la página**	**Parámetros (@tipo='PHP')**	**En**
despues_html	Después de terminar de generar salida HTML (después de </html>)	Conexion 'conexion' HistorialNavegacion 'historial'	l, s, sm
en_head	Antes de generar el elemento head. Lo que se emite a la salida se incluye dentro del head.	-	l, s, sm
inicio_body	Justo después de abrir el elemento body	Conexion 'conexion' HistorialNavegacion 'historial' boolean 'pagina_simple'	l, s, sm
fin_body	Justo antes de cerrar el elemento body	Conexion 'conexion' boolean 'pagina_simple'	l, s, sm
despues_abrir_estructura	Justo después de abrir la estructura de la página (al comienzo del contenedor principal)	Conexion 'conexion' HistorialNavegacion 'historial' Flash 'flash' boolean 'pagina_simple'	l, s, sm
antes_cerrar_estructura	Justo antes de cerrar la estructura de la página (al final del contenedor principal)	Conexion 'conexion' HistorialNavegacion 'historial' boolean 'pagina_simple'	l, s, sm
antes_tabla_datos	Justo antes de abrir la tabla principal del listado	Conexion 'conexion' string 'id_tabla'	l, s, sm, mi
despues_tabla_datos	Justo después de cerrar la tabla principal del listado	Conexion 'conexion' string 'id_tabla'	l, s, sm, mi
inicio_celda_encabezado	Justo después de abrir una celda del encabezado de la tabla principal	Conexion 'conexion' string 'dato' string &'valor_a_mostrar' (XGAP $\geq$ r2.16.0.568)	l, s, sm, mi
fin_celda_encabezado	Justo antes de cerrar una celda del encabezado de la tabla principal	Conexion 'conexion' string 'dato'	l, s, sm, mi

@ubicacion	**Ubicación en la página**	**Parámetros (@tipo='PHP')**	**En**
antes_fila_datos	Antes de abrir una fila del cuerpo de la tabla principal (XGAP ≥ r2.16.0.700)	Conexion 'conexion' integer 'cant_filas': cantidad total de filas en el cuerpo integer 'fila': número de fila actual array 'valores_fila': valores de todas las columnas de la fila actual string &'nombre': ID de la fila string &'clase': clase de la fila	l, s, sm, mi
inicio_celda_datos	Justo después de abrir una celda del cuerpo de la tabla principal	Conexion 'conexion' string 'dato' integer 'fila' array 'valores_fila' string &'valor_a_mostrar'	l, s, sm, mi
fin_celda_datos	Justo antes de cerrar una celda del cuerpo de la tabla principal	Conexion 'conexion' string 'dato' integer 'fila' array 'valores_fila' string 'valor_mostrado'	l, s, sm, mi
antes_consulta	Antes de crear la consulta SQL para obtener los datos para el listado. (XGAP ≥ 2.12.3)	Conexion 'conexion' HistorialNavegacion 'historial'	l, s, sm, mi
despues_consulta	Despues de ejecutar la consulta SQL para obtener los datos para el listado. (XGAP ≥ r2.16.0.732)	Conexion 'conexion' HistorialNavegacion 'historial' ADORecordSet 'recordset' int 'total_filas': cantidad total de filas obtenidas por la consulta, sin tener en cuenta paginación string 'sql_filas': código SQL para obtener los datos de las filas string 'sql_total_filas': código SQL para obtener el valor de 'total_filas'	l, s, sm, mi

@ubicacion	**Ubicación en la página**	**Parámetros (@tipo='PHP')**	**En**
crea_consulta	Permite crear manualmente la consulta SQL para obtener los datos para el listado. Debe establecer los valores de $params['SQL'] y $params['SQLCount']	Conexion 'conexion' string &'SQL' int &'SQLCount' string 'select' string 'from' string 'where' string 'orderBy' string 'groupBy' string 'having' array 'filtros2' array 'columnasFiltros' string 'condiciones' string 'paramsAExluir' string 'condiciones_personalizadas' string 'orden'	l, s, sm, mi
arma_link	Antes de generar un link en una columna que tiene @llevaALink	Conexion 'conexion' ADORecordSet 'recordset' string 'dato' int 'fila' string &'paginaForm': el link array &'paramPagina': parámetros para agregar al link	l, mi

En `listado_csv`, `listado_ods`, `listado_pdf`, `listado_rtf` y `listado_xls`

@ubicacion	**Ubicación en la página**	**Parámetros (@tipo='PHP')**
despues_inicializacion	Después de inicializar las variables de la página; justo antes de comenzar a emitir a la salida	Conexion 'conexion' ADORecordSet 'recordset' (XGAP $\geq$ 2.12.3) HistorialNavegacion 'historial' string &'titulo' string &'nombre_archivo'
antes_consulta	Antes de crear la consulta SQL para obtener los datos para el listado.	Conexion 'conexion' HistorialNavegacion 'historial'

@ubicacion	Ubicación en la página	Parámetros (@tipo='PHP')
despues_consulta	Despues de ejecutar la consulta SQL para obtener los datos para el listado. En listado_csv y listado_xls la consulta no se ejecuta, por lo que recordset es nulo y total_filas es 0.	Conexion 'conexion' HistorialNavegacion 'historial' ADORecordSet 'recordset' int 'total_filas': cantidad total de filas obtenidas por la consulta, sin tener en cuenta paginación string 'sql_filas': código SQL para obtener los datos de las filas string 'sql_total_filas': código SQL para obtener el valor de 'total_filas'

En contenido

@ubicacion	Ubicación en la página	Parámetros (@tipo='PHP') [1]
despues_inicializacion	Después de inicializar las variables de la página; justo antes de comenzar a emitir a la salida	Conexion 'conexion' HistorialNavegacion 'historial' Flash 'flash' string &'titulo'
despues_html	Después de terminar de generar salida html (después de </html>)	Conexion 'conexion' HistorialNavegacion 'historial'
en_head	Antes de generar el elemento head. Lo que se emite a la salida se incluye dentro del head.	-
inicio_body	Justo después de abrir el elemento body	Conexion 'conexion' HistorialNavegacion 'historial' boolean 'pagina_simple'
fin_body	Justo antes de cerrar el elemento body	Conexion 'conexion' boolean 'pagina_simple'
despues_abrir_estructura	Justo después de abrir la estructura de la página (al comienzo del contenedor principal)	Conexion 'conexion' HistorialNavegacion 'historial' Flash 'flash' boolean 'pagina_simple'
antes_cerrar_estructura	Justo antes de cerrar la estructura de la página (al final del contenedor principal)	Conexion 'conexion' HistorialNavegacion 'historial' boolean 'pagina_simple'

En formulario

[1] El parámetro 'conexión' tiene valor null en todas las ubicaciones si /pagina/@con-conexion ='false'.

@ubicacion	**Ubicación en la página**	**Parámetros (@tipo='PHP')**
antes_procesar_uploads	Antes de procesar los campos de tipo "Archivo". (XGAP ≥ r2.16.0.649) ref	Conexion 'conexion' Seguridad 'seguridad' HistorialNavegacion 'historial' Configuracion 'configuracion' string 'accion' array 'metadatos' array 'campos' array &'registro' Flash 'flash'
despues_obtener_vbles_sistema	Después de inicializar las variables de la página, pero antes de comenzar a procesar la acción o la presentación de la página (XGAP ≥ r2.16.0.678)	Conexion 'conexion' iSeguridad 'seguridad' HistorialNavegacion 'historial' Configuracion 'configuracion' boolean 'agregado'
despues_inicializacion	Después de inicializar las variables de la página, justo antes de comenzar a emitir a la salida; no se llega a esta ubicación si el formulario ejecuta correctamente una acción, dado que se redirige a la página destino antes; es decir, sólo se llega si el formulario se muestra	Conexion 'conexion' HistorialNavegacion 'historial' Flash 'flash' boolean 'agregado' ADORecordSet 'recordset': el registro completo obtenido desde la base de datos; sólo está disponible si $params['agregado'] == false; en caso contrario, es null (XGAP ≥ r2.16.0.736) string &'titulo' array &'valores' boolean &'sololectura'
despues_html	Después de terminar de generar salida html (después de </html>)	Conexion 'conexion' HistorialNavegacion 'historial'
en_head	El contenido sale dentro del elemento head	-
inicio_body	Justo después de abrir el elemento body	Conexion 'conexion' HistorialNavegacion 'historial' boolean 'pagina_simple'
fin_body	Justo antes de cerrar el elemento body	Conexion 'conexion' boolean 'pagina_simple'
despues_abrir_estructura	Justo después de abrir la estructura de la página (al comienzo del contenedor principal)	Conexion 'conexion' HistorialNavegacion 'historial' Flash 'flash' boolean 'pagina_simple'
antes_cerrar_estructura	Justo antes de cerrar la estructura de la página (al final del contenedor principal)	Conexion 'conexion' HistorialNavegacion 'historial' boolean 'pagina_simple'

@ubicacion	**Ubicación en la página**	**Parámetros (@tipo='PHP')**
antes_accion	Antes de ejecutar la acción, cualquiera sea. Esta ubicación se coloca justo antes que las ubicaciones antes_{insert,update,delete}. (XGAP $\geq$ r2.17.0.958)	Conexion 'conexion' Seguridad 'seguridad' HistorialNavegacion 'historial' Configuracion 'configuracion' string 'accion' string 'nombre_tabla' array 'metadatos' array &'registro' boolean &'realizar_operacion': inicialmente tiene valor true; si se cambia a false, no se realiza la operación y se continúa cargando el formulario actual
antes_insert	Antes de validar los datos y ejecutar la inserción en la BD	Conexion 'conexion' string 'nombre_tabla' array 'metadatos' array &'registro' boolean &'realizar_operacion': inicialmente tiene valor true; si se cambia a false, no se realiza la operación y se continúa cargando el formulario actual
despues_insert	Después de ejecutar la inserción en la BD	Conexion 'conexion' string 'nombre_tabla' array 'metadatos' Flash flash array &'registro' boolean &'realizar_redireccion': inicialmente tiene valor true; si se cambia a false, no se hace la redirección a la página de retorno y se continúa cargando el formulario actual
antes_update	Antes de validar los campos y ejecutar la actualización en la BD	Conexion 'conexion' string 'nombre_tabla' array 'metadatos' array &'registro' boolean &'realizar_operacion': inicialmente tiene valor true; si se cambia a false, no se realiza la operación y se continúa cargando el formulario actual

@ubicacion	**Ubicación en la página**	**Parámetros (@tipo='PHP')**
despues_update	Después de ejecutar la actualización en la BD	Conexion 'conexion' string 'nombre_tabla' array 'metadatos' Flash 'flash' array &'registro' boolean &'realizar_redireccion': inicialmente tiene valor true; si se cambia a false, no se hace la redirección a la página de retorno y se continúa cargando el formulario actual
antes_delete	Antes de ejecutar el borrado en la BD	Conexion 'conexion' string 'nombre_tabla' string 'clave' array 'metadatos' array &'registro' boolean &'realizar_operacion': inicialmente tiene valor true; si se cambia a false, no se realiza la operación y se continúa cargando el formulario actual
despues_delete	Después de ejecutar el borrado en la BD	Conexion 'conexion' string 'nombre_tabla' string 'clave' array 'metadatos' Flash flash array &'registro' boolean &'realizar_redireccion': inicialmente tiene valor true; si se cambia a false, no se hace la redirección a la página de retorno y se continúa cargando el formulario actual
antes_redirigir	Después de ejecutar la acción y antes de redirigir a la página destino. Sólo se ejecuta si la acción se completó correctamente. (XGAP $\geq$ r2.12.8.65)	string 'accion' array 'registro': El registro final sobre el que se operó. Es vacío si la acción es "cancelar". HistorialNavegacion 'historial' boolean 'redireccionar_master' Flash 'flash' boolean 'es_destino_predeterminado': ¿El destino es el que se sigue de manera predeterminada (true), o está dado en el request, via RequestXgap::PARAMETRO_PAG_RETORNO (false)? (XGAP $\geq$ r2.16.0.741) string &'destino': El URL al que se va a redirigir. Se puede modificar para cambiar el destino, o dejar con un valor vacío para cancelar la redirección.

@ubicacion	**Ubicación en la página**	**Parámetros (@tipo='PHP')**
despues_accion	Después de ejecutar la acción y justo antes de comenzar la carga del formulario. Esta ubicación se coloca después de la redirección que se realiza luego de la ejecución correcta de la acción, por lo que sólo se invoca si la acción no se completa correctamente o se cancela la redirección. (XGAP $\geq$ r2.16.1.794)	Conexion 'conexion' Seguridad 'seguridad' HistorialNavegacion 'historial' Configuracion 'configuracion' Flash 'flash' string 'accion' boolean 'accion_ok': ¿La acción se completó correctamente? array 'registro': El registro final sobre el que se operó. Es vacío si la acción es "cancelar".
antes_abrir_form	Antes de abrir el formulario principal	Conexion 'conexion' string 'nombre' array 'metadatos' array 'valores' boolean 'agregado'
despues_abrir_form	Después de abrir el formulario principal	Conexion 'conexion' string 'nombre' array 'metadatos' array 'valores' boolean 'agregado'
antes_cerrar_form	Antes de cerrar el formulario principal	Conexion 'conexion' string 'nombre' array 'metadatos' array 'valores' boolean 'agregado'
despues_cerrar_form	Después de cerrar el formulario principal	Conexion 'conexion' string 'nombre' array 'metadatos' array 'valores' boolean 'agregado'
antes_campo_link	Antes de generar un campo con @tipo='Link'. (XGAP $\geq$ 2.12.1)	Conexion 'conexion' HistorialNavegacion 'historial' array 'metadatos' (XGAP $\geq$ r2.16.0.584) array 'valores' (XGAP $\geq$ r2.16.0.584) boolean 'agregado' (XGAP $\geq$ r2.16.0.584) string 'id' string &'url' array &'parametros' string &'contenido' string &'click' string &'tip'

Referencias
El parámetro "accion", que se pasa en varias ubicaciones, puede tomar uno de los siguientes valores: { 'agregar' | 'agregar-volver' | 'agregar-seguir' | 'modificar' | 'borrar' | 'cancelar' }.
El parámetro "campos" de la ubicación "antes_procesar_uploads" es un array que contiene los datos de los campos de tipo "Archivo", en el formato:

```php
<?php
// Lista de campos indexada por elemento "campo/dato"
$params['campos'] = array(
    '{dato #1}' => array(
        // [string] nombre del archivo
        'nombrearchivo' => '{nombre archivo}',
        // [boolean|null] ¿se debe conservar el archivo original?
        // true: si; false; no; null: no aplicable
        'conservar' => $c,
        // [string] valor del elemento "campo/destino"
        'destino' => '{destino #1}',
        // [FormFileUpload] objeto que representa el upload
        'upload' => $u
    )
    //, '{dato #2}' => array(...), ...
);
```

En `formulario_master`

`@ubicacion`	Ubicación en la página	Parámetros (`@tipo='PHP'`)

`despues_inicializacion`	Después de inicializar las variables de la página; justo antes de comenzar a emitir a la salida	`Conexion 'conexion'`
		`HistorialNavegacion 'historial'`
		`Flash 'flash'`
		`ADORecordSet 'recordset'` (XGAP ≥ r2.16.1.813)
		`string &'titulo'`
		`array &'valores'`
		`boolean &'sololectura'`

`antes_html`	Antes de comenzar a generar salida html (antes de `<html>`, después de enviar los headers HTTP)	`Conexion 'conexion'` `HistorialNavegacion 'historial'`
`despues_html`	Después de terminar de generar salida html (después de `</html>`)	`Conexion 'conexion'` `HistorialNavegacion 'historial'`
`en_head`	El contenido sale dentro del elemento `head`	-

`inicio_body`	Justo después de abrir el elemento `body`	`Conexion 'conexion'`
		`HistorialNavegacion 'historial'`
		`boolean 'pagina_simple'`

`fin_body`	Justo antes de cerrar el elemento `body`	`Conexion 'conexion'` `boolean 'pagina_simple'`

@ubicacion	**Ubicación en la página**	**Parámetros (@tipo='PHP')**
despues_abrir_estructura	Justo después de abrir la estructura de la página (al comienzo del contenedor principal)	Conexion 'conexion' HistorialNavegacion 'historial' Flash 'flash' boolean 'pagina_simple'
antes_cerrar_estructura	Justo antes de cerrar la estructura de la página (al final del contenedor principal)	Conexion 'conexion' HistorialNavegacion 'historial' boolean 'pagina_simple'
antes_abrir_datos	Antes de abrir la estructura que contiene los datos de la entidad.	Conexion 'conexion' array 'metadatos' array 'valores'
despues_abrir_datos	Después de abrir la estructura que contiene los datos de la entidad.	Conexion 'conexion' array 'metadatos' array 'valores'
antes_cerrar_datos	Antes de cerrar la estructura que contiene los datos de la entidad.	Conexion 'conexion' array 'metadatos' array 'valores'
despues_cerrar_datos	Después de cerrar la estructura que contiene los datos de la entidad.	Conexion 'conexion' array 'metadatos' array 'valores'
antes_titulo_campo	Antes de mostrar el título de un campo.	Conexion 'conexion' string 'dato' string &'valor_a_mostrar'
despues_titulo_campo	Después de mostrar el título de un campo.	Conexion 'conexion' string 'dato' string 'valor_a_mostrar'
antes_valor_campo	Antes de mostrar el valor de un campo.	Conexion 'conexion' string 'dato' ADORecordSet 'recordset' array 'valores' string &'valorAMostrar'
despues_valor_campo	Después de mostrar el valor de un campo.	Conexion 'conexion' string 'dato' ADORecordSet 'recordset' array 'valores' string 'valorAMostrar'

En formulario_reporte

@ubicacion	**Ubicación en la página**	**Parámetros (@tipo='PHP')**
antes_html	Antes de comenzar a generar salida html (antes de <html>, después de enviar los headers HTTP)	Conexion 'conexion' HistorialNavegacion 'historial'
despues_html	Después de terminar de generar salida html (después de </html>)	Conexion 'conexion' HistorialNavegacion 'historial'

@ubicacion	Ubicación en la página	Parámetros (@tipo='PHP')
en_head	El contenido sale dentro del elemento head	-
inicio_body	Justo después de abrir el elemento body	Conexion 'conexion' HistorialNavegacion 'historial' boolean 'pagina_simple'
fin_body	Justo antes de cerrar el elemento body	Conexion 'conexion' boolean 'pagina_simple'
despues_abrir_estructura	Justo después de abrir la estructura de la página (al comienzo del contenedor principal)	Conexion 'conexion' HistorialNavegacion 'historial' Flash 'flash' boolean 'pagina_simple'
antes_cerrar_estructura	Justo antes de cerrar la estructura de la página (al final del contenedor principal)	Conexion 'conexion' HistorialNavegacion 'historial' boolean 'pagina_simple'

En grafico

@ubicacion	Ubicación en la página	Parámetros (@tipo='PHP')
despues_inicializacion	Después de inicializar las variables de la página; justo antes de comenzar a emitir a la salida	Conexion 'conexion' HistorialNavegacion 'historial' Flash 'flash' string &'titulo'
despues_inicializacion_grafico	Después de inicializar las variables del gráfico	Conexion 'conexion' HistorialNavegacion 'historial' array &'valores' array &'descripciones' array &'colores' array &'etiquetas' array &'links' string &'leyenda' float &'maximoy' int &'step' int &'orientacion_leyendas'
antes_html	Antes de comenzar a generar salida html (antes de <html>, después de enviar los headers HTTP)	Conexion 'conexion' HistorialNavegacion 'historial'
despues_html	Después de terminar de generar salida html (después de </html>)	Conexion 'conexion' HistorialNavegacion 'historial'
en_head	Antes de generar el elemento head. Lo que se emite a la salida se incluye dentro del head.	-

`@ubicacion`	Ubicación en la página	Parámetros (`@tipo='PHP'`)
`inicio_body`	Justo después de abrir el elemento `body`	`Conexion 'conexion' HistorialNavegacion 'historial' boolean 'pagina_simple'`
`fin_body`	Justo antes de cerrar el elemento `body`	`Conexion 'conexion' boolean 'pagina_simple'`
`despues_abrir_estructura`	Justo después de abrir la estructura de la página (al comienzo del contenedor principal)	`Conexion 'conexion' HistorialNavegacion 'historial' Flash 'flash' boolean 'pagina_simple'`
`antes_cerrar_estructura`	Justo antes de cerrar la estructura de la página (al final del contenedor principal)	`Conexion 'conexion' HistorialNavegacion 'historial' boolean 'pagina_simple'`

En `menu`

En el caso de `menu`, sólo se aplican las ubicaciones que están disponibles en todos los tipos de página. Además, el tipo de código está restringido a `@tipo="PHP"`.

Capítulo 7

Tipos de validación en campos de formulario

A continuación se listan los valores válidos para `/formulario/campos//campo/validacion`.

Nombre	Descripción	Expresión regular	Ejemplo válido	Ejemplo no válido
alpha	String de letras mayúsculas y minúsculas, puntos y guiones	`[a-zA-Z\.\-]`	ab.C-de	a.b c_de
alphanum	String de letras mayúsculas, minúsculas, números y guiones bajos	`[A-Za-z0-9_]`	ab_48_C_De	ab-c34 de
cp4	Código Postal tradicional, de 4 dígitos	`\d{4}`	7000	70000
cp8	Código Postal Argentino, de 8 caracteres	`[A-Za-z]\ d{4}[A-Za-z]{3}`	B7000HGA	B70000HGAX
cp8_manzana	Sección que identifica la manzana en el Código Postal Argentino, de 3 letras	`[A-Za-z]{3}`	HGA	HA
cuit	Número de CUIT válido, con formato “00000000000” o “00-00000000-0”			
cuit_formato	Número de CUIT, con formato “00000000000” o “00-00000000-0”, sin importar si es válido o no			
date	Fecha con año de 4 dígitos, con formato “D/M/AAAA”	`\d{1,2}\/\ d{1,2}\/\d{4}`	19/02/09	19/2/09 19-02-2009
datetime_cseg	Fecha con año de 4 dígitos y hora con segundos, en formato “D/M/AAAA HH:MM:SS”	`\d{1,2}\/\ d{1,2}\/\d{4} \d{2}\:\d{2}\: \d{2}`	19/02/09 15:35	19-2-2009 15.35.7 19/2/2009 15:35:7

Nombre	Descripción	Expresión regular	Ejemplo válido	Ejemplo no válido
datetime_sseg	Fecha con año de 4 dígitos y hora sin segundos, en formato "D/M/AAAA HH:MM"	`\d{1,2}\/\d{1,2}\/\d{4} \d{2}\:\d{2}`	19/02/09 15:05	19/2/2009 15:05:07 19/2/2009 15:5
email	Dirección de correo electrónico	`[\w-\.]+\@[\w\.-]+\.[a-z]{2,4}`	user@servidor.dom	user@servidor
hora	Hora sin segundos con formato "H:M"	`\d{1,2}:\d{1,2}`	15:05 15:5	15:5:7 15.5
horaconceros	Hora sin segundos con formato "HH:MM" (dos dígitos requeridos)	`\d{2}:\d{2}`	15:05	15:5 15.05
ip4	Dirección IP v4 sin máscara	`(\d{1,3})\.(\d{1,3})\.(\d{1,3})\.(\d{1,3})`; cada grupo debe tener valor entre 0 y 255	12.003.9.254	1234.abc.222.256 1,11 111.0
integer	Número entero precedido opcionalmente por un signo	`[\+\-]?\d+`	-522	+52 e2
onlyAlpha	String de letras mayúsculas y minúsculas	`[a-zA-Z]`	abCde	ab2de
phone	Número de teléfono: dígitos separados por puntos, guiones o blancos	`[\d\.\s\-]+`	0249 50 1234	A(022939) 50 1234
porcentaje	Número real con valor entre 0 y 100		45.3	145.3
real	Número real con signo opcional y coma como separador decimal	`[\+\-]?\d*(\,\d+)?`	-23,4	-23.4h -23,
realPos	Número real sin signo, con coma como separador decimal	`\d*(\,\d+)?`	23,4	-23.4h 23,
realGeneral	Número real con signo opcional y punto como separador decimal	`[\+\-]?\d*(\.\d+)?`	-23.4	-23,4h
realGeneralPos	Número real sin signo, con punto como separador decimal	`\d*(\.\d+)?`	23.4	-23,4h 23.
time	Hora con formato "H:M:S" (no se requieren 2 dígitos)	`\d{1,2}\:\d{1,2}\:\d{1,2}`	15:35:07	15:35 15.35.7
unsigned	Número entero sin signo	`\d+`	3984	-45c23

Nombre	Descripción	Expresión regular	Ejemplo válido	Ejemplo no válido
unsignedMayor0	Número entero sin signo, mayor que cero	`\d+`	3984	-45c23 0
year	Año de 4 dígitos	`\d{4}`	2009	9

Capítulo 8

Personalización de la seguridad

Esta sección describe la manera de personalizar los objetos de Seguridad empleados por XGAP, para adecuarlos a las necesidades específicas de una aplicación. Para ello se explica primero el funcionamiento de las clases e interfaces que definen los mecanismos de seguridad predeterminados y a continuación se brinda un ejemplo para ilustrar su extensión.

Esquema de seguridad

Las interfaces que definen el esquema de seguridad se encuentran dentro del motor, en el archivo `interfaces/iSeguridad.inc.php` y son:

- `iSeguridadConexion`
- `iSeguridad`

Las clases que provean la seguridad de una aplicación deben implementar estas interfaces.

La implementación provista por XGAP se encuentra en el archivo `clases/seguridad.inc.php` y está compuesta por las clases:

- `SeguridadConexionBase`, implementa `iSeguridadConexion`
- `SeguridadConexion`, hereda de `SeguridadConexionBase`
- `Seguridad`, implementa `iSeguridad`

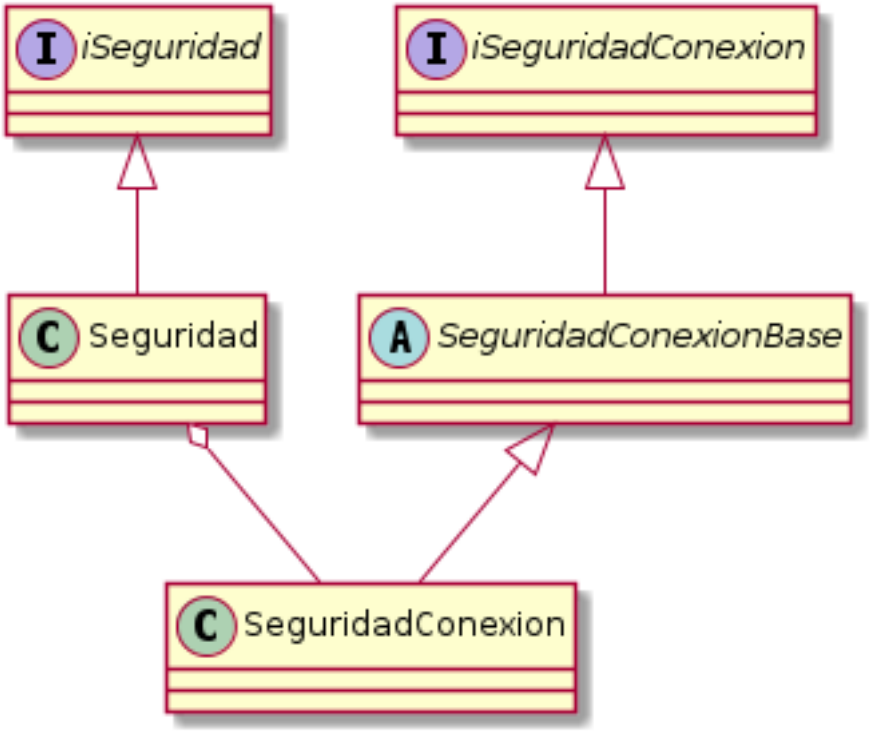


Figura 8.1: Estructura de clases de seguridad

La clase `Seguridad` contiene la lógica necesaria para poder determinar:

- Si un usuario puede o no ver una página determinada
- Obtener las páginas permitidas para un determinado usuario en un determinado rol
- Obtener los roles disponibles para un determinado usuario
- Actualizar los permisos de seguridad para un usuario determinado en una aplicación determinada.

El objetivo de la clase `SeguridadConexion` es servir de nexo entre la clase `Seguridad` y la clase de conexión, de manera que la clase `Seguridad` sea independiente de la base de datos.

Definición de seguridad personalizada

Para personalizar la implementación de seguridad en una aplicación se debe informar a XGAP qué clases alternativas se van a usar. Para ello existen tres parámetros de configuración:

`clase_seguridad`
: Indica la implementación de la interfaz `iSeguridad` que debe emplearse. Si este parámetro no es proporcionado, se emplea la clase predeterminada.

`clase_seguridad_conexion`
: Indica la implementación de la interfaz `iSeguridadConexion` que debe emplearse. Si este parámetro no es proporcionado, se emplea la clase predeterminada.

`archivo_seguridad`
: Especifica el archivo que contiene las clases de seguridad a emplear, definidas por los parámetros `clase_seguridad` y `clase_seguridad_conexion`. Ambas clases deben definirse en este archivo.

nota
Si se define el parámetro `archivo_seguridad` pero no `clase_seguridad` o `clase_seguridad_conexion`, la clase correspondiente es ignorada por XGAP a pesar de estar definida en el archivo indicado.

Ahora nos concentraremos en lo que podemos definir dentro de este archivo.

La manera más simple de personalizar la seguridad consiste en definir nuevas clases que hereden de las predeterminadas; de este modo sólo será necesario reimplementar los métodos que proveen la funcionalidad que se desea modificar. Por otro lado, nada impide que las aplicaciones implementen clases de seguridad completamente nuevas e independientes de las predeterminadas; el único requisito es que implementen las interfaces mencionadas.

El ejemplo que se da a continuación muestra el contenido mínimo que debe tener un archivo de personalización con clases de seguridad que heredan de las predeterminadas.

Ejemplo 8.1 Archivo mínimo para personalización de seguridad

```
<?php
class SeguridadConexionApp extends SeguridadConexion {
    /**
     * @param conexión $conexion
     */
    public function __construct($conexion) {
        parent::__construct($conexion);
    }
}

class SeguridadApp extends Seguridad {
    public function __construct($seguridad_conexion) {
        parent::__construct($seguridad_conexion);
    }
}
```

En este archivo se definen las clases `SeguridadApp` y `SeguridadConexionApp` como subclases de `Seguridad` y `SeguridadConexion`.

Para que XGAP utilice estas clases, es necesario cambiar los parámetros de configuración de la aplicación. Si el archivo de configuración tiene el nombre `seguridadapp.inc.php`, entonces habrá que definir las siguientes constantes dentro del archivo `conf.inc.php`:

```
<?php
define('XGAP_CONF_ARCHIVO_SEGURIDAD', 'seguridadapp.inc.php');
define('XGAP_CONF_CLASE_SEGURIDAD', 'SeguridadApp');
define('XGAP_CONF_CLASE_SEGURIDAD_CONEXION', 'SeguridadConexionApp');
```

El ejemplo anterior simplemente define nuevas clases, pero no modifica la implementación de la seguridad. A continuación se provee un ejemplo más completo.

Ejemplo 8.2 Personalización de seguridad con métodos reimplementados

```
<?php
class SeguridadConexionApp extends SeguridadConexionBase {
    /**
      * @param conexión
      */
    public function __construct($conexion) {
        parent::__construct($conexion);
    }

    /**
      * Retorna una consulta que agrega seguridad a una consulta dada.
```

```
      * @param string $consulta
      * @return mixed string
      */
    public function obtenerConsultaSegura($consulta) {
        // Hacer cambios a $consulta aquí
        return $consulta;
    }

    /**
      * Retorna una consulta SQL para comprobar la seguridad de una página.
      * @param string $pagina
      * @return mixed boolean | lista de permisos
      */
    public function verificarSeguridadPagina($pagina) {
        // .....
    }

    public function borrarPermisosDePagina($aplicacion, $rol_usuario,
            $tipo_pagina) {
        // .....
    }

    public function agregarPaginaSiNoExiste($aplicacion, $pagina) {
        // .....
    }

    public function agregarPermisoAPagina($aplicacion, $rol_usuario, $pagina,
            $operaciones) {
        // .....
    }

    public function obtenerPaginasPermitidas($aplicacion, $rol_usuario,
            $tipo_pagina) {
        // .....
    }

    public function obtenerRolesDisponibles() {
        // .....
    }
}

class SeguridadApp extends Seguridad {
    public function __construct($seguridad_conexion) {
        parent::__construct($seguridad_conexion);
    }

    public function verificarSeguridad($error) {
        // .....
    }
}
```

sugerencia
La implementación provista por XGAP puede observarse en el archivo `clases/seguridad.inc.php` del motor.

Parte IV

Ejemplos

Capítulo 9

Ejemplo: Aplicación “Localidades”

Introducción

La aplicación “Localidades” se provee como un ejemplo simple, cuyo objetivo es demostrar la forma es que se usan algunas de las características principales de XGAP, además de presentar una posible estructura para una aplicación XGAP.

Su funcionalidad principal permite registrar divisiones geopolíticas: localidades, provincias, países. Además permite asociar múltiples fotos a las localidades.

El modelo de datos está representado en el siguiente diagrama:

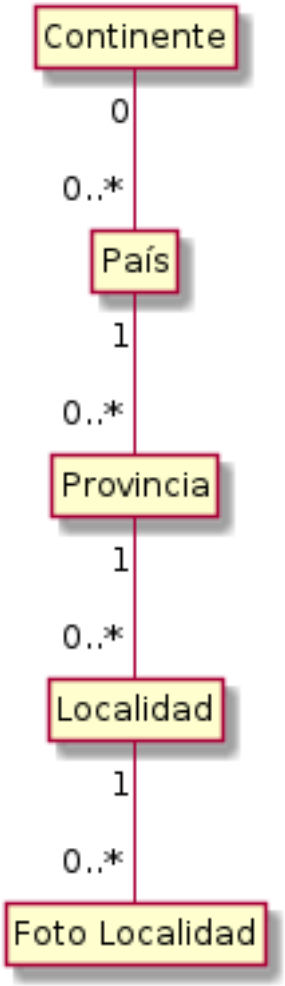


Figura 9.1: Modelo de datos de la aplicación “Localidades”

Cada una de las entidades “País”, “Provincia”, “Localidad” y “Foto Localidad” cuentan con su correspondiente listado y formulario para consultar y operar con sus instancias, en tanto que la entidad “Continente” tiene sus valores predefinidos en la base de datos y no se asocia a páginas de la aplicación.

---

nota
El modelo de datos está diseñado para servir como base para demostrar características de XGAP, no para ofrecer una aplicación que tenga una utilidad real.

Estructura de directorios del proyecto

Parte de la estructura de directorios de una aplicación está predefinida por XGAP; específicamente:

- Un directorio raíz para los fuentes de la aplicación. XGAP espera que este directorio se encuentre dentro del directorio `APPS_DIR` y tenga el mismo nombre que la aplicación, para que el generador lo detecte, pero también es posible ubicarlo en cualquier otro lugar, con cualquier otro nombre, y crear un enlace en `APPS_DIR`.
- Subdirectorios `extras`, `extras/menu` y `resultados` dentro de este directorio raíz.

Fuera de esta estructura predefinida, cada proyecto se puede organizar como resulte más conveniente.

Para el caso de esta aplicación, se optó por una estructura autocontenida: tanto la raíz de la aplicación como otros directorios adicionales se encuentran dentro de un único directorio que corresponde al proyecto entero. Este directorio se puede ubicar en cualquier parte del sistema de archivos, independientemente de dónde se encuentre la distribución de XGAP. Para poder generar la aplicación, se debe colocar un enlace a la raíz de la aplicación dentro de `APPS_DIR`, con nombre `localidades`.

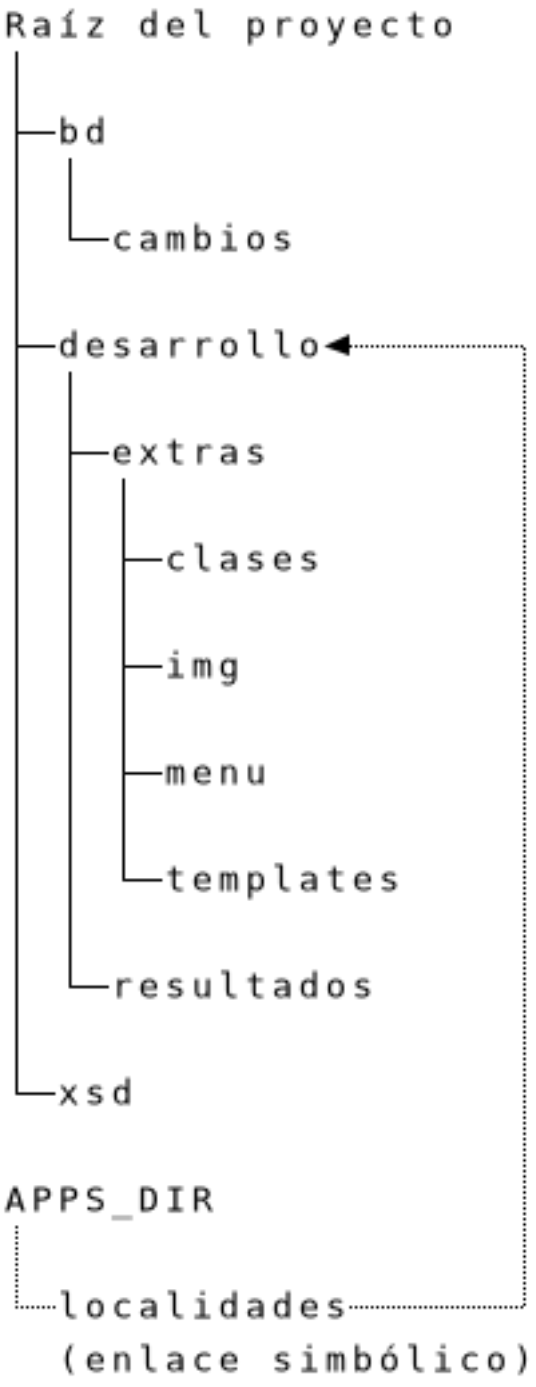


Figura 9.2: Árbol de directorios de la aplicación “Localidades”

La raíz para XGAP es el subdirectorio `desarrollo`, mientras que los demás subdirectorios del mismo nivel contienen archivos que no son usados directamente por XGAP. El subdirectorio `desarrollo/extras` contiene subdirectorios adicionales con recursos que va a usar en tiempo de ejecución la aplicación generada.

bd
Scripts SQL para crear la base de datos inicial completa.

bd/cambios
Scripts SQL incrementales, para actualizar la base de datos desde una versión anterior.

desarrollo
Raíz para XGAP, conteniendo los fuentes XML.

desarrollo/extras
Requerido por XGAP. Fuentes adicionales y otros recursos que se copian directamente a la aplicación generada.

desarrollo/extras/clases
Clases PHP.

desarrollo/extras/img
Imágenes.

desarrollo/extras/menu
Definiciones de menúes.

desarrollo/extras/templates
Plantillas para usar en reportes.

desarrollo/resultados
Requerido por XGAP.

xsd
Contiene los esquemas XML (`*.xsd`) del motor en uso. Este directorio no forma parte de la distribución de la aplicación (no se guarda en el repositorio de código) pero está referenciado por el atributo `xsi:noNamespaceSchemaLocation` del elemento raíz de los fuentes XML.

Una forma práctica de completar este directorio es crear enlaces simbólicos hacia los archivos `*.xsd` del directorio `plantillas` del motor.

Ejemplo 9.1 Definición de enlaces simbólicos para la aplicación “Localidades”, en Linux

En Linux se pueden usar los comandos dados a continuación para crear los enlaces simbólicos necesarios para poder trabajar con la aplicación descargada.
Para este ejemplo, se asume que:

- La aplicación se encuentra en `/home/user/Projects/xgap/examples/localidades`.
- `APPS_DIR` está definido como `/home/user/Projects/xgap/apps`.
- El motor usado se encuentra en `/home/user/Projects/xgap/dist/motores/ultimo`.
- `{APPS_DIR}/localidades` no existe previamente.

```
ln -s /home/user/Projects/xgap/examples/localidades/desarrollo \
  /home/user/Projects/xgap/apps/localidades
for file in /home/user/Projects/xgap/dist/motores/ultimo/plantillas/*.xsd; do
  ln -s "$file" /home/user/Projects/xgap/examples/localidades/xsd/
done
```

Esquema de la base de datos

El esquema de base de datos que usa la aplicación (creado por los scripts que se encuentran en el subdirectorio `bd`) está representado en la figura [1] siguiente.

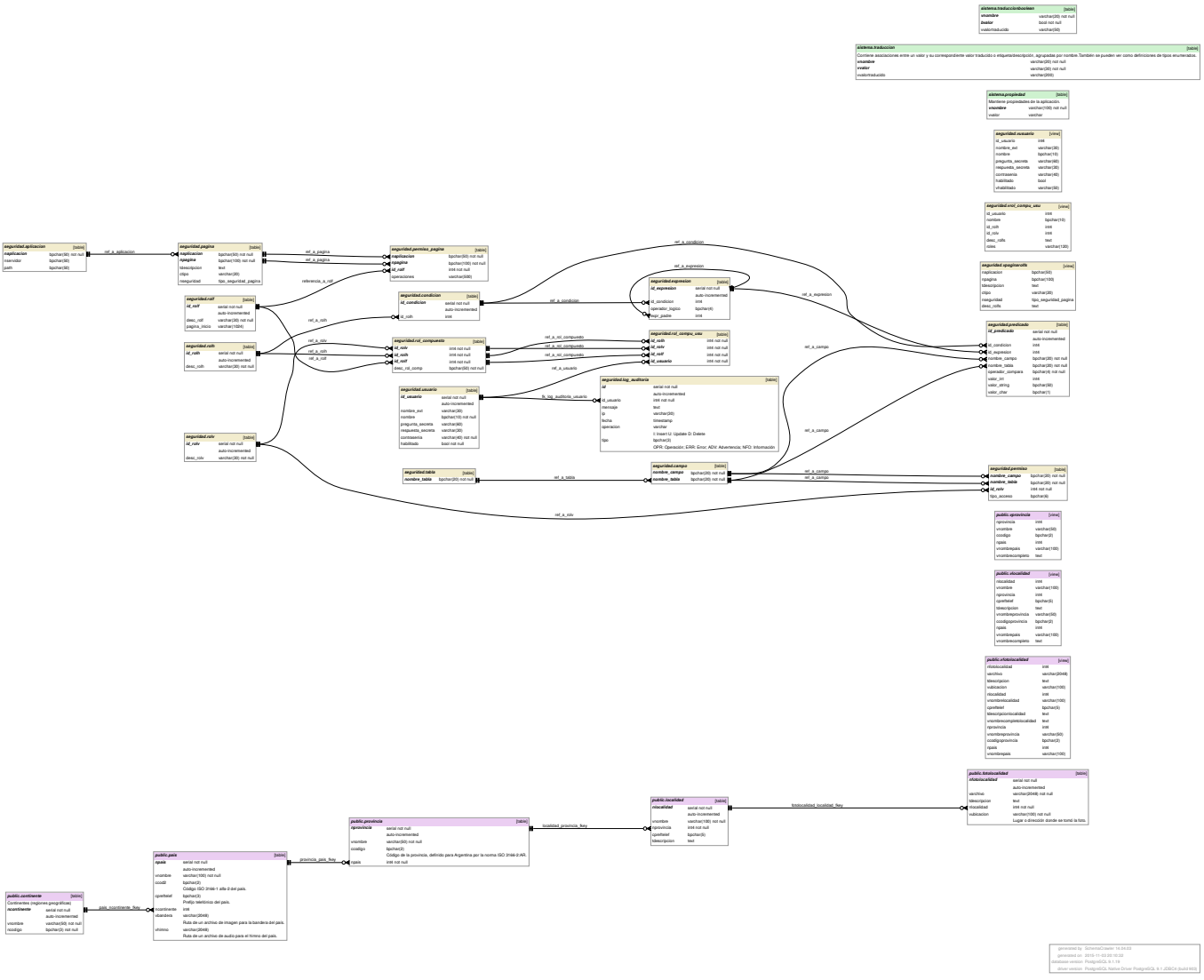


Figura 9.3: Diagrama del esquema de base de datos de la aplicación “Localidades”

Además de las tablas y vistas predefinidas por XGAP, el esquema contiene las correspondientes al modelo de datos y algunas auxiliares:

- Modelo
  - Tabla `public.continente`
  - Tabla `public.pais`
  - Tabla `public.provincia`
  - Tabla `public.localidad`
  - Tabla `public.fotolocalidad`

[1] Diagrama generado con SchemaCrawler

- Vista `public.vprovincia`
- Vista `public.vlocalidad`
- Vista `public.vfotolocalidad`

- Auxiliares
  - Tabla `sistema.traduccion`
  - Tabla `sistema.traduccionboolean`
  - Tabla `sistema.propiedad`

Componentes

A continuación se listan los principales fuentes XML, y las tablas y vistas más relevantes que usa cada uno de ellos.

Cuadro 9.1: Relación entre los fuentes XML y las tablas/vistas en la base de datos de la aplicación "Localidades"

Fuente	**Usa**
`fotolocalidad_contenido.xml`	`public.vfotolocalidad`
`fotolocalidad_formulario.xml`	`public.fotolocalidad`
`fotolocalidad_listado.xml`	`public.fotolocalidad`
`fotolocalidad_reporte_odt.xml`	`public.vlocalidad, public.vfotolocalidad`
`index_admin_contenido.xml`	`public.pais, public.provincia, public.localidad, public.fotolocalidad`
`index_contenido.xml`	`--`
`login_contenido.xml`	`seguridad.usuario, seguridad.rol_compu_usu, seguridad.rolf`
`localidad_formulario.xml`	`public.localidad`
`localidad_listado.xml`	`public.vlocalidad`
`localidad_master.xml`	`public.vlocalidad`
`paginaapp_formulario.xml`	`seguridad.pagina`
`paginaapp_listado.xml`	`seguridad.vpaginarolfs`
`pais_formulario.xml`	`public.pais, public.continente`
`pais_listado.xml`	`public.pais`
`permiso_pagina_export_listado.xml`	`seguridad.permiso_pagina`
`provincia_formulario.xml`	`public.provincia`
`provincia_listado.xml`	`public.vprovincia`
`rol_compu_usu_formulario.xml`	`seguridad.vrol_compu_usu`
`rol_compu_usu_listado.xml`	`seguridad.vrol_compu_usu`
`rolf_formulario.xml`	`seguridad.rolf`
`rolf_listado.xml`	`seguridad.rolf`
`seleccionarrol_contenido.xml`	`seguridad.rol_compuesto, seguridad.rol_compu_usu, seguridad.rolf`
`usuario_cambioclave_formulario.xml`	`seguridad.usuario`
`usuario_cambioclaveadmin_formulario.xml`	`seguridad.usuario`
`usuario_formulario.xml`	`seguridad.usuario`
`usuario_listado.xml`	`seguridad.vusuario`
`usuario_sinclave_formulario.xml`	`seguridad.usuario`

El diagrama siguiente presenta las páginas (archivos generados a partir de los fuentes XML) principales que componen la apli-

cación y los caminos que se pueden seguir para navegar a cada una de ellas. Para simplificar el diagrama, no se incluyen las versiones de los listados en formatos no HTML (`configuracion/imprimibles/imprimir`).

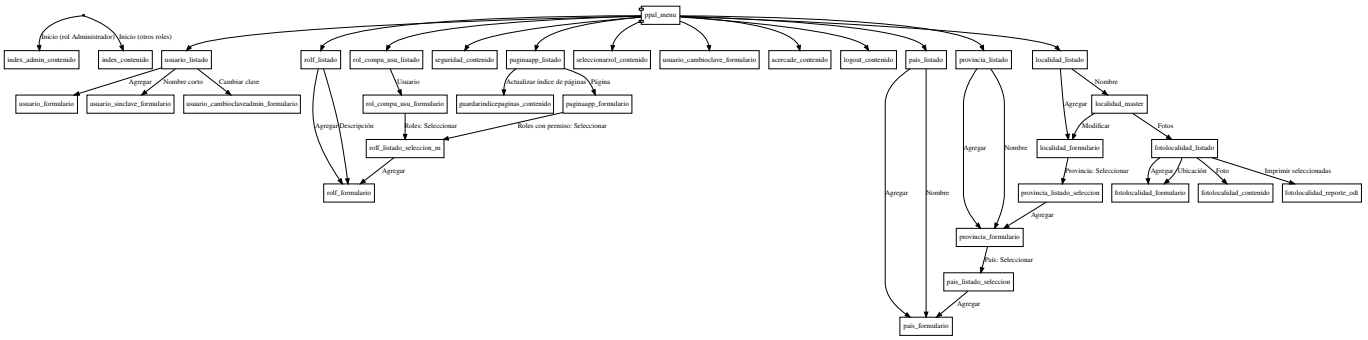


Figura 9.4: Páginas de la aplicación “Localidades” y navegación entre ellas

Contenido común a todas las páginas

Las opciones de configuración de la aplicación `norte`, `sur`, `este` y `oeste` permiten especificar archivos PHP que se incluyen en todas las páginas, dentro de la sección de la página que indica el nombre de cada opción. En el caso de esta aplicación, sólo se usan las secciones `norte` y `sur`. La configuración, en `conf.inc.php`, es:

```
<?php
define('XGAP_CONF_NORTE', 'norte.php');
define('XGAP_CONF_SUR', 'sur.php');
define('XGAP_CONF_ESTE', 'ignorar');
define('XGAP_CONF_OESTE', 'ignorar');
```

Los archivos `norte.php` y `sur.php` se encuentran dentro del directorio `desarrollo/extras`. La salida que producen se emite en la parte superior e inferior, respectivamente, de todas las páginas.

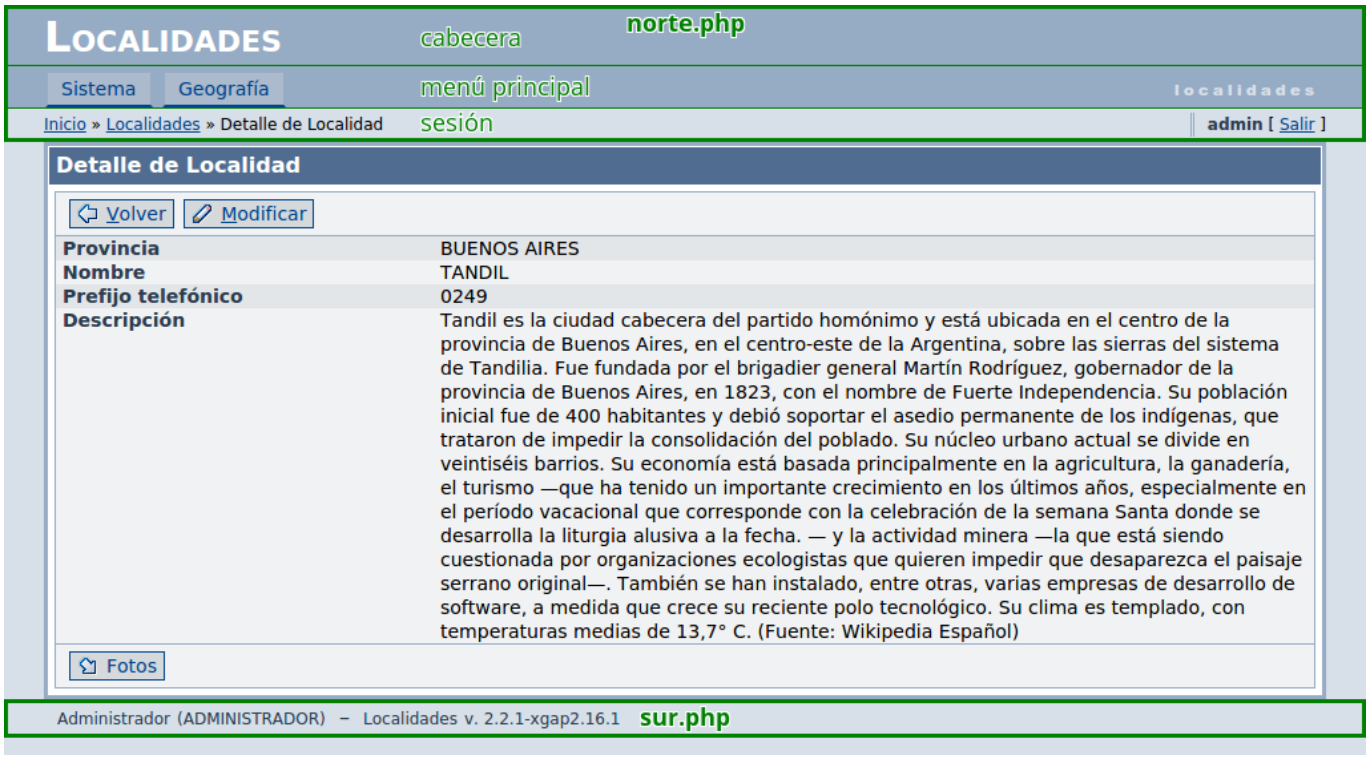


Figura 9.5: Secciones `norte` y `sur` en una página de la aplicación “Localidades”

En `norte.php` se definen tres áreas:

- La cabecera, que contiene el título de la aplicación envuelto en un enlace que lleva a la página de inicio.

```
<?php
Html::link(Configuracion::paginaInicio() . '?' . PARAMETRO_HISTORIAL . '=reset',
    XGAP_APP_TITULO, null, null, false, null, null, 'Inicio');
```

- El menú de la aplicación.

```
<?php
Html::abrirTag('div', 'menu-ppal', null, null, false, null, false, null, false); // ❶
Html::cerrarTag('div');
// ...
$menu_params = array(PARAMETRO_HISTORIAL => 'skip');
$param_recarga_menu = Request::obtener(RequestXgap::PARAMETRO_RECARGA_MENU, null);
if (param2boolean($param_recarga_menu)) {
    $menu_params[RequestXgap::PARAMETRO_RECARGA_MENU] = $param_recarga_menu;
}
$menu = crearDireccionPagina('ppal_menu.php', $menu_params, false);              // ❷
$gfxpath = XGAP_CONF_PREFIJO_WEB . '/recursos/imagenes/menu/';
Html::abrirJavascript();
echo <<<MENU
if (typeof dhtmlXMenuBarObject != 'undefined') {
    aMenuBar = new dhtmlXMenuBarObject(
        document.getElementById("menu-ppal"),                                    // ❸
        "100%", 23, ""
    );
    aMenuBar.setGfxPath("$gfxpath");
    aMenuBar.loadXML("$menu");
    aMenuBar.showBar();
```

```
}
MENU;
Html::cerrarJavascript();
unset($param_recarga_menu, $menu_params, $menu, $gfxpath);
```

1. Emite el elemento vacío dentro del cual se va a construir el menú.
2. El archivo `ppal_menu.php` produce la estructura del menú. Ver sección Menú de la aplicación.
3. Se crea el menú dentro del elemento creado para ello.

- Una barra de sesión, con la ruta de navegación (*breadcrumb*), el nombre del usuario logueado y el link para cerrar la sesión.

```
<?php
echo Componentes::barraHistorial(
    $GLOBALS['objeto_historial'],
    'breadcrumb',
    ' &raquo; ',
    XGAP_CONF_ITEMS_BARRA_HISTORIAL
);
// ...
if (Contexto::existe('usuario_sistema')) {
    // ...
    Html::elemento(
        'span',
        Contexto::obtener('usuario_sistema'),
        null,
        'sesion-usuario',
        null
    );
    Html::elemento(
        'span',
        '[ '
                . Html::link(
                    'logout_contenido.php?' . PARAMETRO_HISTORIAL . '=reset',
                    'Salir', null, null, true)
                . ' ]',
        null,
        'sesion-cerrar',
        null
    );
    // ...
}
```

En `sur.php` se muestra:

- El nombre y rol del usuario logueado

```
<?php
$usuario = Contexto::existe('nombre_usuario') // 1
        ? Contexto::obtener('nombre_usuario')
        : Contexto::obtener('usuario_sistema', null); // 2
if (!empty($usuario)) {
    Html::elemento(
        'span',
        Formateo::prepararSalidaString($usuario, null, null, true, true),
        null,
        'sesion-usuario',
        null
    );
    if (Contexto::existe('rol')) { // 3
        Html::elemento(
```

```
            'span',
            ' ('
                    . Formateo::prepararSalidaString(
                            Contexto::obtener('rol'),
                            null, null, true, true)
                    . ')',
            null,
            'sesion-rol',
            null
        );
    }
    echo '  &ndash;  ';
}
```

❶, ❷, ❸ Las variables `'nombre_usuario'`, `'usuario_sistema'` y `'rol'` se almacenan en la sesión después del login y selección de rol.

- El nombre y versión de la aplicación

```
<?php
Html::elemento('span',
        XGAP_APP_TITULO, // ❶
        null, 'app-titulo', null);
Html::elemento('span',
        ' v. ' . XGAP_CONF_VERSION_APLICACION, // ❷
        null, 'app-version', null);
```

❶, ❷ `XGAP_APP_TITULO` y `XGAP_CONF_VERSION_APLICACION` son dos constantes predefinidas por XGAP.

Menú de la aplicación

El menú de la aplicación está definido en el archivo `desarrollo/extras/menu/ppal_menu.xml`. Este archivo se procesa durante la generación de la aplicación, produciendo el archivo `ppal_menu.php`, que se carga en `norte.php` (ver sección Contenido común a todas las páginas) para construir el menú cuando se cargan las páginas.

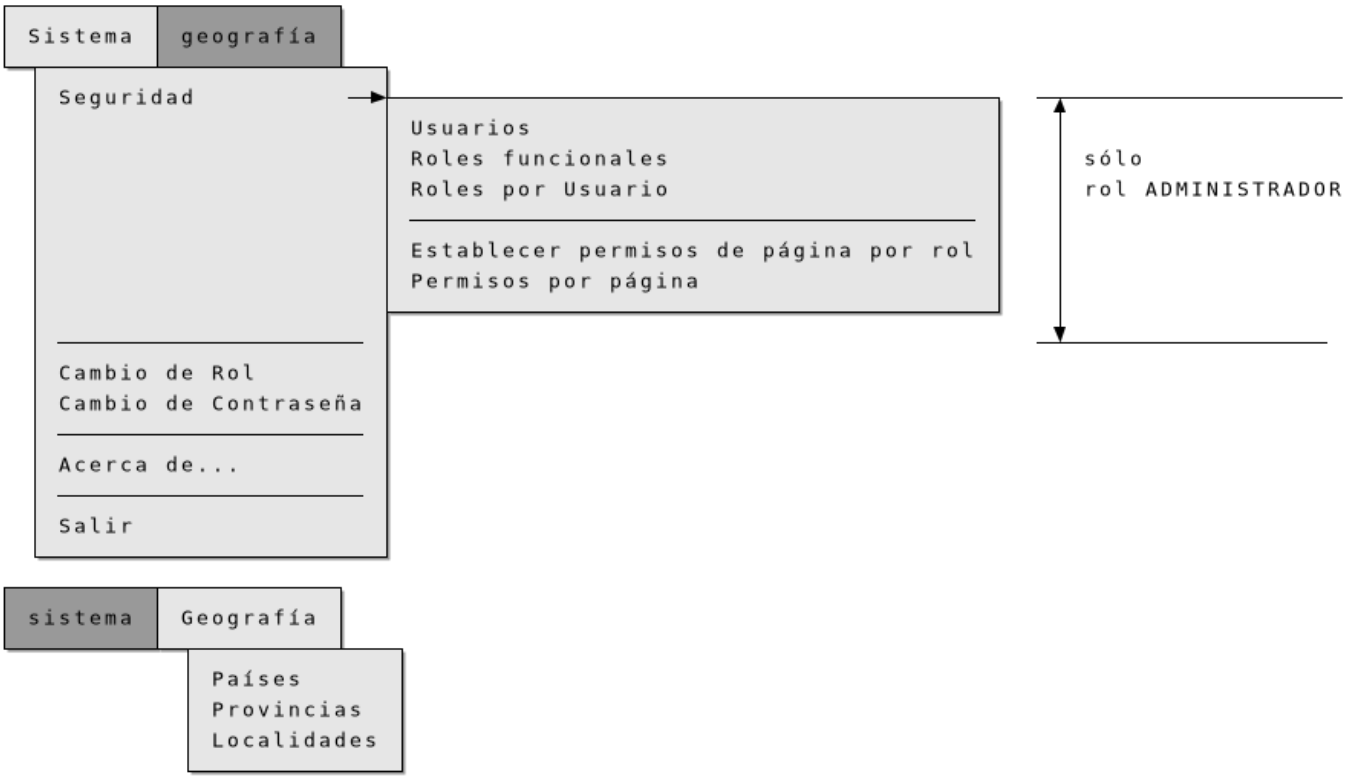


Figura 9.6: Elementos del menú principal de la aplicación “Localidades”

```
<menu xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="../../../xsd/menu.xsd"
      aplicacion="localidades"
      cache="rol">                                                          ❶
    <carpeta nombre="Sistema">
        <carpeta nombre="Seguridad" roles="ADMINISTRADOR">                  ❷
            <item nombre="Usuarios" url="usuario_listado.php?xgap_historial=reset"/>
            <item nombre="Roles funcionales" url="rolf_listado.php?xgap_historial=reset"/>
            <item nombre="Roles por Usuario" url="rol_compu_usu_listado.php?xgap_historial= ↩
                reset"/>
            <separador/>
            <item nombre="Establecer permisos de página por rol" url="seguridad_contenido. ↩
                php?xgap_historial=reset"/>
            <item nombre="Permisos por página" url="paginaapp_listado.php?xgap_historial= ↩
                reset"/>
        </carpeta>
        <separador roles="ADMINISTRADOR"/>                                  ❸
        <item nombre="Cambio de Rol" url="seleccionarrol_contenido.php?xgap_historial=reset ↩
            "/>
        <item nombre="Cambio de Contraseña" url="usuario_cambioclave_formulario.php? ↩
            xgap_historial=reset&amp;agregado=false"/>
        <separador/>
        <item nombre="Acerca de..." url="acercade_contenido.php?xgap_historial=reset"/>
        <separador/>
        <item nombre="Salir" url="logout_contenido.php?xgap_historial=reset"/>
    </carpeta>
    <carpeta nombre="Geografía">
        <item nombre="Países" url="pais_listado.php?xgap_historial=reset"/>
        <item nombre="Provincias" url="provincia_listado.php?xgap_historial=reset"/>
        <item nombre="Localidades" url="localidad_listado.php?xgap_historial=reset"/>
```

```
    </carpeta>
</menu>
```

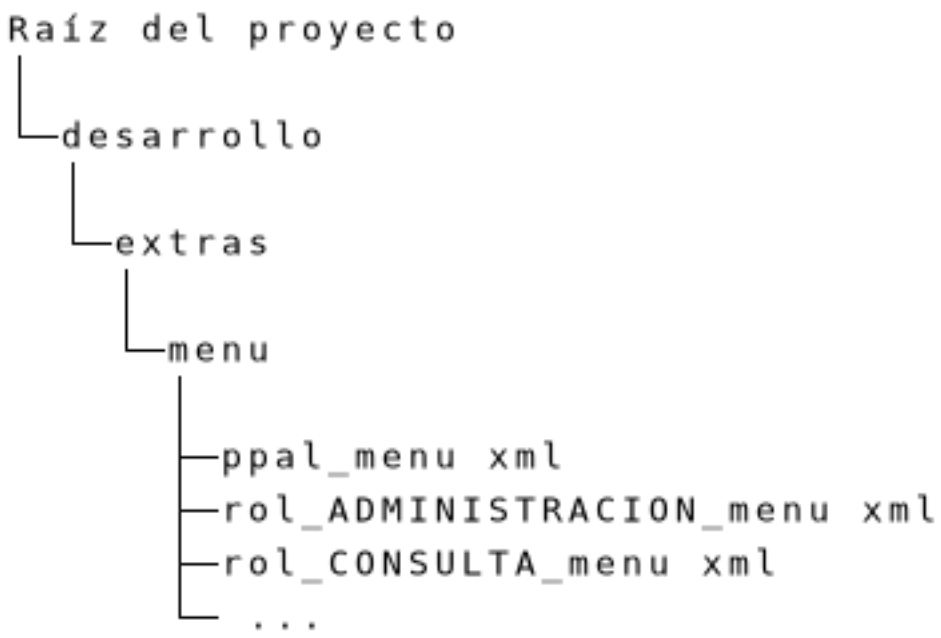
❶ El valor `"rol"` en el atributo `cache` (o la ausencia del atributo, dado que `"rol"` es su valor por defecto) hace que la estructura que define el menú (la salida emitida por `ppal_menu.php`) se guarde en la sesión del usuario, asociada al rol actual. Si el rol cambia, la estructura se reconstruye y se vuelve a guardar. Si se usara `cache="no"`, la estructura del menú se reconstruiría en cada solicitud a `ppal_menu.php`. Si se usara `cache="usuario"`, la estructura se guardaría en el cache asociada al usuario, no al rol, y se seguiría retornando la misma aún ante un cambio de rol por parte del usuario; como en este menú se usa el atributo `roles` en algunos de los elementos (explicado a continuación), este comportamiento sería incorrecto, porque evitaría que se determinen los elementos a mostrar para el nuevo rol seleccionado.

❷, ❸ El atributo `roles`, disponible en los elementos `carpeta`, `item` y `separador`, permite indicar una lista de roles para los cuales se debe mostrar el elemento. En este caso, la carpeta "Sistema" → "Seguridad" y el separador adyacente sólo se muestran para el rol "ADMINISTRADOR". Los elementos que no tienen el atributo se muestran para todos los roles.

En esta aplicación se utiliza un único menú para todos los usuarios y roles, pero también es posible definir más de un menú y cargar uno u otro de acuerdo a algún criterio relevante para la aplicación. Por ejemplo, se podría definir un menú diferente para cada rol y cargar el que corresponda al rol activo. El ejemplo siguiente muestra una posible forma de implementar este caso.

Ejemplo 9.2 Uso de menúes diferentes de acuerdo al rol del usuario

- Especificar un archivo de definición separado para cada rol.



- En el código donde se carga el menú, seleccionar el archivo php que corresponda al rol activo.

  En norte.inc.php:

```
<?php
// ...
$archivo_menu = 'rol_' . Contexto::obtener('rol') . '_menu.php'; // ❶
if (!file_exists($archivo_menu)) {
    $archivo_menu = 'ppal_menu.php'; // ❷
}
$menu_params = array(PARAMETRO_HISTORIAL => 'skip');
$param_recarga_menu = Request::obtener(RequestXgap::PARAMETRO_RECARGA_MENU, null);
if (param2boolean($param_recarga_menu)) {
    $menu_params[RequestXgap::PARAMETRO_RECARGA_MENU] = $param_recarga_menu;
```

```
}
$menu = crearDireccionPagina($archivo_menu, $menu_params, false); // ❸
// ...
```

❶ Se construye el nombre del archivo en base al rol del usuario.

❷ Si el archivo no existe (el rol actual no tiene un menú específico) se usa un menú genérico.

❸ Se construye la dirección de la página a cargar, usando el archivo seleccionado.

Páginas y documentos

En esta sección se describen las páginas y documentos más relevantes de la aplicación, destacando algunas de las características de XGAP que utilizan, el código relacionado y cómo se refleja en la interfaz a usuario, si corresponde.

Cuadro 9.2: Agrupación por tipo de las páginas y documentos de la aplicación “Localidades”

Tipo	Fuentes
Contenido	acercade_contenido.xml, error_contenido.xml, error_formulario_contenido.xml, fotolocalidad_contenido.xml, guardarindicepaginas_contenido.xml, index_admin_contenido.xml, index_contenido.xml, login_contenido.xml, seguridad_contenido.xml, seleccionarrol_contenido.xml
Formulario	fotolocalidad_formulario.xml, localidad_formulario.xml, paginaapp_formulario.xml, pais_formulario.xml, provincia_formulario.xml, rol_compu_usu_formulario.xml, rolf_formulario.xml, usuario_cambioclave_formulario.xml, usuario_cambioclaveadmin_formulario.xml, usuario_formulario.xml, usuario_sinclave_formulario.xml
Listado	fotolocalidad_listado.xml, localidad_listado.xml, paginaapp_listado.xml, pais_listado.xml, permiso_pagina_export_listado.xml, provincia_listado.xml, rol_compu_usu_listado.xml, rolf_listado.xml, usuario_listado.xml
Master	localidad_master.xml
Reporte ODT	fotolocalidad_reporte_odt.xml

Cuadro 9.3: Guía de características destacadas de XGAP usadas en la aplicación “Localidades”

Característica	**Usada en**
Código extra	acercade_contenido.xml (despues_inicializacion PHP), fotolocalidad_formulario.xml (antes_procesar_uploads PHP), fotolocalidad_listado.xml (despues_inicializacion PHP), paginaapp_listado.xml (xgap_cargado PHP, antes_consulta PHP, despues_inicializacion PHP, antes_tabla_datos PHP, fin_body JAVASCRIPT), pais_formulario.xml (antes_procesar_uploads PHP), permiso_pagina_export_listado.xml (antes_consulta PHP, despues_inicializacion PHP), rol_compu_usu_formulario.xml (inicio_pagina PHP), usuario_cambioclave_formulario.xml (despues_inicializacion PHP, fin_body JAVASCRIPT), usuario_cambioclaveadmin_formulario.xml (despues_inicializacion PHP, fin_body JAVASCRIPT), usuario_formulario.xml (xgap_cargado PHP, fin_body JAVASCRIPT), usuario_sinclave_formulario.xml (xgap_cargado PHP, despues_inicializacion PHP)
Comando “Volver” con destino variable	error_contenido.xml, error_formulario_contenido.xml
Comando “Volver” condicional	error_contenido.xml, error_formulario_contenido.xml
Contenido de página personalizado	acercade_contenido.xml, index_admin_contenido.xml, error_contenido.xml, error_formulario_contenido.xml, fotolocalidad_contenido.xml, guardarindicepaginas_contenido.xml, login_contenido.xml
Contenido extra en la cabecera HTML	acercade_contenido.xml, index_admin_contenido.xml, login_contenido.xml, fotolocalidad_contenido.xml
Especificación de variedades de página a generar	fotolocalidad_listado.xml, localidad_listado.xml, paginaapp_listado.xml, pais_listado.xml, permiso_pagina_export_listado.xml, provincia_listado.xml, rolf_listado.xml
Formato personalizado en columna de listado	paginaapp_listado.xml
Formulario con campo de tipo `ComboBD` o `RadioBD`	pais_formulario.xml, usuario_formulario.xml, usuario_sinclave_formulario.xml
Formulario con campo de tipo `SeleccionableMultiple`	paginaapp_formulario.xml, rol_compu_usu_formulario.xml
Formulario con operación personalizada	paginaapp_formulario.xml (custom_update), rol_compu_usu_formulario.xml (custom_insert y custom_update)
Listado con acciones que operan sobre filas seleccionadas	fotolocalidad_listado.xml
Listado con buscador personalizado	localidad_listado.xml, paginaapp_listado.xml, provincia_listado.xml
Listado con columnas condicionales	usuario_listado.xml
Listado con columnas de imágenes	fotolocalidad_listado.xml, pais_listado.xml
Listado con columnas de acciones	usuario_listado.xml
Mensajes de página	guardarindicepaginas_contenido.xml
Página con acciones generales	paginaapp_listado.xml, seguridad_contenido.xml
Uso de XInclude	usuario_cambioclave_formulario.xml, usuario_cambioclaveadmin_formulario.xml, usuario_sinclave_formulario.xml

acercade_contenido.xml

Página que muestra información acerca de la aplicación.

Figura 9.7: Captura de pantalla de acercade_contenido.php en la aplicación “Localidades”

Contenido extra en la cabecera HTML

Se agregan reglas CSS adicionales que definen estilos específicos para esta página.

```
<pagina>
    ....
    <head-extra>
        <![CDATA[
<style type="text/css">
....
</style>
        ]]>
    </head-extra>
```

---

sugerencia

Una desventaja de incluir el código CSS en el fuente XML es que los estilos especificados permanecen fijos por más que se cambie el skin en uso.

En lugar de hacerlo de esta manera, se puede cargar CSS adicional incluyendo uno o más archivos externos, a través del elemento `configuracion/inclusiones/archivo`. Si la ruta incluye la constante `XGAP_CONF_SKIN_DIR`, el archivo cargado va a depender del skin en uso. Por ejemplo:

```
    <configuracion>
        <inclusiones>
            <archivo tipo="css">
                <path>XGAP_CONF_SKIN_DIR . 'extras.css'</path>
            </archivo>
            ...
```

Contenido de la página

Hace uso de varias constantes predefinidas por XGAP, que contienen información de la aplicación, el motor y el entorno.

```
<?php
//...
        Html::elemento('h1', XGAP_APP_TITULO, 'acercade-app');
        Html::elemento('p', 'Versi&oacute;n: ' . XGAP_CONF_VERSION_APLICACION, ' ↩
            acercade-version');
        Html::elemento('p', 'Aplicaci&oacute;n de ejemplo de XGAP.', 'acercade-texto');
        if (!XGAP_CONF_EN_PRODUCCION) {
            Html::elemento('p', 'Generada con XGAP ' . XGAP_VERSION_MOTOR_GEN . ' en '  ↩
                . date('c', XGAP_TIMESTAMP_GEN), 'info-generacion');
            Html::elemento('p', 'Corriendo sobre XGAP ' . XGAP_VERSION_MOTOR, 'info- ↩
                ejecucion');
        }
```

Código extra PHP en `despues_inicializacion`

Incluye el título de la aplicación en el título de la página.

```
            <?php
            $params['titulo'] = 'Acerca de ' . XGAP_APP_TITULO;
```

error_contenido.xml

Figura 9.8: Captura de pantalla de error_contenido.php en la aplicación “Localidades”

Comando “Volver” condicional

El comando sólo se muestra cuando la página no es simple.

```
<pagina>
    <configuracion>
        <volver>
            <!-- ... -->
            <condicion>return !defined('XGAP_PAGINA_SIMPLE') || !XGAP_PAGINA_SIMPLE;</ ↩
                condicion>
```

Comando “Volver” con destino variable

El destino del comando “Volver” en esta página varía de acuerdo al estado del historial. Se usa `volver/codigo-destino` para seleccionar la página de retorno correcta.

```
<pagina>
    <configuracion>
        <volver>
            <codigo-destino>
                <![CDATA[
                global $objeto_historial;
                if (!$objeto_historial->vacio() &&
                        strpos(
                            $objeto_historial->actual()->uri(),
                            basename($_SERVER['REQUEST_URI'])
                        ) !== false) {
                    $objeto_historial->remover();
                }
                if (Request::existe('origen') &&
                        !$objeto_historial->vacio() &&
                        strpos(
                            $objeto_historial->actual()->uri(),
                            Request::obtener('origen')
                        ) !== false) {
                    $objeto_historial->remover();
                }
                $retorno = !$objeto_historial->vacio()
                    ? $objeto_historial->actual()->uri()
                    : Configuracion::paginaInicio();
                $retorno = 'location="' . htmlspecialchars($retorno) . '"';
                return $retorno;
                ]]>
            </codigo-destino>
```

error_formulario_contenido.xml

Figura 9.9: Captura de pantalla de error_formulario_contenido.php en la aplicación “Localidades”

Comando “Volver” condicional
Ver error_contenido.xml.

Comando “Volver” con destino variable
El destino del comando “Volver” en esta página varía de acuerdo al estado del historial. Se usa `volver/codigo-destino` para seleccionar la página de retorno correcta.

En `error_formulario_contenido.xml`:

```
<pagina>
    <configuracion>
        <volver>
            <codigo-destino>
                <![CDATA[
                global $retorno;
                return 'location="' . htmlspecialchars($retorno) . '"';
                ]]>
            </codigo-destino>
```

En `error_formulario_contenido-anterior.inc.php`:

```
<?php
//...
$cur = basename($_SERVER['REQUEST_URI']);
if (!$objeto_historial->vacio() &&
        strpos($objeto_historial->actual()->uri(), $cur) !== false)
```

```
    $objeto_historial->remover();
if (!$objeto_historial->vacio() &&
        strpos($objeto_historial->actual()->uri(), $cur) !== false)
    $objeto_historial->remover();
$retorno = !$objeto_historial->vacio()
    ? $objeto_historial->primero(array($cur))->uri()
    : Configuracion::paginaInicio();
//...
```

fotolocalidad_contenido.xml

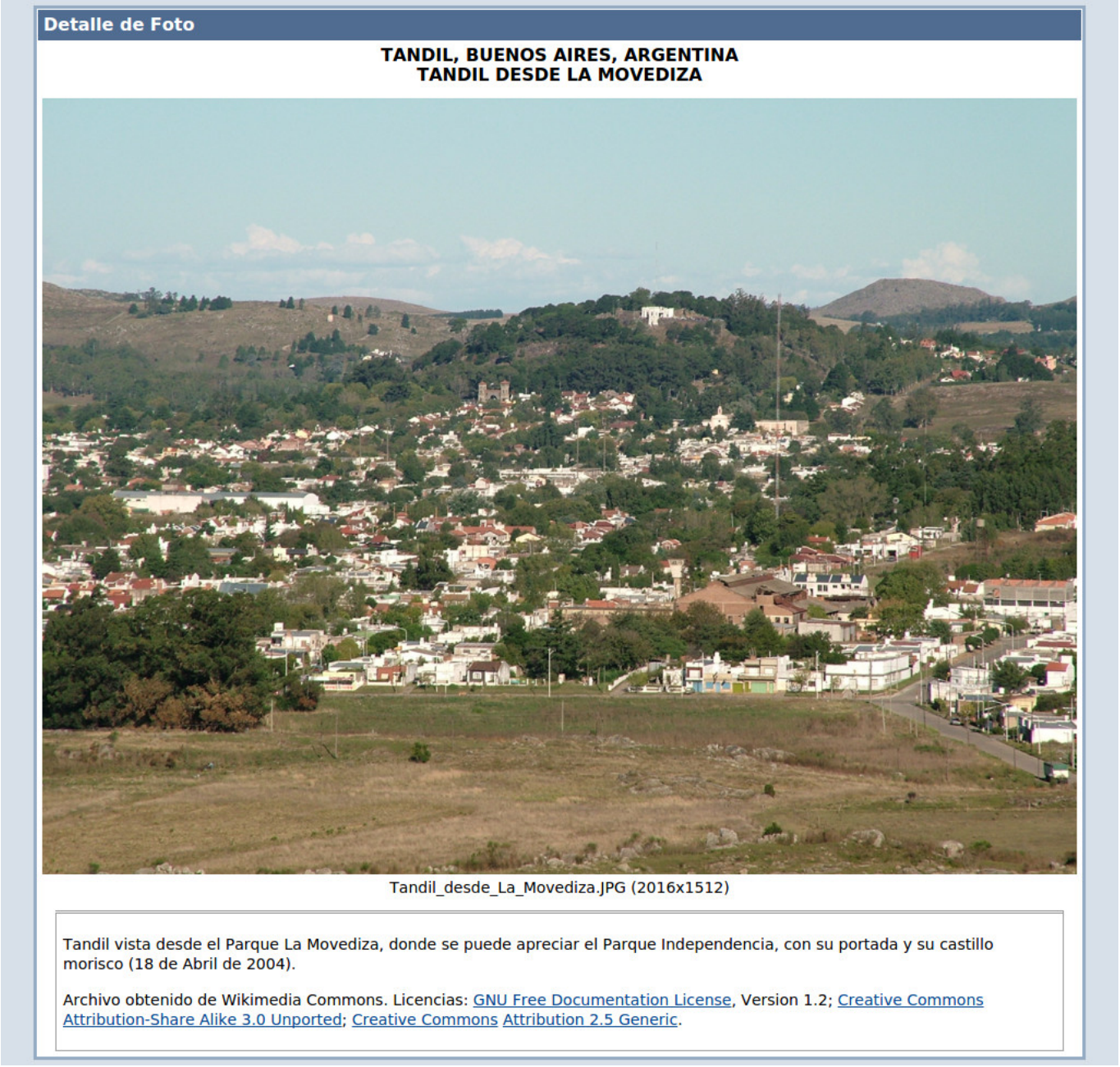


Figura 9.10: Captura de pantalla de fotolocalidad_contenido.php en la aplicación “Localidades”

Esta página está construida principalmente con código PHP. El código en `fotolocalidad_contenido-anterior.inc.php` valida los parámetros de request y obtiene los datos necesarios desde la base de datos. El código en `fotolocalidad_contenido.inc.php` produce el HTML del cuerpo de la página.

fotolocalidad_formulario.xml

Formulario de alta/baja/modificación de foto de localidad.

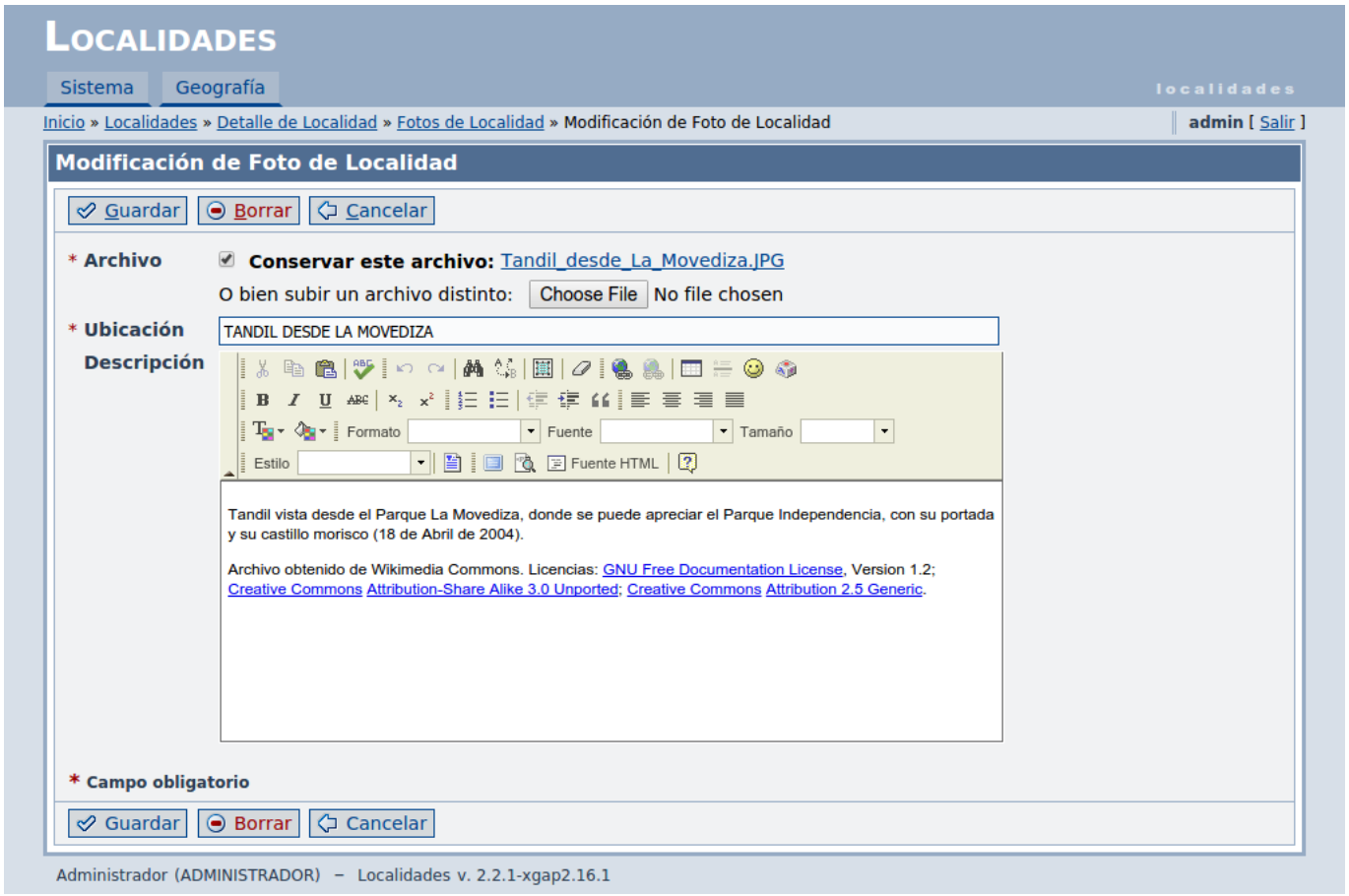


Figura 9.11: Captura de pantalla de fotolocalidad_formulario.php en la aplicación “Localidades”

Definición de campos

Los campos del formulario están definidos como sigue:

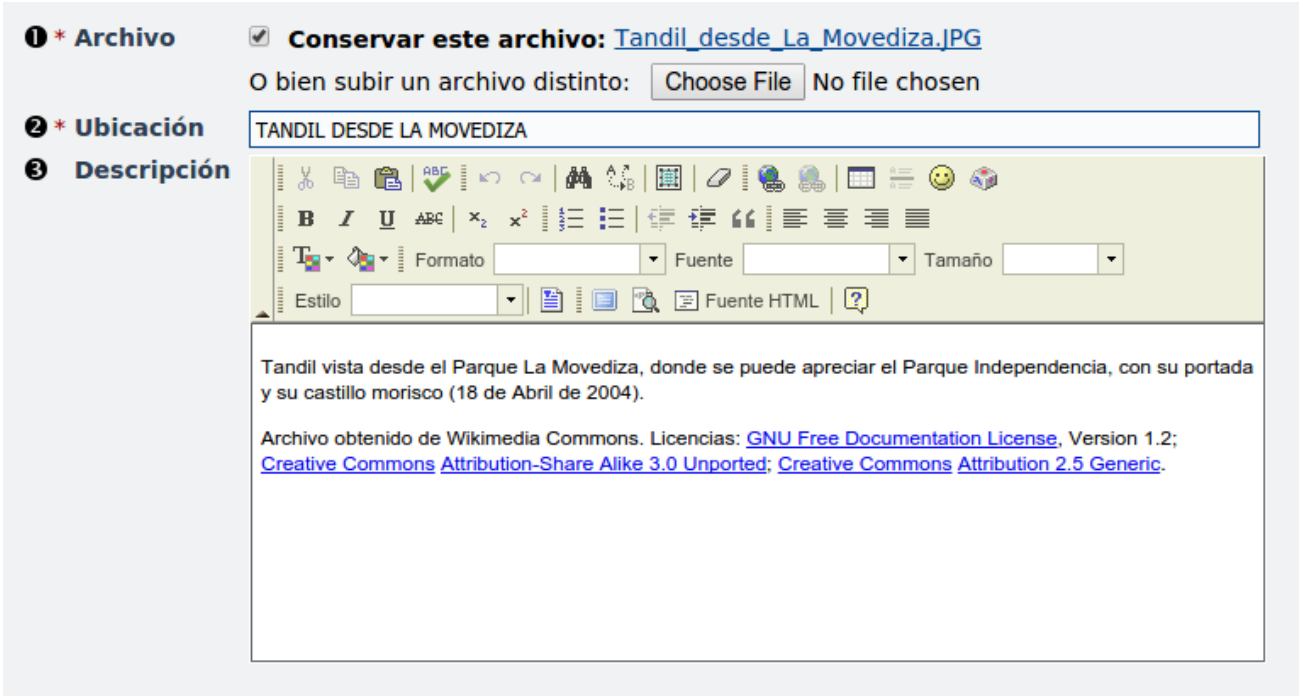
```
<tabla nombre="fotolocalidad"/>
...
<campos>
    <campo tipo="Oculto">
        <dato>nlocalidad</dato>
    </campo>
    <campo tipo="Archivo">                                    ❶
        <dato>varchivo</dato>
        <titulo>Archivo</titulo>
        <tip>Archivo de imagen</tip>
    </campo>
    <campo ancho="80">                                        ❷
        <dato>vubicacion</dato>
```

```
            <titulo>Ubicación</titulo>
            <tip>Lugar o dirección donde se tomó la foto.</tip>
            <texto-autocompletable comparacion="cont" demora="500"/>
        </campo>
        <campo tipo="HtmlArea" columnas="600" filas="300">                    ❸
            <dato>tdescripcion</dato>
            <titulo>Descripción</titulo>
        </campo>
    </campos>
```

❶ `fotolocalidad.varchivo character varying(2048) NOT NULL`. El nombre y destino del archivo subido se construye en código extra.

❷ `fotolocalidad.vubicacion character varying(100) NOT NULL`.

❸ `fotolocalidad.tdescripcion text`.



Código extra PHP en `antes_procesar_uploads`

Modifica la ruta y nombre del archivo subido:

- Los archivos se guardan separados por localidad. La ruta tiene la forma `{XGAP_CONF_UPLOAD_DIR}/localidad/fotos/{nlocalidad}/`.
- El nombre de cada archivo se prefija con la fecha y hora de alta, y el identificador del usuario que la realiza.

```
<?php
$campo_varchivo = $params['campos']['varchivo'];
if (!empty($campo_varchivo['upload'])) {
    $upload = $campo_varchivo['upload'];
    $upload->cambiarDirDestino(
        ruta_archivo(array('localidad', 'fotos', $params['registro'][' ↩
            nlocalidad'])),
        true
    );
    $upload->cambiarNombreArchivo(
        date('YmdHis') . '-' .
                Contexto::obtener('ident_usuario') .'-' .
```

```
                                $upload->nombreArchivo()
                );
            }
```

fotolocalidad_listado.xml

Listado de fotos de localidades.

Figura 9.12: Captura de pantalla de fotolocalidad_listado.php en la aplicación “Localidades”

Definición de columnas

Las columnas del listado están definidas como sigue:

```
    <columnas>
        <checkbox>                                                    ❶
            <tip>Seleccione las fotos a imprimir</tip>
            <nombre>fotosimp</nombre>
            <controlSeleccionTodos/>
        </checkbox>
        <columna intervieneEnbusqueda="no" permiteOrdenar="false">    ❷
```

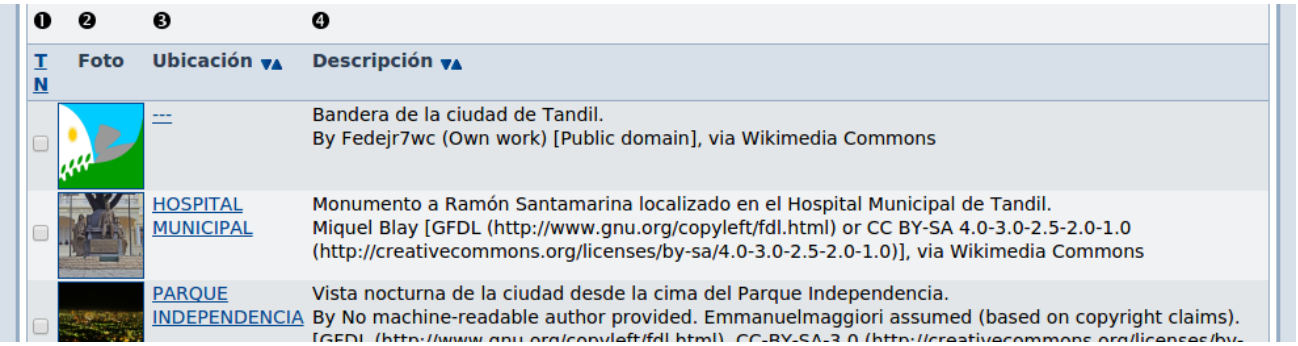
```
            <titulo>Foto</titulo>
            <dato>varchivo</dato>
            <imagen>
                <miniatura/>
                <link pasarParametros="true" historial="skip">
                    <url>fotolocalidad_contenido.php</url>
                    <target inline="true"/>
                </link>
            </imagen>
        </columna>
        <columna llevaAForm="true">                                    ❸
            <titulo>Ubicación</titulo>
            <tip>Lugar o dirección donde se tomó la foto</tip>
            <dato>vubicacion</dato>
        </columna>
        <columna>                                                      ❹
            <titulo>Descripción</titulo>
            <dato>tdescripcion</dato>
        </columna>
    </columnas>
```

❶ Columna de checkboxes, usada para seleccionar las filas a incluir en la acción "Imprimir seleccionadas".

❷ `fotolocalidad.varchivo character varying(2048)`. Columna de imágenes, que muestra miniaturas de las fotos y permite abrir fotolocalidad_contenido.php.

❸ `fotolocalidad.vubicacion character varying(100)`.

❹ `fotolocalidad.tdescripcion text`.



Especificación de variedades a generar

Sólo se genera el listado normal, dado que las demás variedades no se usan.

```
    <generacion>
        <salidas>
            <normal/>
        </salidas>
    </generacion>
```

Acción sobre filas seleccionadas

El listado provee la acción “Imprimir seleccionadas”, que genera el reporte ODT `fotolocalidad_reporte_odt` con las filas seleccionadas por el usuario a través de los checkboxes en la columna `fotosimp`.

```
    <configuracion>
        <!-- ... -->
        <acciones>
            <accion nombre="imprimir" clase="ButtonImprimir">
                <descripcion>Imprimir seleccionadas</descripcion>
                <destino historial="skip">
                    <url>fotolocalidad_reporte_odt.php</url>           ❶
```

```
                    <parametros>
                        <columna>
                            <nombre>fotosimp</nombre>                    ❷
                        </columna>
                        <parametro nombre="ancho" valor="90"/>
                    </parametros>
                </destino>
                <reset>
                    <columna>fotosimp</columna>                          ❸
                </reset>
            </accion>
        </acciones>
    </configuracion>
```

❶ La acción realiza un request a `fotolocalidad_reporte_odt.php`.

❷ Se pasa como parámetro la columna de tipo `checkbox` `fotosimp`. El resultado es que el request lleva un parámetro con nombre `fotosimp` que tiene como valor la lista de filas seleccionadas, dada como una secuencia de los valores correspondientes de `nfotolocalidad` (`/listado/clave/campo`) separados por comas.

❸ Después de ejecutar la acción se reinicia el estado de selección los checkboxes.

Código extra PHP en `despues_inicializacion`

Agrega el nombre de la ciudad al título de la página.

```
<?php
if (Request::existe('nlocalidad')) {
    $nlocalidad = intval(Request::obtener('nlocalidad'), 10);
    $sql = "SELECT vnombrecompleto FROM vlocalidad WHERE nlocalidad =  ↩
        $nlocalidad";
    $localidad = $params['conexion']->obtenerPrimero($sql);
    if (!empty($localidad)) {
        $params['titulo'] .= ' &ndash; ' . preparar_salida($localidad);
    }
}
```

fotolocalidad_reporte_odt.xml

Reporte ODT invocado desde la acción "Imprimir seleccionadas" en `fotolocalidad_listado_xml`.

TANDIL, BUENOS AIRES, ARGENTINA

Tandil es la ciudad cabecera del partido homónimo y está ubicada en el centro de la provincia de Buenos Aires, en el centro-este de la Argentina, sobre las sierras del sistema de Tandilia.

Fue fundada por el brigadier general Martín Rodríguez, gobernador de la provincia de Buenos Aires, en 1823, con el nombre de Fuerte Independencia.

Su población inicial fue de 400 habitantes y debió soportar el asedio permanente de los indígenas, que trataron de impedir la consolidación del poblado.

Su núcleo urbano actual se divide en veintiséis barrios.

Su economía está basada principalmente en la agricultura, la ganadería, el turismo –que ha tenido un importante crecimiento en los últimos años, especialmente en el período vacacional que corresponde con la celebración de la semana Santa donde se desarrolla la liturgia alusiva a la fecha. – y la actividad minera –la que está siendo cuestionada por organizaciones ecologistas que quieren impedir que desaparezca el paisaje serrano original–. También se han instalado, entre otras, varias empresas de desarrollo de software, a medida que crece su reciente polo tecnológico.

Su clima es templado, con temperaturas medias de 13,7° C.

(Fuente: Wikipedia Español)

PARQUE INDEPENDENCIA

Monumento al general Martín Rodríguez, fundador de Tandil, en el Parque Independencia. Hecho en bronce por el artista Arturo Dresco.
By Arturo Dresco (Own work) [CC BY-SA 3.0 (http://creativecommons.org/licenses/by-sa/3.0)], via Wikimedia Commons

Figura 9.13: Primera página de un reporte generado por fotolocalidad_reporte_odt.php en la aplicación “Localidades”

```
<configuracion>
    <template path="templates/fotolocalidad.odt">                          ❶
        <xslt-adicional incluir-templates-predefinidos="true"/>            ❷
    </template>
    <xml nombre="fotos_localidad"/>
    <reporte nombre="fotos_localidad" incluir-fechahora-emision="true">
        <meta>                                                             ❸
            <item nombre="dc:title"><![CDATA[Fotografías de {$ ↩
                var_nombrecompletolocalidad}]]></item>
            <item nombre="dc:language">es</item>
            <item nombre="meta:editing-cycles">1</item>
            <item nombre="meta:editing-duration">P0D</item>
            <item nombre="meta:creation-date"><![CDATA[{{ date('Y-m-d\TH:i:s') }}]]></ ↩
                item>
            <item nombre="meta:initial-creator"><![CDATA[{{ Contexto::obtener(' ↩
                nombre_usuario') }}]]></item>
        </meta>
    </reporte>
</configuracion>
<variables>                                                                ❹
    <variable nombre="nlocalidad" tipo="request">
        <requerida permitir-vacio="false" cero-es-vacio="true"/>
    </variable>
    <variable nombre="fotosimp_nfotolocalidad" tipo="request"/>
</variables>
<php>
    <![CDATA[
    function sanitize_db_serial($valor) {
        return filter_var(
            $valor,
            FILTER_VALIDATE_INT,
            array('options' => array('default' => 0, 'min_range' => 1))
        );
    }

    $var_nlocalidad = sanitize_db_serial($var_nlocalidad);                 ❺

    $html_entity_decode_flags = ENT_QUOTES;                                ❻
    if (defined('ENT_XHTML')) { // PHP >= 5.4.0
        $html_entity_decode_flags |= constant('ENT_XHTML');
    }
    ]]>
</php>
<consultas>
    <consulta>                                                             ❼
        <sql>
            <![CDATA[
            SELECT vnombrecompleto, tdescripcion
            FROM vlocalidad
            WHERE nlocalidad = ?
            ]]>
        </sql>
        <parametros>
            <parametro nombre="nlocalidad"/>
        </parametros>
        <variables>
            <variable columna="vnombrecompleto" nombre="nombrecompletolocalidad"/>
            <variable columna="tdescripcion" nombre="descripcionlocalidad"/>
        </variables>
    </consulta>
</consultas>
```

```
<php>
    <![CDATA[
    $var_fotosimp_nfotolocalidad = implode(                                    ❽
        ',',
        array_filter(
            array_map(
                'sanitize_db_serial',
                explode(
                    ',',
                    $var_fotosimp_nfotolocalidad
                )
            )
        )
    );
    if (!empty2($var_fotosimp_nfotolocalidad, false, true)) {                  ❾
    ]]>
</php>
<iterator nombre="fotos">                                                      ❿
    <consulta>                                                                 ⓫
        <sql>
            <![CDATA[
            SELECT varchivo, vubicacion, tdescripcion
            FROM vfotolocalidad
            WHERE nfotolocalidad IN (?)
            ORDER BY vubicacion, nfotolocalidad DESC
            ]]>
        </sql>
        <parametros>
            <parametro nombre="fotosimp_nfotolocalidad"/>
        </parametros>
        <variables>
            <variable columna="varchivo" nombre="archivo"/>
            <variable columna="vubicacion" nombre="ubicacion"/>
            <variable columna="tdescripcion" nombre="descripcion"/>
        </variables>
    </consulta>
    <contenido>
        <php>
            <![CDATA[
            $var_descripcion = html_entity_decode(
                strip_tags($var_descripcion),
                $html_entity_decode_flags
            );                                                                 ⓬
            $var_archivo = realpath(
                ruta_archivo(array(Upload::dirBase(), $var_archivo), DIR_SEP, false)
            );                                                                 ⓭
            if (!$var_archivo) {
                $var_archivo = '';
            }
            ]]>
        </php>
        <xml nombre="foto">                                                    ⓮
            <![CDATA[
            <foto>
                <ubicacion>&lt;![CDATA[{$var_ubicacion}]]&gt;</ubicacion>
                <descripcion>&lt;![CDATA[{$var_descripcion}]]&gt;</descripcion>
                <archivo>&lt;![CDATA[{$var_archivo}]]&gt;</archivo>
            </foto>
            ]]>
        </xml>
    </contenido>
</iterator>
```

```
<php>
    <![CDATA[
    } else { // empty2($var_fotosimp_nfotolocalidad, false, true)        ⓯
        $odt_xml_node_fotos = '';
    }
    ]]>
</php>
<!--
Estructura generada:
<localidad>
1   <nombre></nombre>
1   <descripcion></descripcion>
*   <foto>
        <ubicacion></ubicacion>
        <descripcion></descripcion>
        <archivo></archivo>
    </foto>
</localidad>
-->
<xml nombre="fotos_localidad" main="si">                                ⓰
    <![CDATA[
    <localidad>
        <nombre>&lt;![CDATA[{$var_nombrecompletolocalidad}]]&gt;</nombre>
        <descripcion>&lt;![CDATA[{$var_descripcionlocalidad}]]&gt;</descripcion>
        {$odt_xml_node_fotos}
    </localidad>
    ]]>
</xml>
```

❶ La plantilla del reporte es el archivo `templates/fotolocalidad.odt`.

❷ Se incluyen los templates xslt predefinidos, para usarlos en `fotolocalidad.odt`, como se muestra en la descripción de la plantilla.

❸ Se personalizan los metadatos que va a contener el ODT generado.

❹ Se definen variables a partir de los parámetros de request, para usar más adelante.

❺ Saneado de una variable con valor proveniente del request. La otra variable definida hasta ese punto se procesa más adelante.

❻ Más adelante será necesario usar `html_entity_decode()`, dentro de un iterador. Aquí se define el valor de su segundo parámetro.

❼ Consulta para obtener los datos necesarios acerca de la localidad solicitada, cuya clave está dada por la variable `nlocalidad` que se definió a partir del request. Con el resultado de esta consulta se definen las variables `nombrecompletolocalidad` y `descripcionlocalidad`.

❽ Saneado de una variable con valor proveniente del request. En este caso se asegura que el valor se pueda utilizar en una cláusula SQL `IN` sin riesgo de inyección de código.

❾ Sólo se debe hacer la consulta para obtener los datos de las fotos si la especificación de sus claves no es vacía.

❿ Iterador para construir los datos de las fotos solicitadas.

⓫ Consulta para obtener los datos de las fotos solicitadas, cuyas claves se recibieron en el request y se sanearon en ❽. Para cada fila se asignan valores a las variables `archivo`, `ubicacion` y `descripcion`, que se usan a continuación.

⓬ La descripción de la foto tiene formato HTML con entidades codificadas. Aquí se eliminan los elementos HTML y se decodifican las entidades, para que se pueda incluir sin problemas en el ODT.

⓭ Se construye la ruta completa al archivo de imagen.

⓮ Para cada fila retornada por la consulta, se construye un fragmento de XML con los valores de esa fila. El fragmento XML final para el iterador resulta de la concatenación de los fragmentos de cada iteración.

⓯ `else` del `if` en ❾: si no hay claves de fotos a obtener, el fragmento de XML correspondiente queda vacío.

⓰ El elemento `xml` principal define el XML final que se procesa con la plantilla dada en ❶. Usa las variables definidas en ❼ y el fragmento de XML creado por el iterador ❿ (o el fragmento vacío en ⓯).

El XML generado por el código descripto, que produce el reporte mostrado al comienzo de esta sección, es el siguiente:

```
<localidad>
  <nombre><![CDATA[TANDIL, BUENOS AIRES, ARGENTINA]]></nombre>
  <descripcion><![CDATA[Tandil es la ciudad cabecera del partido homónimo y está ubicada en ↩
       el centro de la provincia de Buenos Aires, en el centro-este de la Argentina, sobre  ↩
      las sierras del sistema de Tandilia.

Fue fundada {...}

Su clima es templado, con temperaturas medias de 13,7\textdegree{} C.

(Fuente: Wikipedia Español)]]></descripcion>
  <foto>
    <ubicacion><![CDATA[PARQUE INDEPENDENCIA]]></ubicacion>
    <descripcion><![CDATA[Monumento al general Martín Rodríguez, fundador de Tandil, en el  ↩
        Parque Independencia. Hecho en bronce por el artista Arturo Dresco.
By Arturo Dresco (Own work) [CC BY-SA 3.0 (http://creativecommons.org/licenses/by-sa/3.0)], ↩
     via Wikimedia Commons]]></descripcion>
    <archivo><![CDATA[{ruta completa a XGAP_CONF_UPLOAD_DIR}/localidad/fotos ↩
        /1/20151113125222-1-MonumentoGralRodriguez.jpg]]></archivo>
  </foto>
  <foto>
    <ubicacion><![CDATA[TANDIL DESDE LA MOVEDIZA]]></ubicacion>
    <descripcion><![CDATA[La ciudad rodeada de las sierras. {...}]]></descripcion>
    <archivo><![CDATA[{ruta completa a XGAP_CONF_UPLOAD_DIR}/localidad/fotos ↩
        /1/20151113124722-1-Tandil_desde_La_Movediza_2.JPG]]></archivo>
  </foto>
  <foto>
    <ubicacion><![CDATA[VILLA DEL LAGO]]></ubicacion>
    <descripcion><![CDATA[Monumento de Don Quijote y Sancho Panza, {...}]]></descripcion>
    <archivo><![CDATA[{ruta completa a XGAP_CONF_UPLOAD_DIR}/localidad/fotos ↩
        /1/20151113123118-2-tandil-monumento_don_quijote-28-12-08_1834.jpg]]></archivo>
  </foto>
</localidad>
```

La plantilla `templates/fotolocalidad.odt` contiene los elementos que se describen a continuación, los cuales construyen el ODT final a partir del XML generado.

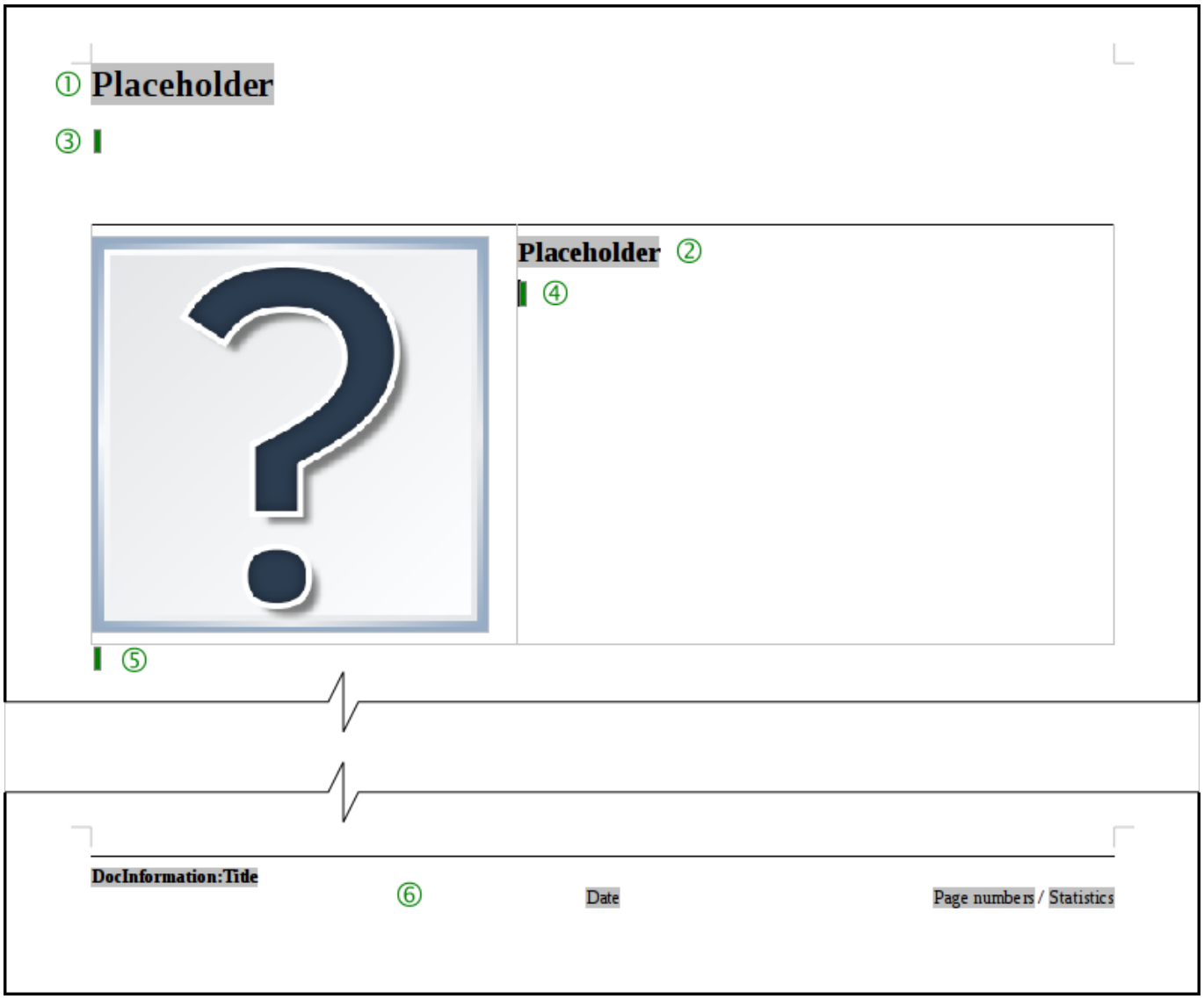


Figura 9.14: Elementos en la plantilla fotolocalidad.odt, en la aplicación “Localidades”

① y ② son campos placeholder de tipo text (Insert → Fields → More Fields; Pestaña Functions, Type=Placeholder, Format=Text). ③, ④ y ⑤ son scripts con tipo `ODF-XSLT` (Insert → Script; Script type=`ODF-XSLT`).

①

Campo placeholder con valores Placeholder=`LOCALIDAD` y Reference=`/localidad/nombre`.

②

Campo placeholder con valores Placeholder=`UBICACION` y Reference=`ubicacion`. La referencia es relativa porque el campo se encuentra dentro del ciclo que recorre cada foto, con lo cual el elemento de contexto en este punto es `/localidad/foto`.

③

Script para incluir la descripción de la localidad:

```
{@before ancestor::text:p[1]                                          ❶
  <xslt:if test="normalize-space(/localidad/descripcion) != ''">}
{@after ancestor::text:p[1]                                           ❷
  </xslt:if>}
```

```
{@replace .                                                                    ❸
  <xslt:call-template name="eol-to-line-break">
    <xslt:with-param name="src" select='/localidad/descripcion'/>
  </xslt:call-template>}
```

❶ Se evita emitir el párrafo que corresponde a la descripción de la localidad cuando ésta es vacía.

❷ Cierre del `xslt:if` anterior.

❸ Se reemplaza el nodo actual con la descripción de la localidad.

④

Script para incluir la descripción de la foto:

```
{@replace .                                                                    ❶
  <xslt:choose>
    <xslt:when test="normalize-space(descripcion) != ''">
      <xslt:call-template name="eol-to-line-break">
        <xslt:with-param name="src" select='descripcion'/>
      </xslt:call-template>
    </xslt:when>
    <xslt:otherwise>(sin descripción)</xslt:otherwise>                         ❷
  </xslt:choose>}
```

❶ Se reemplaza el nodo actual con la descripción de la foto.

❷ Si la foto no tiene descripción, se incluye un texto por defecto.

⑤

Script para recorrer las fotos incluidas y referenciar el archivo de imagen de cada una:

```
{@before //table:table[@table:name="TablaFoto"]                                ❶
  <xslt:for-each select="/localidad/foto">}

{@after //table:table[@table:name="TablaFoto"]                                 ❷
  </xslt:for-each>}

{@child //draw:frame[@draw:name="foto"]/draw:image                             ❸
  <xslt:if test="normalize-space(archivo) != ''">                              ❹
    <xslt:attribute name="xlink:href">                                         ❺
      <xslt:value-of select="archivo"/>
    </xslt:attribute>
  </xslt:if>
}
```

❶ Ciclo para procesar cada foto incluida, que encierra la tabla con nombre `TablaFoto`. El resultado del ciclo es que esta tabla se repite para cada una de las fotos incluidas, en tanto que sus contenidos se reemplazan con los valores propios de la foto por medio de los placeholders y scripts que quedan dentro del ciclo.

❷ Cierre del `xslt:for-each` anterior.

❹ Si no hay un archivo, se mantiene la imagen original del template.

❸, ❺ Se agrega un atributo `xlink:href` que apunta al archivo de la foto, como hijo del elemento `draw:image` correspondiente a la imagen insertada en la plantilla ODT con nombre `foto`.

⑥

El contenido del pie de página no depende de los valores recibidos en el XML de entrada; sólo se incluyeron campos comunes, no procesados por la transformación XSLT.

Notar que en ③ y ④ se hace un llamado al template XSLT `eol-to-line-break`, para preservar en el documento generado los saltos de línea que pudiera haber en el texto origen (`/localidad/descripcion` y `/localidad/foto/descripcion`, respectivamente); si no se hiciera así, los saltos de línea serían ignorados. Dicho template está disponible porque en `fotolocalidad_reporte_odt.xml` se indica que se deben incluir los templates predefinidos por XGAP (`/pagina/configuracion/template/xslt-adicional/@incluir-templates-predefinidos="true"`).

guardarindicepaginas_contenido.xml

Actualiza el índice de páginas en la base de datos (tabla `seguridad.pagina`).

La implementación de la funcionalidad de la página se encuentra en el archivo `extras/guardarindicepaginas_contenido-anterior.inc.php`; el código que genera la salida HTML está en `extras/guardarindicepaginas_contenido.inc.php`.

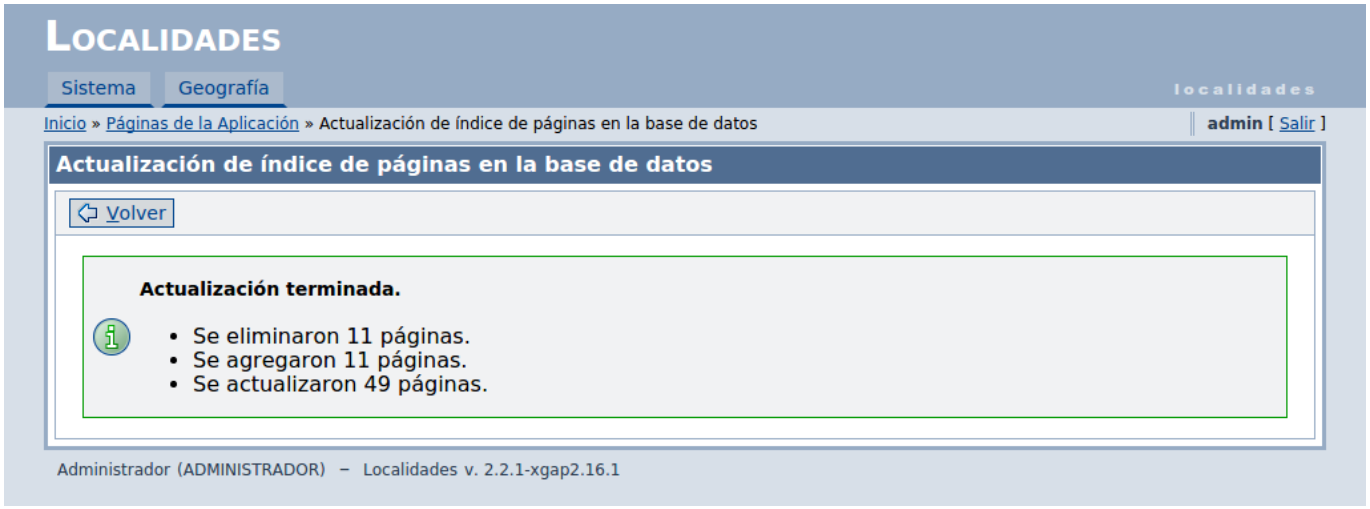


Figura 9.15: Captura de pantalla de guardarindicepaginas_contenido.php en la aplicación “Localidades”

Uso de mensajes de página

XGAP provee un mecanismo para registrar mensajes que se deben mostrar al usuario en un request particular a una página. Esta página lo utiliza para informar cualquier problema que pudiera producirse durante la ejecución de su funcionalidad principal.

En `guardarindicepaginas_contenido-anterior.inc.php`:

```
<?php
//...
if (isset($indice_paginas)) {
    //...
    if ($rs) {
        //...
    } else {
        Pagina::instancia()->mensajes()->agregar(ObjetoMensaje::nuevo(
            'No se pudieron limpiar las páginas sin permisos.',
            Mensaje::TIPO_ALERTA)
        );
    }
    //...
    if ($n_errores > 0) {
        Pagina::instancia()->mensajes()->agregar(ObjetoMensaje::nuevo(
            "Hubo errores en la actualización de $n_errores páginas.",
            Mensaje::TIPO_ALERTA)
        );
    }
    //...
} else {
    Pagina::instancia()->mensajes()->agregar(ObjetoMensaje::nuevo(
        'No se encuentra el índice de páginas.',
        Mensaje::TIPO_ERROR)
    );
```

```
}
```

En `guardarindicepaginas_contenido.inc.php`:

```
<?php
//...
if (!Pagina::instancia()->mensajes()->isEmpty()) {
    Mensaje::mostrarLista(
        Pagina::instancia()->mensajes(),
        true,
        null,
        'cont-msjs-pag'
    );
}
//...
```

El último fragmento de código se encarga de mostrar la lista de mensajes que fueron agregados en el request actual. Este código ya se encuentra incluido en los otros tipos de página, pero en las de tipo Contenido se debe agregar manualmente en el lugar deseado.

Presentación de resultados

El único contenido que se emite en el cuerpo de esta página es un reporte de resultados.

```
<?php
if ($agregadas > 0 || $actualizadas > 0 || $eliminadas > 0) {
    Mensaje::info(
        Html::elemento(
            'p',
            Html::elemento(
                'strong',
                'Actualizaci&oacute;n terminada.',
                null, null, null, true
            ),
            null, 'first', null, true
        ) . Html::lista(
            array(
                $eliminadas == 0
                        ? 'No se eliminaron p&aacute;ginas.'
                        : "Se eliminaron $eliminadas p&aacute;ginas.",
                $agregadas == 0
                        ? 'No se agregaron p&aacute;ginas.'
                        : "Se agregaron $agregadas p&aacute;ginas.",
                $actualizadas == 0
                        ? 'No se actualizaron p&aacute;ginas.'
                        : "Se actualizaron $actualizadas p&aacute;ginas."
            ),
            false, null, 'last', null, null, true
        ),
        null, 'icon'
    );
} else {
    Html::elemento(
        'p',
        'El &iacute;ndice ya estaba actualizado. No se realizaron cambios.'
    );
}
```

index_admin_contenido.xml

Página de inicio para el rol ADMINISTRADOR. Presenta un resumen de la cantidad de elementos que hay en las entidades principales del modelo de datos.

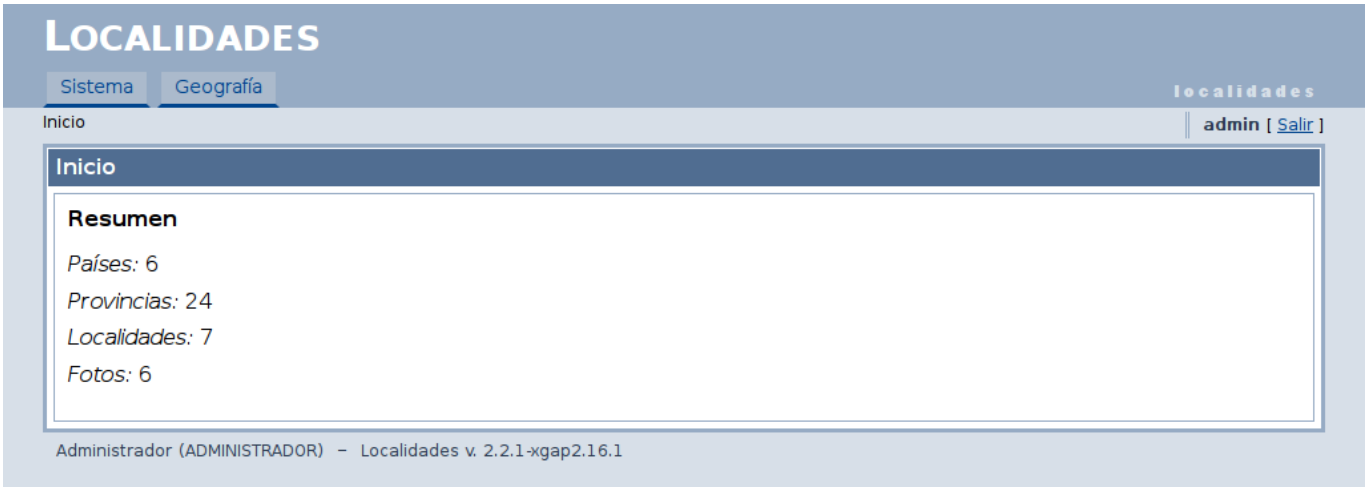


Figura 9.16: Captura de pantalla de index_admin_contenido.php en la aplicación "Localidades"

---

sugerencia

El código PHP de los elementos `/pagina/contenido-anterior` y `/pagina/contenido` se colocó en archivos `.php` separados, para simplificar su edición: si este código se edita como texto genérico dentro de un bloque XML CDATA, no se pueden aprovechar las facilidades específicas para el lenguaje PHP que ofrecen muchos editores, como resaltado de sintaxis y errores o autocompletado. Esta técnica se utiliza también en otros fuentes XML.

```
    <contenido-anterior>
        <![CDATA[
        include 'index_admin_contenido-anterior.inc.php';
        ]]>
    </contenido-anterior>
    <contenido>
        <![CDATA[
        <?php include 'index_admin_contenido-contenido.inc.php'; ?>
        ]]>
    </contenido>
```

Los archivos `index_admin_contenido-anterior.inc.php` y `index_admin_contenido-contenido.inc.php` se encuentran dentro del subdirectorio `extras`, para que se copien automáticamente al directorio de salida durante la generación de la aplicación.

Contenido extra en la cabecera HTML

Ver acercade_contenido.xml.

Contenido de la página

En `index_admin_contenido-anterior.inc.php`:

```
<?php
//...
$n_paises = $objeto_conexion->obtenerPrimero(
    'SELECT COUNT(npais) FROM pais'
);
$n_provincias = $objeto_conexion->obtenerPrimero(
    'SELECT COUNT(nprovincia) FROM provincia'
);
$n_localidades = $objeto_conexion->obtenerPrimero(
    'SELECT COUNT(nlocalidad) FROM localidad'
);
$n_fotos = $objeto_conexion->obtenerPrimero(
```

```
    'SELECT COUNT(nfotolocalidad) FROM fotolocalidad'
);
```

En `index_admin_contenido-contenido.inc.php`:

```
<?php
//...
$info = '';
$resumen = array();
if (!is_null($n_paises)) {
    $resumen[] = array('Países', $n_paises);
}
if (!is_null($n_provincias)) {
    $resumen[] = array('Provincias', $n_provincias);
}
if (!is_null($n_localidades)) {
    $resumen[] = array('Localidades', $n_localidades);
}
if (!is_null($n_fotos)) {
    $resumen[] = array('Fotos', $n_fotos);
}
if (!empty($resumen)) {
    foreach ($resumen as $n => $v) {
        $resumen[$n] = Html::elemento (
            'em',
            preparar_salida("{$v[0]}:", array(), null, true, true),
            null, null, null, true
        ) .
            ' ' . $v[1];
    }
    $info .= Html::abrirDiv('resumen', null, null, true) .
            Html::elemento('h2', 'Resumen', null, null, null, true) .
            Html::lista($resumen, false, null, null, null, null, true) .
            Html::cerrarDiv(true);
}
print $info;
//...
```

localidad_formulario.xml

Formulario de alta/baja/modificación de localidad.

Figura 9.17: Captura de pantalla de localidad_formulario.php en la aplicación “Localidades”

Definición de campos

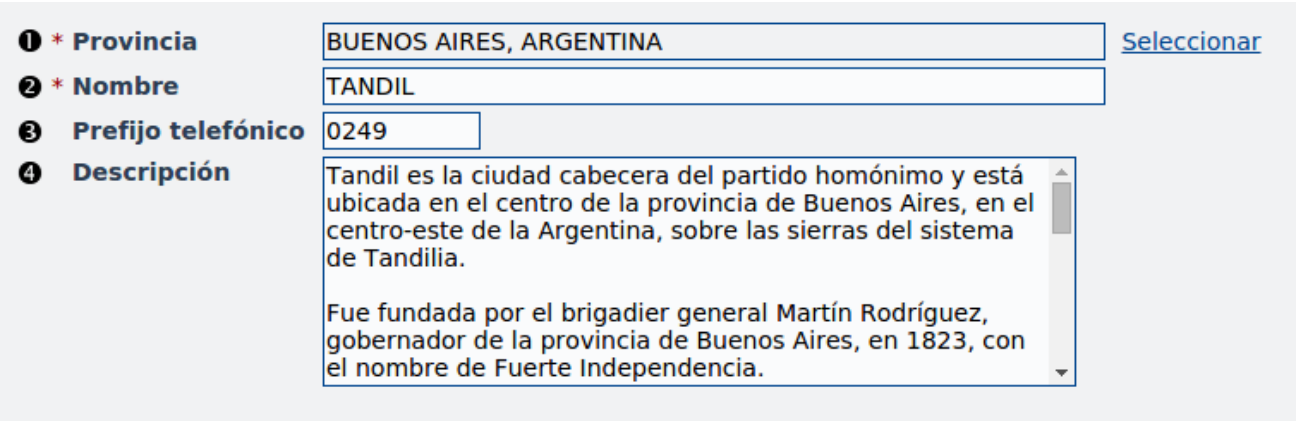
Los campos del formulario están definidos como sigue:

```
<tabla nombre="localidad"/>
...
<campos>
    <campo tipo="Seleccionable" ancho="60">                          ❶
        <titulo>Provincia</titulo>
        <dato>nprovincia</dato>
        <entidad>provincia</entidad>
        <tabla>vprovincia</tabla>
        <descripcion>vnombrecompleto</descripcion>
        <seleccionar>nprovincia</seleccionar>
    </campo>
    <campo ancho="60">                                               ❷
        <titulo>Nombre</titulo>
        <dato>vnombre</dato>
    </campo>
    <campo>                                                          ❸
        <titulo>Prefijo telefónico</titulo>
        <dato>cpreftelef</dato>
    </campo>
    <campo tipo="TextAreaSinTamanio" filas="8"
           columnas="60" mayusculas="false">                         ❹
        <titulo>Descripción</titulo>
        <dato>tdescripcion</dato>
    </campo>
</campos>
```

❶ `localidad.nprovincia integer NOT NULL`. El Seleccionable abre `provincia_listado_seleccion.php`.

❷ `localidad.vnombre character varying(100) NOT NULL`.

3. `localidad.cpreftelef character(5).`
4. `localidad.tdescripcion text.`



localidad_listado.xml

Listado de Localidades.

Figura 9.18: Captura de pantalla de localidad_listado.php en la aplicación “Localidades”

Definición de columnas

Las columnas del listado están definidas como sigue:

```
<columnas>
    <columna>                                                    ❶
        <titulo>País</titulo>
        <dato>vnombrepais</dato>
    </columna>
    <columna>                                                    ❷
        <titulo>Provincia</titulo>
        <dato>vnombreprovincia</dato>
    </columna>
    <columna llevaAForm="true">                                  ❸
        <titulo>Nombre</titulo>
        <dato>vnombre</dato>
    </columna>
    <columna>                                                    ❹
        <titulo>Pref. tel.</titulo>
        <tip>Prefijo telefónico</tip>
        <dato>cpreftelef</dato>
    </columna>
    <columna>                                                    ❺
        <titulo>Descripción</titulo>
        <dato>tdescripcion</dato>
        <recortar longitud="100" prefijo="" sufijo=" ..."/>
    </columna>
</columnas>
```

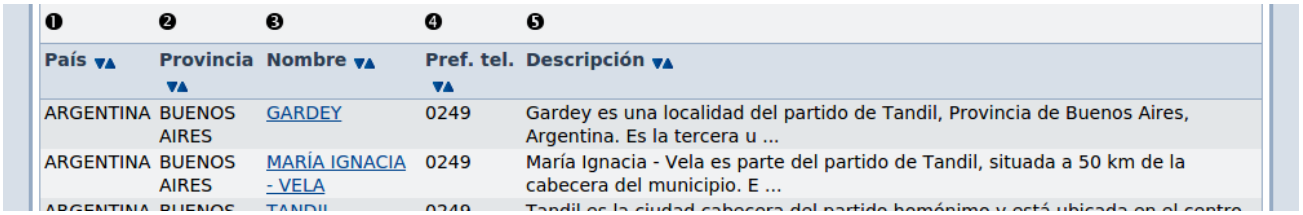
❶ `vlocalidad.vnombrepais character varying(100).`

❷ `vlocalidad.vnombreprovincia character varying(50).`

❸ `vlocalidad.vnombre character varying(100).` La columna provee acceso al master de Localidad, dado que incluye `@llevaAForm="true"` y `/listado/entidad/@master="si"`.

❹ `vlocalidad.cpreftelef character(5).`

❺ `vlocalidad.tdescripcion text`. Se limita la longitud máxima del texto que se puede presentar en esta columna, haciendo uso del elemento `columna/recortar`.



❶ País	❷ Provincia	❸ Nombre	❹ Pref. tel.	❺ Descripción
ARGENTINA	BUENOS AIRES	GARDEY	0249	Gardey es una localidad del partido de Tandil, Provincia de Buenos Aires, Argentina. Es la tercera u ...
ARGENTINA	BUENOS AIRES	MARÍA IGNACIA - VELA	0249	María Ignacia - Vela es parte del partido de Tandil, situada a 50 km de la cabecera del municipio. E ...
ARGENTINA	BUENOS	TANDIL	0249	Tandil es la ciudad cabecera del partido homónimo y está ubicada en el centro

Especificación de variedades a generar

Sólo se genera el listado normal y la exportación en PDF, ODS y CSV.

```
<salidas>
    <normal/>
    <pdf/>
    <ods/>
    <csv/>
</salidas>
```

Buscador personalizado

El buscador del listado se personaliza para ofrecer como criterios de búsqueda los nombres de país, provincia y localidad.

```
<!-- ... -->
<configuracion>
    <!-- ... -->
    <buscador>
```

```
            <personalizado-simple>                                          ❶
                <texto nombre="pvnombrepais" valor="$pvnombrepais"
                       etiqueta="País" ancho="60" teclaacceso="a"/>
                <texto nombre="pvnombreprovincia" valor="$pvnombreprovincia"
                       etiqueta="Provincia" ancho="60" teclaacceso="r"/>
                <texto nombre="pvnombre" valor="$pvnombre"
                       etiqueta="Nombre" ancho="60" teclaacceso="n"/>
            </personalizado-simple>
        </buscador>
        <!-- ... -->
    </configuracion>
    <consulta>
        <!-- ... -->
        <condiciones_de_parametros>                                         ❷
            <and>
                <equal>                                                     ❸
                    <col>vnombrepais</col>
                    <param destacar="no">pvnombrepais</param>
                </equal>
                <like>                                                      ❹
                    <col>vnombreprovincia</col>
                    <param destacar="si">pvnombreprovincia</param>
                </like>
                <like>                                                      ❺
                    <col>vnombre</col>
                    <param destacar="si">pvnombre</param>
                </like>
            </and>
        </condiciones_de_parametros>
    </consulta>
<!-- ... -->
```

❶ Los campos del buscador se definen en el elemento `personalizado-simple`. Este buscador, en particular, contiene tres campos de texto.

❷ La consulta incluye el elemento `condiciones_de_parametros`, para filtrar los resultados de acuerdo a los valores que haya en los campos del buscador. Se define una condición para cada uno de los tres campos y se combinan las condiciones con AND para que el filtro considere simultáneamente todos los criterios que ingrese el usuario.

❸, ❹, ❺ Las tres condiciones comparan una columna de la consulta (`col`) con el valor de una variable (`param`), definida en este caso por el campo correspondiente. El contenido del elemento `param` hace referencia al nombre del campo.

localidad_master.xml

Detalle de una localidad, con acceso a entidades dependientes.

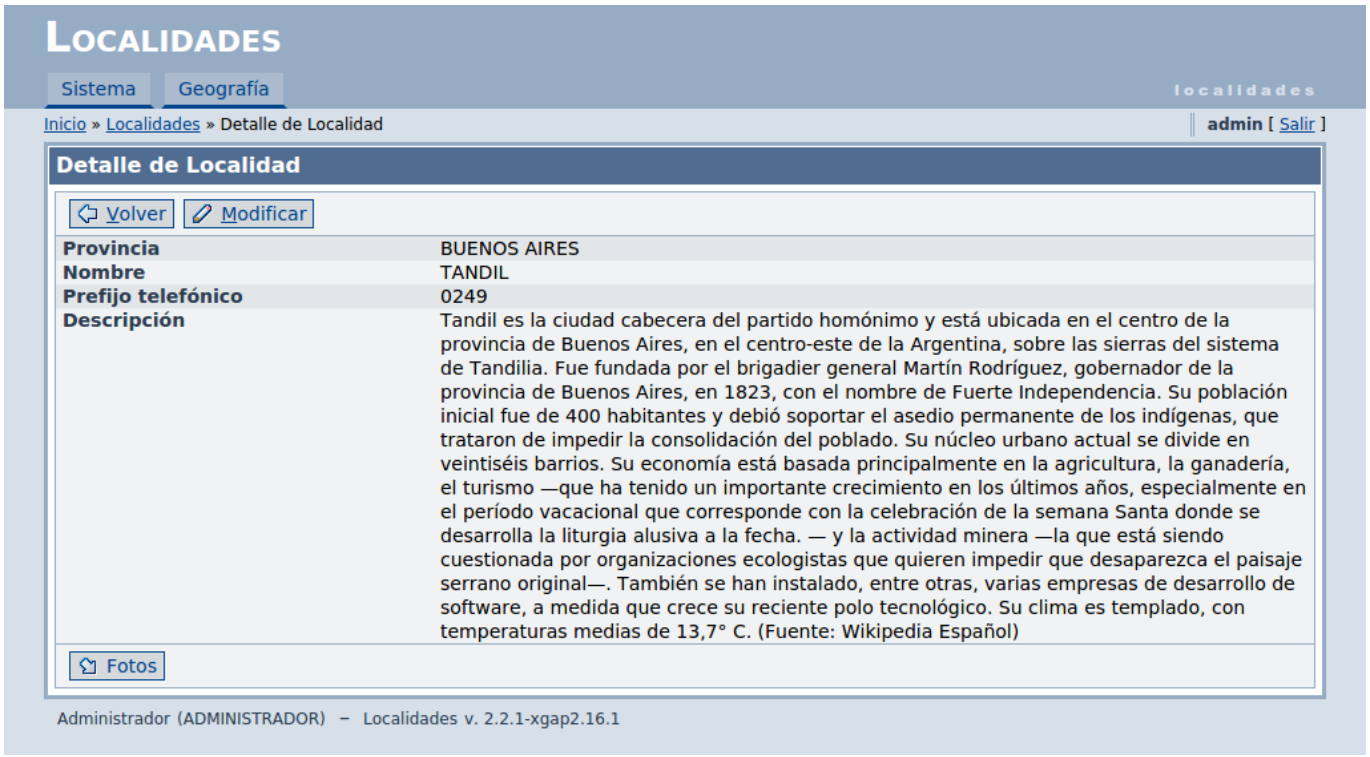


Figura 9.19: Captura de pantalla de localidad_master.php en la aplicación “Localidades”

Definición de campos

Los campos que se muestran en el master están definidos como sigue:

```
<tabla nombre="vlocalidad"/>
<!-- ... -->
<campos>
    <campo>
        <titulo>Provincia</titulo>
        <dato>vnombreprovincia</dato>
    </campo>
    <campo>
        <titulo>Nombre</titulo>
        <dato>vnombre</dato>
    </campo>
    <campo>
        <titulo>Prefijo telefónico</titulo>
        <dato>cpreftelef</dato>
    </campo>
    <campo>
        <titulo>Descripción</titulo>
        <dato>tdescripcion</dato>
    </campo>
</campos>
```

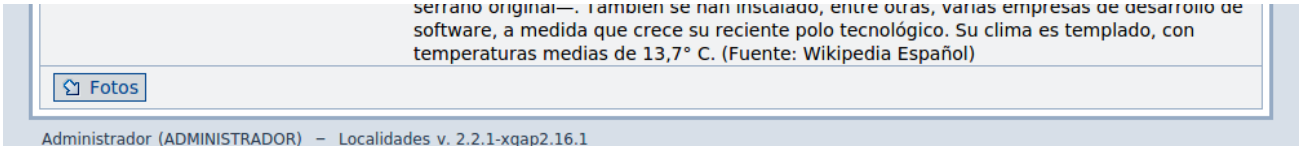
Acceso a entidades dependientes

En este modelo, la entidad `localidad` tiene como dependiente a la entidad `fotolocalidad`. En el master se usa el elemento `details` para proveer acceso a las Fotos de la Localidad que se está consultando.

```
<details>
    <detail>
        <descripcion>Fotos</descripcion>
```

```
            <entidad>fotolocalidad</entidad>
            <campos>
                <campo detail="nlocalidad" master="nlocalidad"/>
            </campos>
        </detail>
    </details>
```

La especificación usada para `detail`, sin incluir el atributo `detail/@listado="true"`, produce como resultado un botón en la parte inferior de la página, que lleva al listado de la entidad indicada — `fotolocalidad_listado.php` en este caso. El mismo resultado se obtiene incluyendo `detail/@tradicional="true"`.



También es posible presentar dentro del mismo master el listado de la entidad dependiente, agregando a la especificación el atributo `detail/@listado="true"` y opcionalmente usando el atributo `detail/@tipo-listado` para indicar el tipo de listado a usar.

paginaapp_formulario.xml

Formulario para establecer los roles que tienen permiso de acceso a una página de la aplicación. Debe recibir como parámetro el identificador de la página y opera sólo sobre las asociaciones entre ella y los roles, sin hacer modificaciones a los datos de la página en sí.

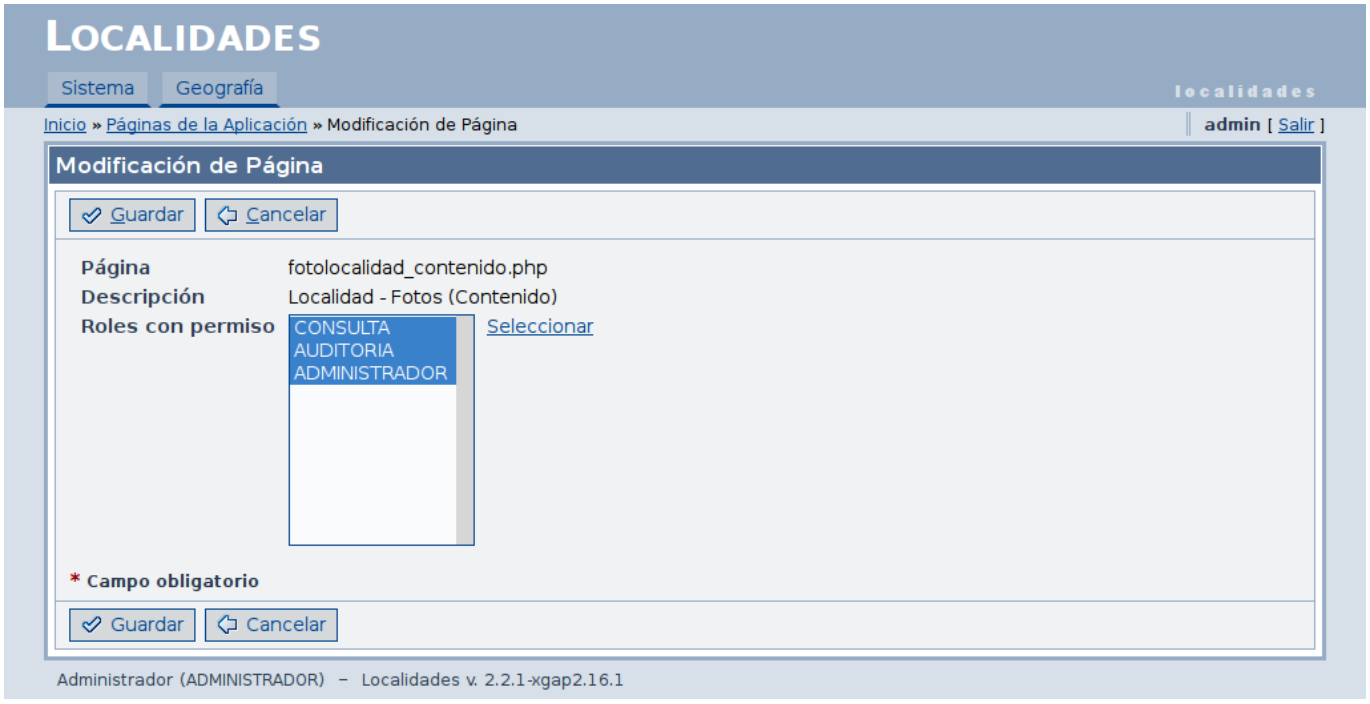


Figura 9.20: Captura de pantalla de paginaapp_formulario.php en la aplicación “Localidades”

Definición de campos

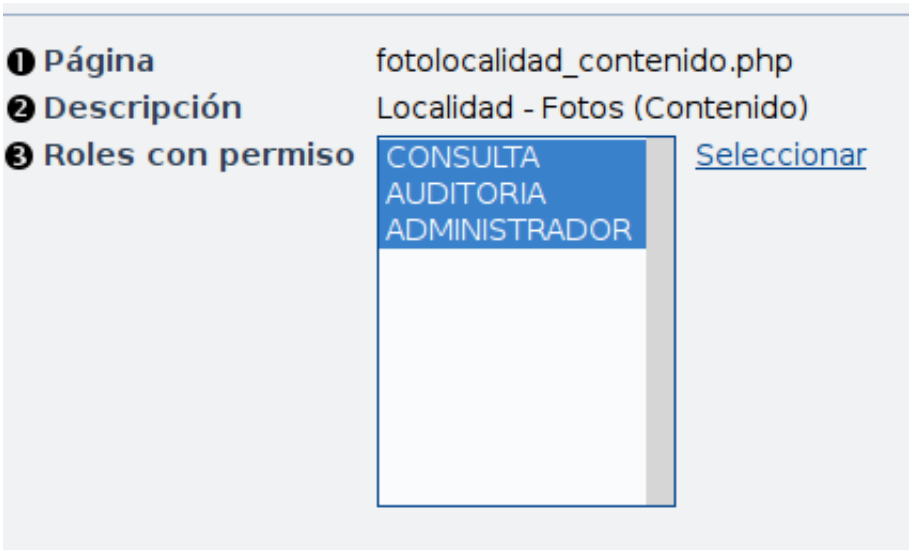
Los campos del formulario están definidos como sigue:

```
<tabla nombre="pagina" esquema="seguridad"/>
<clave>
    <campo>naplicacion</campo>
    <campo>npagina</campo>
</clave>
<campos>
    <campo tipo="Oculto">
        <dato>naplicacion</dato>
    </campo>
    <campo tipo="SoloLectura">                                          ❶
        <titulo>Página</titulo>
        <dato>npagina</dato>
    </campo>
    <campo tipo="SoloLectura">                                          ❷
        <titulo>Descripción</titulo>
        <dato>tdescripcion</dato>
    </campo>
    <campo tipo="SeleccionableMultiple" filas="10">                     ❸
        <titulo>Roles con permiso</titulo>
        <dato>id_rolfs</dato>
        <entidad>rolf</entidad>
        <seleccionar>id_rolf</seleccionar>
        <descripcion>desc_rolf</descripcion>
        <consulta-sm>
            <tabla>permiso_pagina</tabla>
            <valor>id_rolf</valor>
            <where>naplicacion = '$naplicacion' AND npagina = '$npagina'</where>
        </consulta-sm>
    </campo>
</campos>
```

❶ `seguridad.pagina.npagina character(100) NOT NULL`. Se muestra como texto plano por ser de tipo "SoloLectura".

❷ `seguridad.pagina.tdescripcion text`. Se muestra como texto plano por ser de tipo "SoloLectura".

❸ Ver a continuación.



Campo SeleccionableMultiple

Permite seleccionar los roles.

```
        <campo tipo="SeleccionableMultiple" filas="10">
            <titulo>Roles con permiso</titulo>
            <dato>id_rolfs</dato>                ❶
            <entidad>rolf</entidad>              ❷
            <seleccionar>id_rolf</seleccionar>   ❸
            <descripcion>desc_rolf</descripcion> ❹
            <consulta-sm>                        ❺
                <tabla>permiso_pagina</tabla>
                <valor>id_rolf</valor>
                <where>naplicacion = '$naplicacion' AND npagina = '$npagina'</where>
            </consulta-sm>
        </campo>
```

❶ El campo `id_rolfs` no existe en la base de datos; sus valores se obtienen como resultado de la ejecución de `campo/consulta-sm`.

❷ La acción "Seleccionar" del campo abre `rolf_listado_seleccion_m.php`.

❸, ❹, ❺ La consulta SQL definida por `campo/consulta-sm` indica los valores que tiene el campo cuando se carga el formulario. Los valores y claves de los items iniciales se obtienen a partir de una consulta SQL que combina `campo/consulta-sm`, `campo/tabla` o `campo/entidad`, `campo/descripcion` y `campo/seleccionar`: `SELECT id_rolf, desc_rolf FROM rolf WHERE id_rolf IN (SELECT DISTINCT id_rolf FROM permiso_pagina WHERE naplicacion ='$naplicacion' AND npagina ='$npagina')` (en general: `SELECT {campo/seleccionar}, {campo/descripcion} FROM {campo/tabla|campo/entidad} WHERE {campo/seleccionar} IN (SELECT DISTINCT {campo/consulta-sm/valor} FROM {campo/consulta-sm/tabla} WHERE {campo/consulta-sm/where})`).

El campo se genera como un elemento HTML `select` con el atributo `multiple` establecido y nombre `"{campo/dato}[]"`, es decir:

```
<select multiple="multiple" name="id_rolfs[]" ...>
<!-- lista de elementos option -->
</select>
```

XGAP no provee un procesamiento predeterminado para los valores seleccionados en el campo, sino que es responsabilidad de la aplicación utilizarlos de acuerdo a la funcionalidad que se desee proveer. Cuando se envía el formulario y la página recibe la solicitud POST resultante, los valores de `id_rolf` correspondientes a los roles seleccionados quedan disponibles en un arreglo que se puede acceder en diversas ubicaciones de código personalizado, como por ejemplo en la variable `$registro['id_rolfs']` que está disponible en `/formulario/custom_update` (ver a continuación).

Código de actualización personalizado (`/formulario/custom_update`)

Como se mencionó al comienzo de esta sección, la funcionalidad del formulario no consiste en actualizar un único registro de la base de datos, sino las asociaciones entre la página indicada y los roles seleccionados. Esto significa que se debe redefinir el comportamiento predeterminado de la operación de actualización; por otro lado, no es necesario alterar la operación de agregado, porque este formulario no se usa con ese fin.

Para realizar la actualización se hace uso del método `iSeguridad::actualizarPermisosPorPagina($aplicacion, $pagina, $roles_usuario, $operaciones)`, provisto por el objeto `$objeto_seguridad`. El parámetro `$aplicacion` corresponde al valor del campo `naplicacion`, el parámetro `$pagina` al campo `npagina` y el parámetro `$roles_usuario` al array construído con las claves de los roles seleccionados en el campo `id_rolfs`; el parámetro `$operaciones` no se utiliza en este caso.

```
    <custom_update>
        <![CDATA[
        $objeto_seguridad->actualizarPermisosPorPagina(
                $registro['naplicacion'],
                $registro['npagina'],
                $registro['id_rolfs'],
                ''
        );
```

```
        ]]>
    </custom_update>
```

paginaapp_listado.xml

Listado de páginas que componen la aplicación y permisos de acceso por página.

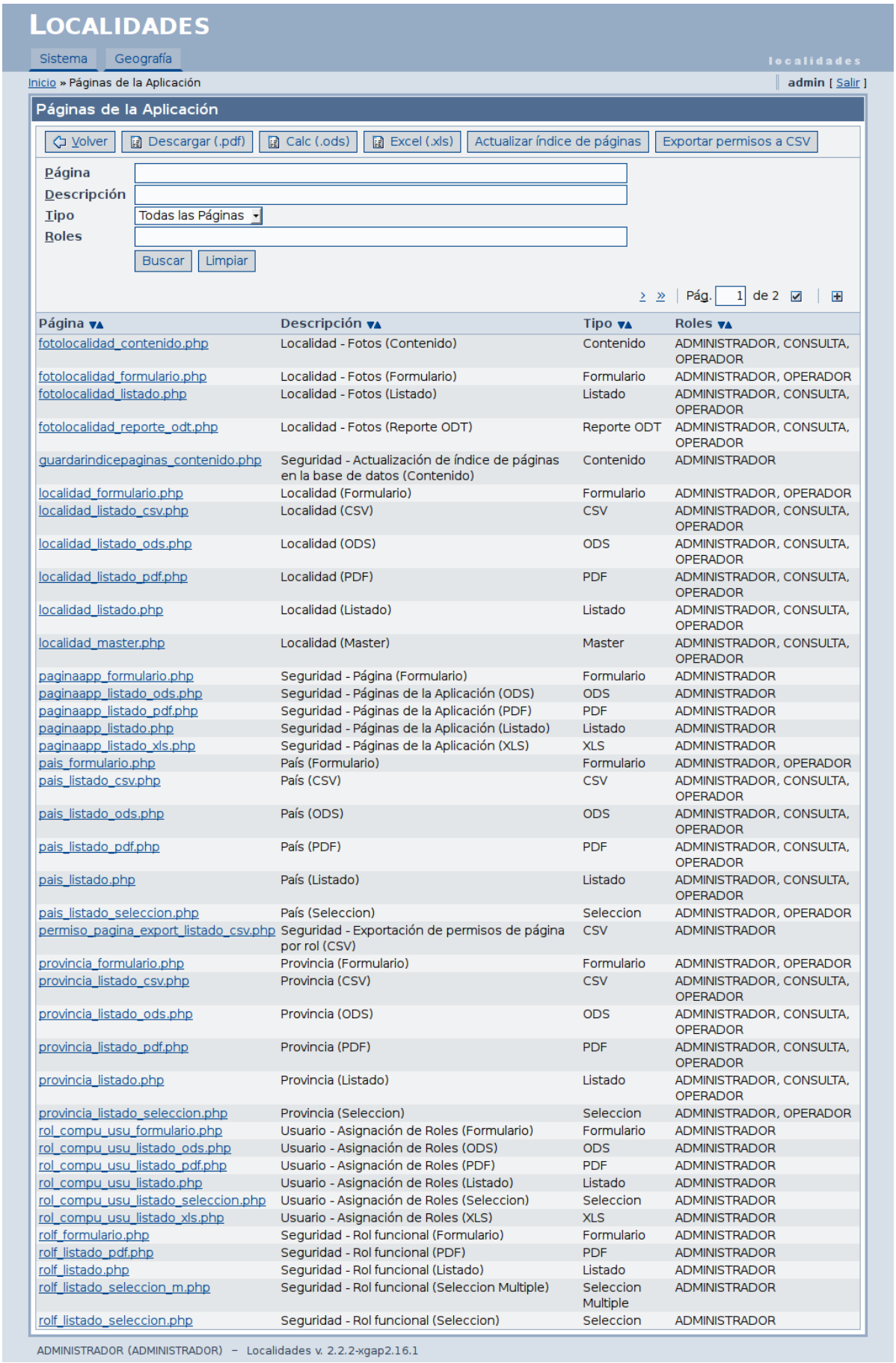


Figura 9.21: Captura de pantalla de paginaapp_listado.php en la aplicación “Localidades”

Definición de columnas

Las columnas del listado están definidas como sigue:

```
<consulta>
    <select><!-- ... --></select>
    <from>seguridad.vpaginarolfs</from>
    <!-- ... -->
</consulta>
<columnas>
    <columna llevaAForm="true">                                          (1)
        <titulo>Página</titulo>
        <dato>npagina</dato>
    </columna>
    <columna>                                                            (2)
        <titulo>Descripción</titulo>
        <dato>tdescripcion</dato>
    </columna>
    <columna>                                                            (3)
        <titulo>Tipo</titulo>
        <dato>ctipo</dato>
        <formato>
            <codigo>
                <![CDATA[
                return isset($GLOBALS['tipos_pag'][$valor])
                        ? $GLOBALS['tipos_pag'][$valor]
                        : ' ';
                ]]>
            </codigo>
        </formato>
    </columna>
    <columna>                                                            (4)
        <titulo>Roles</titulo>
        <tip>Roles con permiso de acceso a la página</tip>
        <dato>desc_rolfs</dato>
    </columna>
</columnas>
```

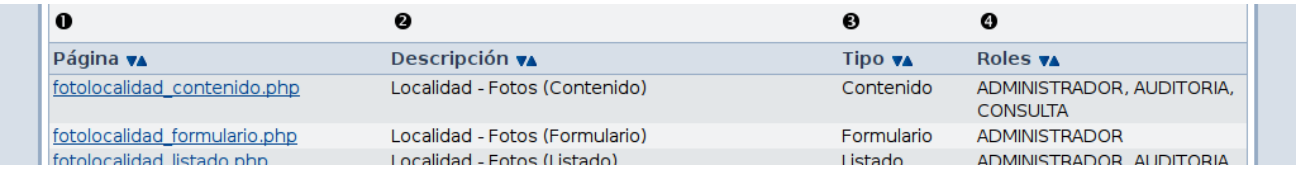
(1) `seguridad.vpaginarolfs.npagina character(100)`.

(2) `seguridad.vpaginarolfs.tdescripcion text`.

(3) `seguridad.vpaginarolfs.ctipo`. Ver a continuación.

(4) `seguridad.vpaginarolfs.desc_rolfs text`. El valor de esta columna se construye en la vista `seguridad.vpaginarolfs`, como la concatenación de los nombres de los roles que tienen permiso de acceso a la página. Su definición es:

```
SELECT ...,
        array_to_string(
            ARRAY(
                SELECT btrim(r.desc_rolf) AS desc_rolf
                    FROM seguridad.permiso_pagina pp
                        JOIN seguridad.rolf r USING (id_rolf)
                    WHERE p.naplicacion = pp.naplicacion AND p.npagina = pp. ↩
                        npagina
                    ORDER BY r.desc_rolf
            ),
            ', ') AS desc_rolfs
    FROM seguridad.pagina p;
```



❶ Página	❷ Descripción	❸ Tipo	❹ Roles
fotolocalidad_contenido.php	Localidad - Fotos (Contenido)	Contenido	ADMINISTRADOR, AUDITORIA, CONSULTA
fotolocalidad_formulario.php	Localidad - Fotos (Formulario)	Formulario	ADMINISTRADOR
fotolocalidad_listado.php	Localidad - Fotos (Listado)	Listado	ADMINISTRADOR, AUDITORIA

Columna con formato personalizado

En la columna `ctipo`, que corresponde al identificador interno de tipo de página, se usa `columna/formato/codigo` para mostrar la descripción del tipo en lugar de dicho identificador. El código de formato utiliza el array `$GLOBALS['tipos_pag']`, construído en otra sección de código, que mapea cada identificador con su descripción.

```
<?php
                    return isset($GLOBALS['tipos_pag'][$valor])
                            ? $GLOBALS['tipos_pag'][$valor]
                            : ' ';
```

Especificación de variedades a generar

Sólo se genera el listado normal y la exportación en PDF, ODS y XLS.

```
        <salidas>
            <normal/>
            <pdf/>
            <ods/>
            <xls/>
        </salidas>
```

Buscador personalizado

El buscador del listado se personaliza para permitir la búsqueda por Página, Descripción, Tipo y/o Roles. Los controles de búsqueda se implementan como campos de texto, excepto el Tipo, que se presenta como una lista desplegable.

```
    <configuracion>
        <!-- ... -->
        <buscador>
            <personalizado-simple>
                <texto nombre="pnpagina" valor="$pnpagina"
                       etiqueta="Página" ancho="60" teclaacceso="p"
                       mayusculas="false"/>                                  ❶
                <texto nombre="ptdescripcion" valor="$ptdescripcion"
                       etiqueta="Descripción" ancho="60" teclaacceso="d"
                       mayusculas="false"/>                                  ❷
                <combo nombre="pctipo" actual="$pctipo"
                       etiqueta="Tipo" teclaacceso="t">                      ❸
                    <variable>tipos_pag</variable>
                </combo>
                <texto nombre="pdesc_rolfs" valor="$pdesc_rolfs"
                       etiqueta="Roles" ancho="60" teclaacceso="r"/>          ❹
            </personalizado-simple>
        </buscador>
        <!-- ... -->
    </configuracion>
    <consulta>
        <!-- ... -->
        <condiciones_de_parametros>
            <and>
                <!-- ... -->
                <like>                                                       ❺
                    <col>npagina</col>
                    <param destacar="si">pnpagina</param>
                </like>
                <like>                                                       ❻
                    <col>tdescripcion</col>
```

```
                <param destacar="si">ptdescripcion</param>
            </like>
            <equal>                                                    ❼
                <col>ctipo</col>
                <param>pctipo</param>
            </equal>
            <like>                                                     ❽
                <col>desc_rolfs</col>
                <param destacar="si">pdesc_rolfs</param>
            </like>
        </and>
    </condiciones_de_parametros>
</consulta>
```

❶, ❷, ❹, ❺, ❻, ❽ Los valores de los campos `pnpagina`, `ptdescripcion` y `pdesc_rolfs` se aplican como filtro a las columnas `npagina`, `tdescripcion` y `desc_rolfs`, respectivamente, utilizando el operador SQL `LIKE`.

❸ Los valores de la lista desplegable están dados por el array `$tipos_pag`, construído en una sección de código extra.

❼ El campo `pctipo` sólo permite la selección de un tipo por vez, por lo cual el filtro consiste en aplicar el operador SQL = entre el valor seleccionado y la columna `ctipo`.

La estructura general de la definición de un buscador personalizado se encuentra más detallada en la documentación de localidad_listado.xml.

Acciones

El listado provee dos acciones:

- "Actualizar índice de páginas" causa que se cargue la página guardarindicepaginas_contenido.php, donde está implementada la actualización del índice de páginas en la base de datos.

```
<configuracion>
    <!-- ... -->
    <acciones>
        <accion nombre="updind">
            <descripcion>Actualizar índice de páginas</descripcion>
            <destino pasarParametros="true">
                <url>guardarindicepaginas_contenido.php</url>
            </destino>
        </accion>
        <!-- ... -->
    </acciones>
</configuracion>
```

- "Exportar permisos a CSV" permite exportar a formato CSV los permisos definidos para las páginas, haciendo una solicitud GET a la variedad CSV de permiso_pagina_export_listado.xml. La solicitud incluye el parámetro `Request Xgap::PARAMETRO_DISPOSICION_CONTENIDO_IMPRIMIBLES` con valor `attachment`, lo que causa que el navegador ofrezca guardar o abrir el contenido de la respuesta, sin salir de la página actual.

```
<configuracion>
    <!-- ... -->
    <acciones>
        <!-- ... -->
        <accion nombre="exportar_permisos_csv">
            <descripcion>Exportar permisos a CSV</descripcion>
            <tip>Emite un archivo CVS que contiene todos los permisos definidos</ ↩
                tip>
            <destino historial="skip" pasarParametros="false" tipo-request="GET">
                <url>permiso_pagina_export_listado_csv.php</url>
                <parametros>
                    <parametro
```

```
                            nombre-expr=" ↩
                                RequestXgap::PARAMETRO_DISPOSICION_CONTENIDO_IMPRIMIBLES ↩
                                "
                            valor="attachment"/>
                    </parametros>
                </destino>
            </accion>
        </acciones>
    </configuracion>
```

Código extra PHP en `xgap_cargado`

Incluye el archivo `clases/PropiedadesSistema.inc.php`, el cual define la clase `PropiedadesSistema`, que se utiliza en otras secciones de código.

```
<?php
require 'clases/PropiedadesSistema.inc.php';
```

Código extra PHP en `antes_consulta`

Define variables que se usan, directa o indirectamente, para filtrar los resultados de la consulta.

```
<?php
$GLOBALS['aplicacion_q'] = "'" // ❶
        . $params['conexion']->qstr(Request::obtener('aplicacion',  ↩
            XGAP_CONF_APLICACION))
        . "'";
$GLOBALS['cte_seguridad'] = iSeguridad::PAGINA_USO_CON; // ❷

Seguridad::obtenerInformacionSeguridad($opciones_pagina,  ↩
    $descripciones_pagina, $sufijos_pagina);
$tipos_pag = array('' => 'Todas las Páginas');
asort($descripciones_pagina, SORT_STRING);
foreach ($descripciones_pagina as $pag => $desc) {
    $tipos_pag[$pag] = $desc;
}
$GLOBALS['tipos_pag'] = $tipos_pag; // ❸
```

❶, ❷ Las variables `$aplicacion_q` y `$cte_seguridad` se usan en `consulta/condiciones_de_parametros` para que sólo se muestren las páginas que pertenecen a la aplicación indicada como parámetro `aplicacion` en la solicitud y que tienen seguridad habilitada (`iSeguridad::PAGINA_USO_CON`), respectivamente.

```
<consulta>
    <!-- ... -->
    <condiciones_de_parametros>
        <and>
            <equal>
                <col>naplicacion</col>
                <constante>$aplicacion_q</constante>
            </equal>
            <equal>
                <col>nseguridad</col>
                <constante>$cte_seguridad</constante>
            </equal>
            <!-- ... -->
```

---

sugerencia

En este caso se decidió utilizar `consulta/condiciones_de_parametros` para incluir en la consulta las dos condiciones mencionadas, pero también se podrían haber definido explícitamente en `consulta/where`.

---

❸ Ya se mencionó la variable `$tipos_pag`, que se usa para definir los items en la lista desplegable del campo `pctipo` en el buscador personalizado y para proveer la descripción del tipo en el formato de la columna `ctipo`.

Código extra PHP en `despues_inicializacion`

Determina si es necesario actualizar el índice de páginas y registra este hecho en la variable booleana `$index_update_needed`.

```
<?php
$conexion = $params['conexion'];
$time_last_index_update = (int) PropiedadesSistema::obtenerValor(
    $conexion,
    PropiedadesSistema::PROP_ULT_ACT_INDICE_PAGS
);
$time_current_index = @filemtime(NOMBRE_INDICE);
$GLOBALS['index_update_needed'] =
        !is_null($time_last_index_update) &&
        $time_current_index !== false &&
        $time_current_index > $time_last_index_update;
```

Código extra PHP en `antes_tabla_datos`

Si es necesario actualizar el índice de páginas, según lo indicado por la variable `$index_update_needed` definida en despues_inicializacion, agrega un mensaje de alerta para informar del hecho al usuario.

```
<?php
if ($GLOBALS['index_update_needed']) {
    Mensaje::alerta('Es necesario actualizar el índice de páginas.');
}
```

---

sugerencia

El agregado del mensaje también se podría haber implementado a través del mecanismo de mensajes de página, con código en `despues_inicializacion`, en lugar de `antes_tabla_datos`:

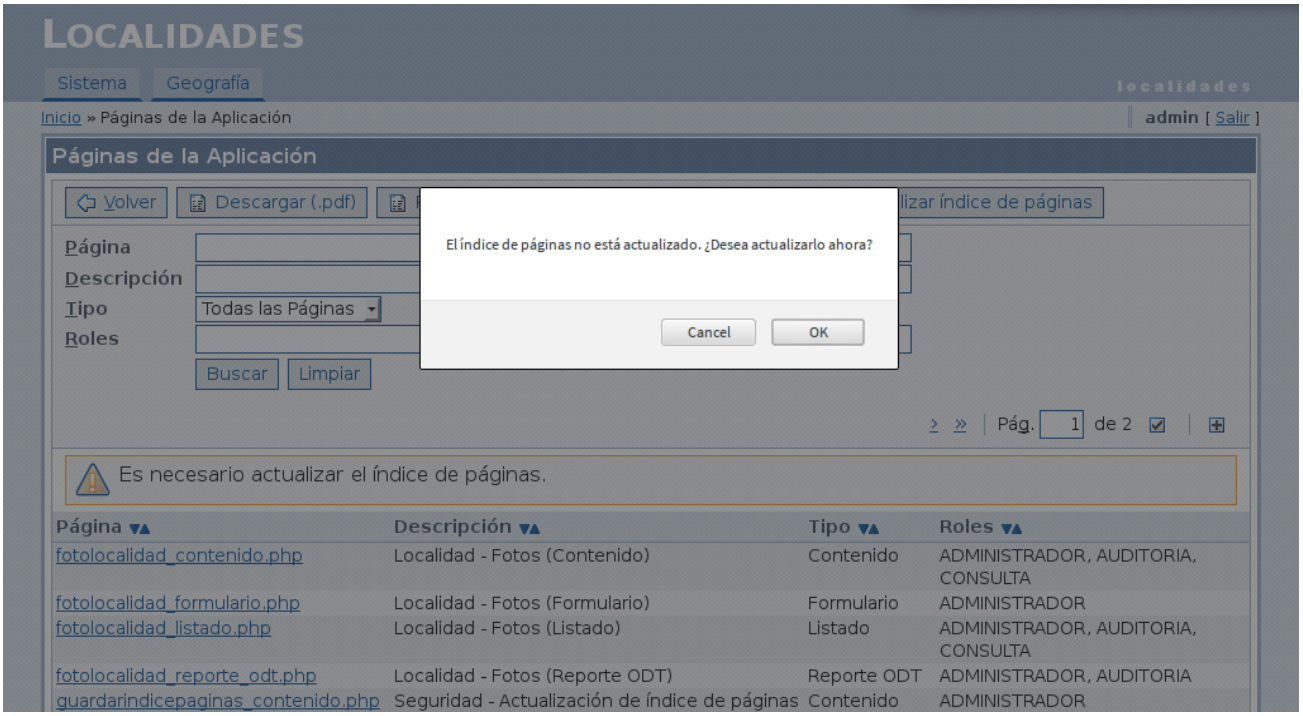
```
<codigo tipo="PHP" ubicacion="despues_inicializacion">
    <![CDATA[
    <?php
    // ...
    global $index_update_needed;
    $index_update_needed = ...
    if ($index_update_needed) {
        Pagina::instancia()->mensajes()->agregar(
            ObjetoMensaje::nuevo(
                'Es necesario actualizar el índice de páginas.',
                Mensaje::TIPO_ALERTA
            )
        );
    }
```

Código extra JAVASCRIPT en `fin_body`

Si es necesario actualizar el índice de páginas, según lo indicado por la variable `$index_update_needed` definida en despues_inicializacion, invoca automáticamente la acción "Actualizar índice de páginas" cuando se carga la página, previa confirmación por parte del usuario.

```
<?php if ($GLOBALS['index_update_needed']) { ?>
jQuery(document).ready(function() {
    var update_now = confirm(
            "El índice de páginas no está actualizado. ¿Desea actualizarlo ahora?"
    );
```

```
                if (update_now) {
                    jQuery('#b_accion_updind').click();
                }
            });
            <?php } ?>
```



pais_formulario.xml

Formulario de alta/baja/modificación de país.

Figura 9.22: Captura de pantalla de pais_formulario.php, presentado para modificación, en la aplicación “Localidades”

Definición de campos

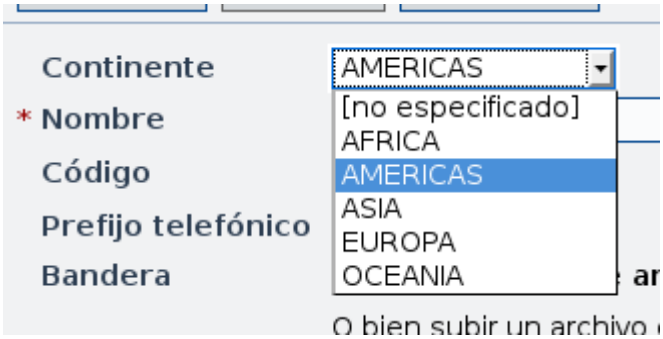
Los campos del formulario están definidos como sigue:

```
<tabla nombre="pais"/>
...
<campos>
    <campo tipo="Oculto">
        <dato>npais</dato>
    </campo>
    <campo tipo="ComboBD">                                                    ❶
        <titulo>Continente</titulo>
        <dato>ncontinente</dato>
        <consulta>
            <tabla>continente</tabla>
            <valor>ncontinente</valor>
            <etiqueta>vnombre</etiqueta>
            <orderby>vnombre</orderby>
        </consulta>
        <opcion-combobd-seleccion-vacia texto="[no especificado]"/>
    </campo>
    <campo ancho="60">                                                        ❷
        <titulo>Nombre</titulo>
        <dato>vnombre</dato>
    </campo>
    <campo>                                                                   ❸
        <titulo>Código</titulo>
        <dato>ccod2</dato>
    </campo>
    <campo>                                                                   ❹
        <titulo>Prefijo telefónico</titulo>
        <dato>cpreftelef</dato>
        <validacion>unsigned</validacion>
    </campo>
    <campo tipo="Archivo">                                                    ❺
```

```
            <dato>vbandera</dato>
            <titulo>Bandera</titulo>
            <tip>Archivo de imagen para la bandera</tip>
            <destino>pais</destino>
        </campo>
        <campo tipo="Archivo">                                                    ❻
            <dato>vhimno</dato>
            <titulo>Himno</titulo>
            <tip>Archivo de audio para el himno</tip>
            <destino>pais</destino>
        </campo>
    </campos>
```

❶ Valor: `pais.ncontinente integer`. Items en la lista desplegable: `continente.ncontinente integer` para los valores; `continente.vnombre character varying(50)` para las etiquetas.

La definición usa `campo/opcion-combobd-seleccion-vacia/@texto` para establecer explícitamente el texto del item en la lista que indica que no se selecciona un valor.



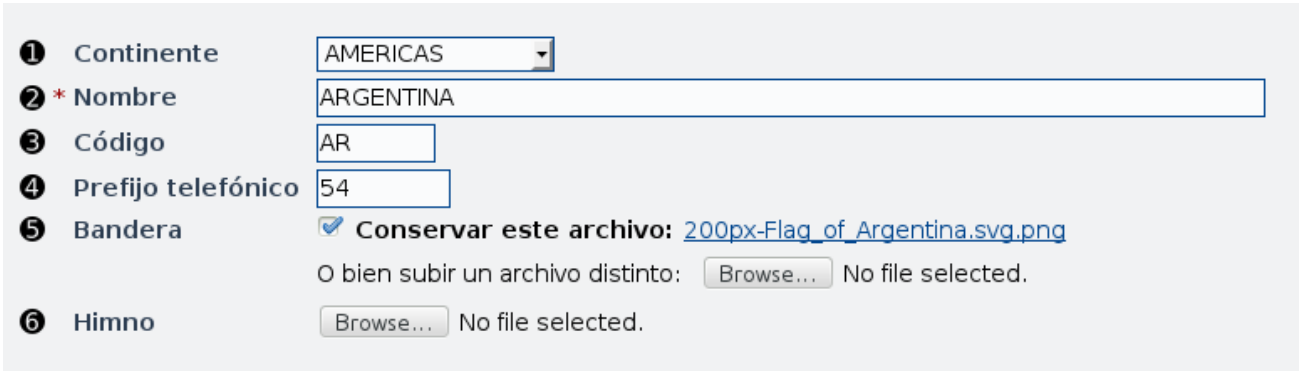
❷ `pais.vnombre character varying(100) NOT NULL`. El ancho del campo de texto se limita a 60 caracteres, en vez de los 100 que tomaría por defecto de acuerdo a la definición de la columna en la base de datos, para que no ocupe demasiado espacio horizontal.

❸ `pais.ccod2 character(2).`

❹ `pais.cpreftelef character(3).`

❺ `pais.vbandera character varying(2048).`

❻ `pais.vhimno character varying(2048)` La ruta del archivo subido, tanto para este campo como para el anterior, se modifica en código extra.



Código extra PHP en `antes_procesar_uploads`

Modifica la ruta de los archivos subidos, concatenándole la clave primaria del país, para que los archivos de cada país queden almacenados en un directorio separado.

```
            <?php
            if (empty2($params['registro']['npais'], false)) {
```

```
            $npais = $params['conexion']->siguienteId('pais_npais_seq');
            $params['registro']['npais'] = $npais;
        } else {
            $npais = $params['registro']['npais'];
        }
        foreach ($params['campos'] as $dato => $info) {
            if ($info['conservar'] === false && isset($info['upload'])) {
                $nuevo_dir_destino = $info['destino'] != ''
                        ? ruta_archivo(array($info['destino'], $npais), DIR_SEP,
                            false)
                        : "pais-$npais";
                $info['upload']->cambiarDirDestino($nuevo_dir_destino, true);
            }
        }
```

pais_listado.xml

Listado de países.

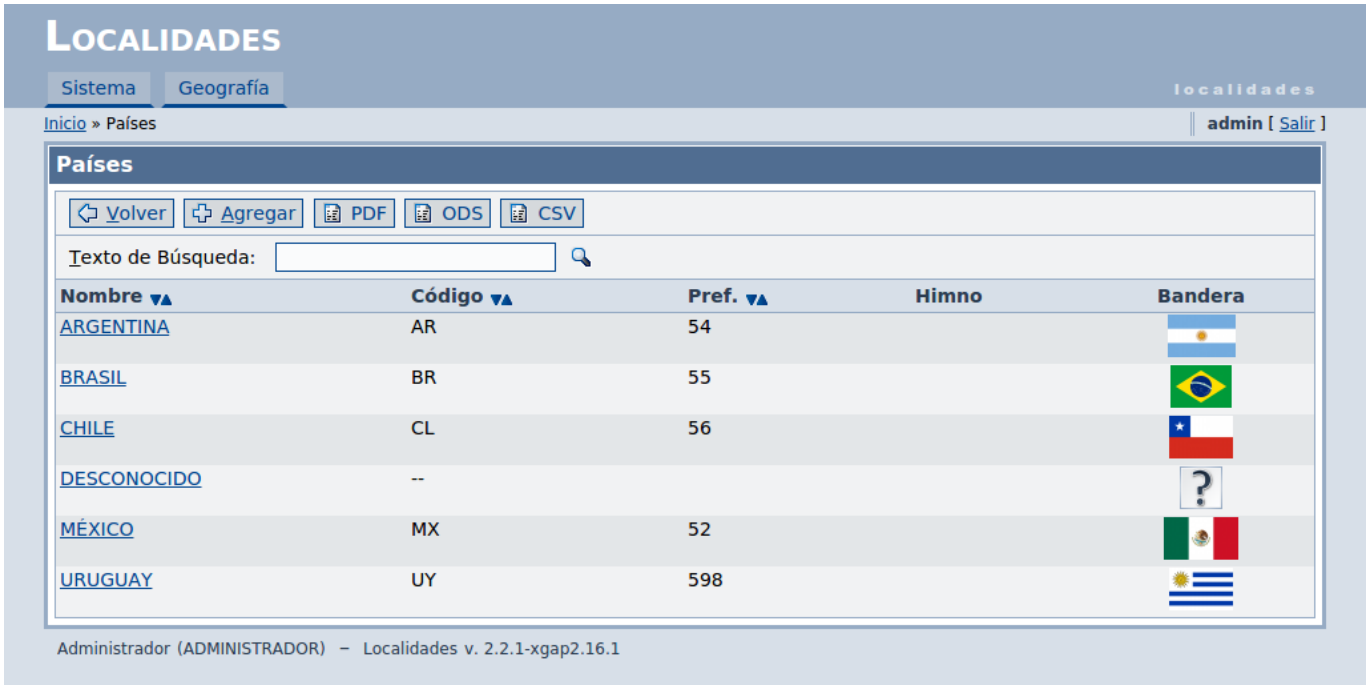


Figura 9.23: Captura de pantalla de pais_listado.php en la aplicación “Localidades”

Definición de columnas

Las columnas del listado están definidas como sigue:

```
<columnas>
    <columna llevaAForm="true">                                    ❶
        <titulo>Nombre</titulo>
        <dato>vnombre</dato>
    </columna>
    <columna>                                                      ❷
        <titulo>Código</titulo>
        <dato>ccod2</dato>
    </columna>
```

```
        <columna>                                                    ❸
            <titulo>Pref.</titulo>
            <tip>Prefijo telefónico</tip>
            <dato>cpreftelef</dato>
        </columna>
        <columna esArchivo="true" permiteOrdenar="false">            ❹
            <dato>vhimno</dato>
            <titulo>Himno</titulo>
        </columna>
        <columna permiteOrdenar="false">                             ❺
            <dato>vbandera</dato>
            <titulo>Bandera</titulo>
            <imagen>
                <miniatura alto="32"/>
            </imagen>
        </columna>
    </columnas>
```

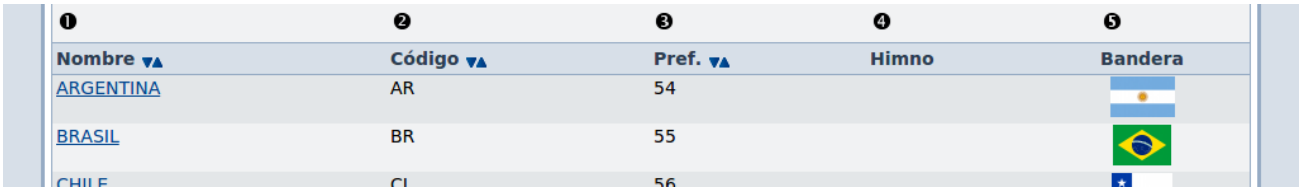
❶ `pais.vnombre character varying(100).`

❷ `pais.ccod2 character(2).`

❸ `pais.cpreftelef character(3).`

❹ `pais.vhimno character varying(2048).`

❺ `pais.vbandera character varying(2048)`. Columna de imágenes, que muestra miniaturas de las banderas, con una altura de 32px.



Especificación de variedades a generar

Se incluye la generación de CSV y se excluye la de XLS.

```
        <salidas>
            <normal/>
            <seleccion/>
            <pdf/>
            <ods/>
            <csv/>
        </salidas>
```

permiso_pagina_export_listado.xml

Listado usado para exportar a formato CSV los permisos de páginas de la aplicación. La generación no incluye ninguna variante HTML, dado que la única función de este listado es proveer exportación de datos.

Definición de consulta y columnas

La consulta obtiene todas las filas de la tabla `seguridad.permiso_pagina` que corresponden a la aplicación actual, mientras que las columnas del listado están definidas como un mapeo directo a las columnas que se quieren exportar de dicha tabla.

```
    <consulta>
        <select>naplicacion, npagina, id_rolf</select>
        <from>seguridad.permiso_pagina</from>
        <where>naplicacion = '{$GLOBALS['naplicacion']}'</where>     ❶
```

```
        <order_by>npagina, id_rolf</order_by>
    </consulta>
    <!-- ... -->
    <columnas>
        <columna>
            <titulo>naplicacion</titulo>
            <dato>naplicacion</dato>
        </columna>
        <columna>
            <titulo>npagina</titulo>
            <dato>npagina</dato>
        </columna>
        <columna>
            <titulo>id_rolf</titulo>
            <dato>id_rolf</dato>
        </columna>
    </columnas>
```

❶ La variable `$GLOBALS['naplicacion']` se define en código extra.

Especificación de variedades a generar

La salida de este listado está limitada a la variedad CSV.

```
        <salidas>
            <csv/>
        </salidas>
```

Código extra PHP en `antes_consulta`

Define la variable global `$naplicacion`, que se usa para filtrar la consulta del listado.

```
            <?php
            $GLOBALS['naplicacion'] = XGAP_CONF_APLICACION;
```

Código extra PHP en `despues_inicializacion`

Establece un valor personalizado para el nombre de archivo que se va a sugerir al usuario, formado con el nombre de la aplicación y la fecha y hora de generación.

```
            <?php
            $params['nombre_archivo'] =
                    XGAP_CONF_APLICACION . '-permisos_pagina-' . date('YmdHis');
```

provincia_formulario.xml

Formulario de alta/baja/modificación de provincia.

Figura 9.24: Captura de pantalla de provincia_formulario.php, presentado para modificación, en la aplicación “Localidades”

Definición de campos

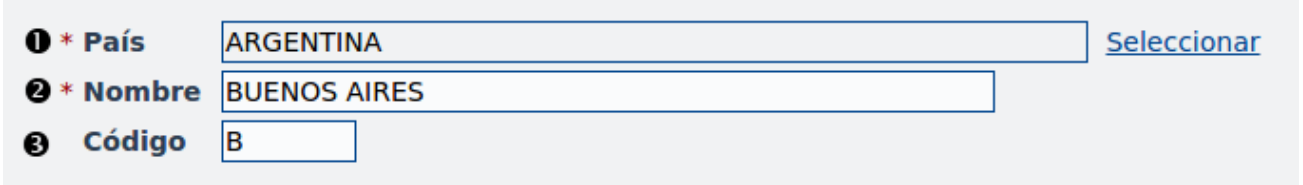
Los campos del formulario están definidos como sigue:

```
<tabla nombre="provincia"/>
...
<campos>
    <campo tipo="Seleccionable" ancho="60">                    ❶
        <titulo>País</titulo>
        <dato>npais</dato>
        <entidad>pais</entidad>
        <seleccionar>npais</seleccionar>
        <descripcion>vnombre</descripcion>
    </campo>
    <campo>                                                    ❷
        <titulo>Nombre</titulo>
        <dato>vnombre</dato>
    </campo>
    <campo>                                                    ❸
        <titulo>Código</titulo>
        <dato>ccodigo</dato>
    </campo>
</campos>
```

❶ provincia.npais integer NOT NULL.

❷ provincia.vnombre character varying(50) NOT NULL.

❸ provincia.ccodigo character(2).



provincia_listado.xml

Listado de provincias.

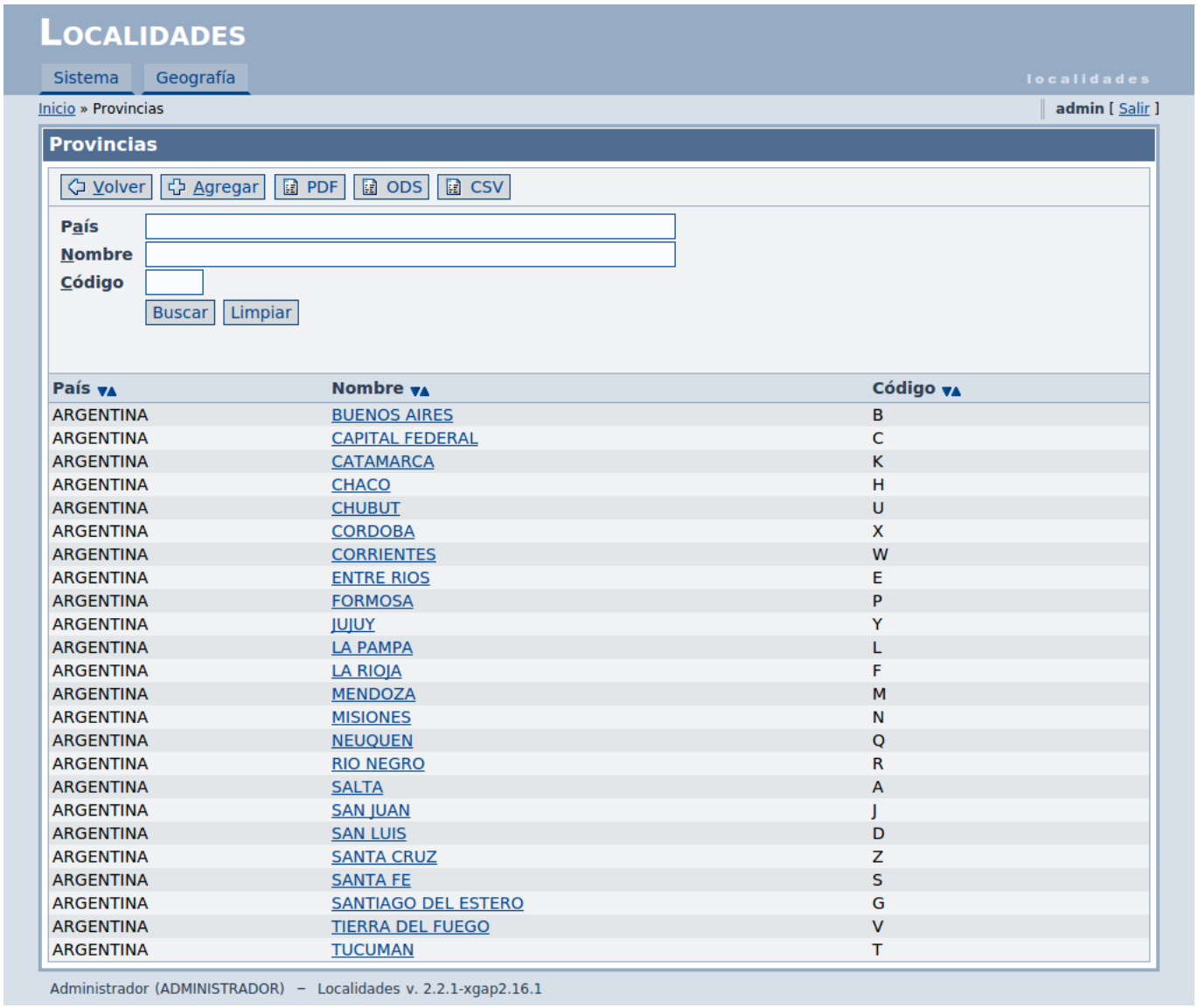


Figura 9.25: Captura de pantalla de provincia_listado.php en la aplicación “Localidades”

Definición de columnas

Las columnas del listado están definidas como sigue:

```
<columnas>
    <columna>                                                    ❶
        <titulo>País</titulo>
        <dato>vnombrepais</dato>
    </columna>
    <columna llevaAForm="true">                                  ❷
        <titulo>Nombre</titulo>
        <dato>vnombre</dato>
    </columna>
    <columna>                                                    ❸
        <titulo>Código</titulo>
        <dato>ccodigo</dato>
    </columna>
</columnas>
```

1. `vprovincia.character varying(100).`
2. `vprovincia.character varying(50).`
3. `vprovincia.ccodigo character(2).`

❶	❷	❸
País	Nombre	Código
ARGENTINA	BUENOS AIRES	B
ARGENTINA	CAPITAL FEDERAL	C

Especificación de variedades a generar

Se incluye la generación de CSV y se excluye la de XLS.

```
        <salidas>
            <normal/>
            <seleccion/>
            <pdf/>
            <ods/>
            <csv/>
        </salidas>
```

Buscador personalizado

El buscador del listado se personaliza para permitir la búsqueda en cada una de las columnas por separado.

```
<!-- ... -->
    <configuracion>
        <!-- ... -->
        <buscador>
            <personalizado-simple>
                <texto nombre="pvnombrepais" valor="$pvnombrepais"
                        etiqueta="País" ancho="60" teclaacceso="a"/>
                <texto nombre="pvnombre" valor="$pvnombre"
                        etiqueta="Nombre" ancho="60" teclaacceso="n"/>
                <texto nombre="pccodigo" valor="$pccodigo"
                        etiqueta="Código" ancho="2" teclaacceso="c"/>
            </personalizado-simple>
        </buscador>
        <!-- ... -->
    </configuracion>
    <consulta>
        <!-- ... -->
        <condiciones_de_parametros>
            <and>
                <like>
                    <col>vnombrepais</col>
                    <param destacar="si">pvnombrepais</param>
                </like>
                <like>
                    <col>vnombre</col>
                    <param destacar="si">pvnombre</param>
                </like>
                <equal>
                    <col>ccodigo</col>
                    <param>pccodigo</param>
                </equal>
            </and>
        </condiciones_de_parametros>
    </consulta>
<!-- ... -->
```

Los valores de los campos `pvnombrepais` y `pvnombre` se aplican como filtro a las columnas `vnombrepais` y `vnombre`, respectivamente, utilizando el operador SQL `LIKE`, en tanto que el valor del campo `pccodigo` se compara con la columna `ccodigo` mediante el operador SQL =.

La estructura general de la definición de un buscador personalizado se encuentra más detallada en la documentación de localidad_listado.xml.

rol_compu_usu_formulario.xml

Formulario de modificación de roles funcionales asignados a un usuario.

Figura 9.26: Captura de pantalla de rol_compu_usu_formulario.php en la aplicación “Localidades”

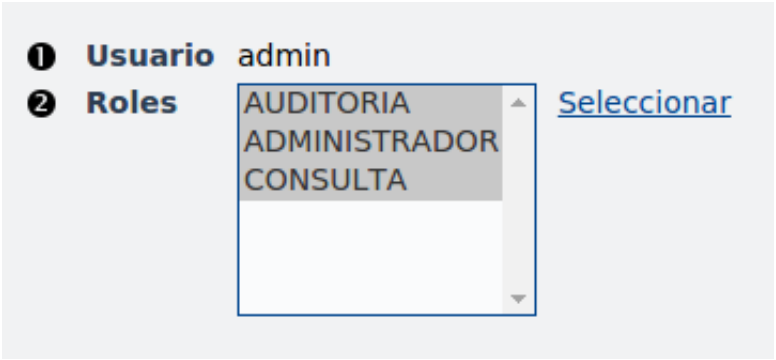
Definición de campos

Los campos del formulario están definidos como sigue:

```
<tabla nombre="vrol_compu_usu" esquema="seguridad"/>
...
<campos>
    <campo tipo="Oculto">
        <dato>id_rolh</dato>
    </campo>
    <campo tipo="Oculto">
        <dato>id_rolv</dato>
    </campo>
    <campo tipo="SoloLectura">                                    ❶
        <dato>id_usuario</dato>
        <titulo>Usuario</titulo>
        <consulta>
            <tabla>usuario</tabla>
            <valor>id_usuario</valor>
            <etiqueta>nombre</etiqueta>
        </consulta>
    </campo>
    <campo tipo="SeleccionableMultiple" filas="6">                ❷
        <dato>id_rolfs</dato>
        <titulo>Roles</titulo>
```

```
            <entidad>rolf</entidad>
            <seleccionar>id_rolf</seleccionar>
            <descripcion>desc_rolf</descripcion>
            <etiqueta>Seleccionar</etiqueta>
            <consulta-sm>
                <tabla>rol_compu_usu</tabla>
                <valor>id_rolf</valor>
                <where>id_usuario = $id_usuario</where>
            </consulta-sm>
        </campo>
    </campos>
```

1. Campo `SoloLectura` que muestra el valor de `seguridad.usuario.nombre` correspondiente a `seguridad.usuario.id_usuario =seguridad.vrol_compu_usu.id_usuario`, por el uso de `campo/consulta`.
2. Campo `SeleccionableMultiple` que permite elegir los roles del usuario. Los valores seleccionados se procesan en las operaciones personalizadas del formulario.
   La documentación del `SeleccionableMultiple` en paginaapp_formulario.xml contiene una explicación más detallada sobre el funcionamiento de este tipo de campo.



Código extra PHP en `inicio_pagina`

Define una función que se usa en las operaciones personalizadas, para evitar la duplicación de código.

```
            <?php
            function extender_registro(&$registro, $id_rolfs) {
                $registro['roles'] = !empty($id_rolfs) ? implode("|", $id_rolfs) : null ↩
                    ;
            }
```

Código personalizado para agregado y actualización (`/formulario/custom_insert` y `/formulario/custom_update`)

Los dos bloques de código son similares: establecen el campo `roles` del registro, con la concatenación de los IDs de los roles seleccionados, e invocan a `Conexion::AutoExecute()` para realizar la operación correspondiente.

```
    <custom_insert>
        <![CDATA[
        extender_registro($registro, $id_rolfs);
        $objeto_conexion->AutoExecute(
            'seguridad.vrol_compu_usu',
            $registro,
            'INSERT'
        );
        ]]>
    </custom_insert>
    <custom_update>
        <![CDATA[
```

```
        extender_registro($registro, $id_rolfs);
        $objeto_conexion->AutoExecute(
            'seguridad.vrol_compu_usu',
            $registro,
            UPDATE',
            $xclave
        );
        ]]>
    </custom_update>
```

El proceso se completa en la base de datos, a través de dos reglas asociadas a la vista `seguridad.vrol_compu_usu`, las cuales se encargan de registrar las asociaciones de acuerdo a los valores del campo `roles` del registro.

```
CREATE FUNCTION seguridad.fins_rol_compu_usu(
        pid_usuario bigint,
        pid_rolh integer,
        pid_rolv integer,
        proles character)
  RETURNS void AS
$BODY$
    DECLARE
        i integer;
        rol_f bpchar;
        rol_f_int integer;
    BEGIN

    i := 1;

    rol_f := split_part(proles, '|', i);
    WHILE ((NOT rol_f IS NULL) AND (rol_f <> '')) LOOP
        rol_f_int = CAST(rol_f AS INTEGER);
        INSERT INTO rol_compu_usu(id_rolh, id_rolv, id_rolf, id_usuario)
        VALUES (pid_rolh, pid_rolv, rol_f_int, pid_usuario);

        i := i + 1;
        rol_f := split_part(proles, '|', i);
    END LOOP;

    END;
$BODY$
  LANGUAGE 'plpgsql' VOLATILE;

CREATE RULE ri_vrol_compu_usu AS
    ON INSERT TO seguridad.vrol_compu_usu DO INSTEAD
        SELECT seguridad.fins_rol_compu_usu(
            new.id_usuario::bigint,
            new.id_rolh,
            new.id_rolv,
            new.roles::bpchar
        ) AS fins_rol_compu_usu;

CREATE RULE ru_vrol_compu_usu AS
    ON UPDATE TO seguridad.vrol_compu_usu DO INSTEAD (
        DELETE FROM seguridad.rol_compu_usu
            WHERE rol_compu_usu.id_usuario = old.id_usuario;
        SELECT seguridad.fins_rol_compu_usu(
            new.id_usuario::bigint,
            new.id_rolh,
            new.id_rolv,
            new.roles::bpchar
        ) AS fins_rol_compu_usu;
    );
```

rol_compu_usu_listado.xml

Listado de asignaciones de roles funcionales a usuarios.

Figura 9.27: Captura de pantalla de rol_compu_usu_listado.php en la aplicación “Localidades”

Definición de columnas

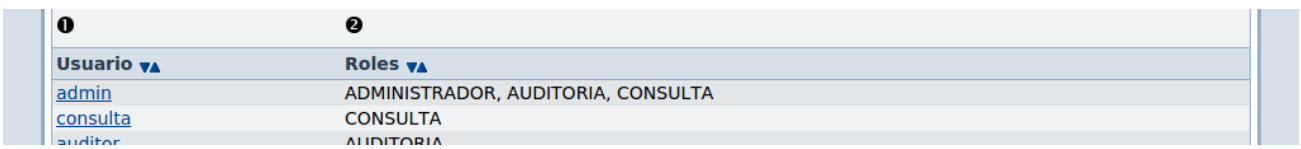
Las columnas del listado están definidas como sigue:

```
<columnas>
    <columna llevaAForm="true">                    ❶
        <titulo>Usuario</titulo>
        <dato>nombre</dato>
    </columna>
    <columna>                                      ❷
        <titulo>Roles</titulo>
        <dato>desc_rolfs</dato>
    </columna>
</columnas>
```

❶ `seguridad.vrol_compu_usu.nombre character(10)`.

❷ `seguridad.vrol_compu_usu.desc_rolfs text`. Esta columna muestra la lista de todos los roles asignados al usuario. Su definición en la vista es la siguiente:

```
SELECT
    -- ...
    array_to_string(
        ARRAY(SELECT btrim(rc.desc_rol_comp) AS desc_rol_comp_trim
            FROM seguridad.rol_compu_usu rcu1
            JOIN seguridad.rol_compuesto rc ON rcu1.id_rolv = rc.id_rolv
                AND rcu1.id_rolh = rc.id_rolh AND rcu1.id_rolf = rc.id_rolf
            WHERE rcu1.id_usuario = u.id_usuario
            ORDER BY rc.desc_rol_comp),
        ', '
    ) AS desc_rolfs
    -- ...
```



rolf_formulario.xml

Formulario de alta y modificación de roles funcionales.

Figura 9.28: Captura de pantalla de rolf_formulario.php en la aplicación “Localidades”

Definición de campos

Los campos del formulario están definidos como sigue:

```
<tabla nombre="rolf" esquema="seguridad"/>
...
<campos>
    <campo>                                                    (1)
        <titulo>Descripción del rol</titulo>
        <dato>desc_rolf</dato>
    </campo>
    <campo ancho="60">                                         (2)
        <titulo>Página de inicio</titulo>
        <dato>pagina_inicio</dato>
    </campo>
</campos>
```

(1) `seguridad.rolf.desc_rolf character varying(30) NOT NULL.`

(2) `seguridad.rolf.pagina_inicio character varying(1024).`

rolf_listado.xml

Listado de roles funcionales.

Figura 9.29: Captura de pantalla de rolf_listado.php en la aplicación “Localidades”

Definición de columnas

Las columnas del listado están definidas como sigue:

```
<columnas>
    <columna llevaAForm="true">                    (1)
        <titulo>Descripción</titulo>
        <dato>desc_rolf</dato>
    </columna>
    <columna>                                      (2)
        <titulo>Página de inicio</titulo>
        <dato>pagina_inicio</dato>
    </columna>
</columnas>
```

(1) `seguridad.rolf.desc_rolf character varying(30).`

(2) `seguridad.rolf.pagina_inicio character varying(1024).`

Especificación de variedades a generar

En este listado se debe incluir la generación de la variedad `seleccion_m`, que se utiliza en paginaapp_formulario.xml y rol_compu_usu_formulario.xml.

```
<salidas>
    <normal/>
    <seleccion/>
    <seleccion_m/>
    <pdf/>
</salidas>
```

seguridad_contenido.xml

Página para establecer los permisos de página por rol funcional.

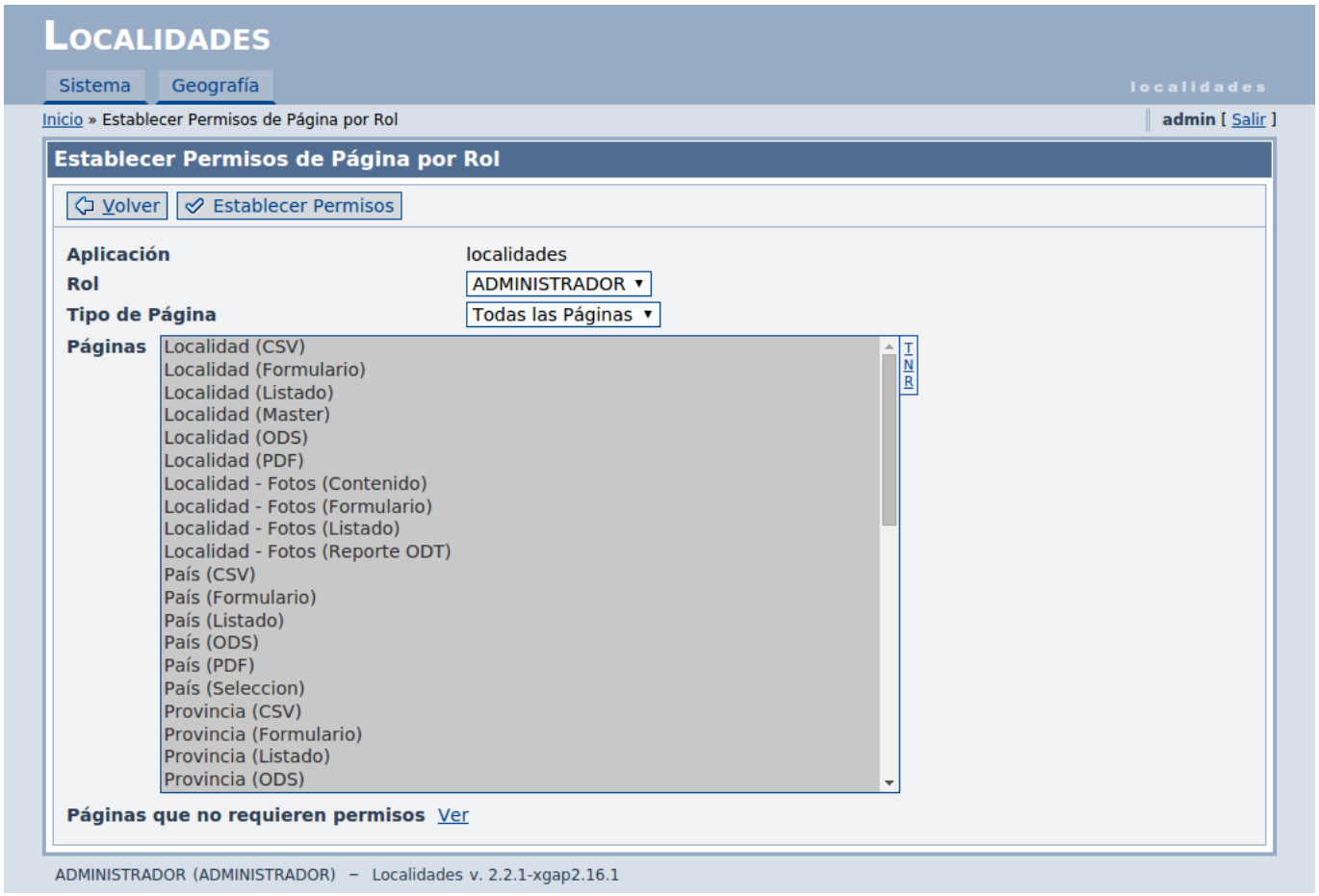


Figura 9.30: Captura de pantalla de seguridad_contenido.php en la aplicación “Localidades”

Comandos

El comando “Establecer Permisos” realiza la actualización de los permisos seleccionados por el usuario.

```
<configuracion>
    <!-- ... -->
    <comandos-genericos>
        <comando nombre="b_aplicar"
                 texto="Establecer Permisos"
                 clase="ButtonGuardar"
                 codigo="enviar_accion(&quot;accion&quot;, &quot;actualizar&quot;,  ↩
                     &quot;form_seguridad&quot;);"> ❶
            <condicion>  ↩

                <![CDATA[
                global $errores;
                return count($errores) == 0;
                ]]>
            </condicion>
            <preparar>  ↩

                <![CDATA[
                global $hay_rol;
                $habilitado = $hay_rol;
                ]]>
            </preparar>
```

```
            </comando>
        </comandos-genericos>
    </configuracion>
```

1. El código del comando realiza el envío del formulario `form_seguridad`, causando una solicitud POST a la misma página, que se procesa en el código incluido en `pagina/contenido-anterior`.
2. El código en el elemento `comando/condicion` hace que el comando no se muestre si hubo algún error durante el procesamiento de la solicitud. El valor de la variable `$errores` se establece en el código incluido en `pagina/contenido-anterior`.
3. El código en el elemento `comando/preparar` deshabilita el comando si no hay un rol seleccionado. El valor de la variable `$hay_rol` se establece en el código incluido en `pagina/contenido-anterior`.

usuario_cambioclave_formulario.xml

Formulario de cambio de contraseña del usuario actual.

Figura 9.31: Captura de pantalla de usuario_cambioclave_formulario.php en la aplicación “Localidades”

Definición de campos

Los campos del formulario están definidos como sigue:

```
    <tabla nombre="usuario" esquema="seguridad"/>
    ...
    <campos>
        <campo tipo="Sesion">
            <dato>id_usuario</dato>
            <valor>ident_usuario</valor>
        </campo>
        <campo tipo="SoloLectura">                                    (1)
            <titulo>Nombre Corto</titulo>
            <dato>nombre</dato>
        </campo>
        <campo tipo="SoloLectura">                                    (2)
            <titulo>Nombre Completo</titulo>
            <dato>nombre_ext</dato>
        </campo>
        <!-- Los dos campos siguientes están definidos con elementos xi:include,
```

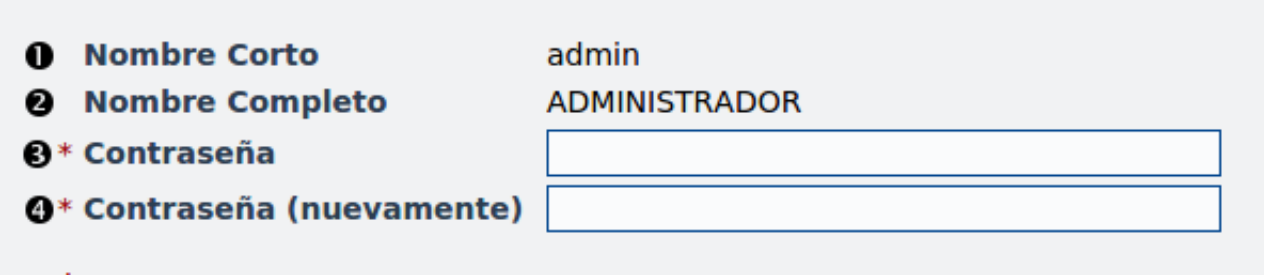
```
            en el código original. Se transcribe el código incluído, para
            facilitar la lectura. -->
        <campo tipo="Clave" ancho="40" requerido="true">                    ❸
            <titulo>Contraseña</titulo>
            <dato>contrasenia</dato>
        </campo>
        <campo tipo="Clave" ancho="40" requerido="true">                    ❹
            <titulo>Contraseña (nuevamente)</titulo>
            <dato>contrasenia2</dato>
        </campo>
    </campos>
```

❶ `seguridad.usuario.nombre character(10).`

❷ `seguridad.usuario.nombre_ext character varying(30)`

❸ `seguridad.usuario.contrasenia character varying(40)`

❹ No corresponde a una columna en la base de datos. Sólo se utiliza para confirmar la contraseña ingresada.



Código extra PHP en `despues_inicializacion`

Asegura que el formulario no se pueda presentar en modo de agregado.

```
            <?php
            // Este formulario no se debe usar para agregados.
            if ($params['agregado']) {
                Http::redirigirAError(NULL, Mensaje::COD_ERR_EJEC_ACCION_DESHABILITADA) ↩
                    ;
            }
```

Código extra JavaScript en `fin_body`

Definido en usuario_formulario.xml.

Uso de XInclude

Este formulario incluye desde `usuario_formulario.xml` los elementos que ambos tienen en común, mediante XInclude, para minimizar la repetición de código.

```
<formulario
            xmlns:xi="http://www.w3.org/2001/XInclude"
            >
    <!-- ... -->
    <xi:include href="usuario_formulario.xml"
                xpointer="xpointer(/formulario/configuracion)"/>
    <xi:include href="usuario_formulario.xml"
                xpointer="xpointer(/formulario/tabla)"/>
    <xi:include href="usuario_formulario.xml"
                xpointer="xpointer(/formulario/clave)"/>
    <campos>
        <!-- ... -->
        <xi:include href="usuario_formulario.xml"
                    xpointer="xpointer(/formulario/campos/campo[dato='contrasenia'])"/>
        <xi:include href="usuario_formulario.xml"
```

```
                    xpointer="xpointer(/formulario/campos/campo[dato='contrasenia2'])"/ ↩
                        >
    </campos>
    <codigoExtra>
        <!-- ... -->
        <xi:include href="usuario_formulario.xml"
                    xpointer="xpointer(/formulario/codigoExtra/codigo[@tipo='JAVASCRIPT ↩
                        ' and @ubicacion='fin_body'])"/>
    </codigoExtra>
</formulario>
```

usuario_cambioclaveadmin_formulario.xml

Formulario de cambio de contraseña de un usuario.

Este formulario es muy similar a usuario_cambioclave_formulario.xml. La única diferencia sustancial entre ambos es que `usuario_cambioclave_formulario.xml` define un campo de tipo `Sesion` para establecer el valor del identificador de usuario (y por lo tanto siempre corresponde al usuario actual), en tanto que `usuario_cambioclaveadmin_formulario.xml` recibe este dato a través de un parámetro de la solicitud.

Uso de XInclude

Para aprovechar la similitud mencionada, la mayor parte del código de `usuario_cambioclaveadmin_formulario.xml` se obtiene de `usuario_formulario.xml` y `usuario_cambioclave_formulario.xml` por medio de elementos XInclude:

```
<formulario
            xmlns:xi="http://www.w3.org/2001/XInclude"
            >
    <!-- ... -->
    <xi:include href="usuario_formulario.xml"
                xpointer="xpointer(/formulario/configuracion)"/>
    <xi:include href="usuario_formulario.xml"
                xpointer="xpointer(/formulario/tabla)"/>
    <xi:include href="usuario_formulario.xml"
                xpointer="xpointer(/formulario/clave)"/>
    <campos>
        <xi:include href="usuario_cambioclave_formulario.xml"
                    xpointer="xpointer(/formulario/campos/campo[dato='nombre'])"/>
        <xi:include href="usuario_cambioclave_formulario.xml"
                    xpointer="xpointer(/formulario/campos/campo[dato='nombre_ext'])"/>
        <xi:include href="usuario_formulario.xml"
                    xpointer="xpointer(/formulario/campos/campo[dato='contrasenia'])"/>
        <xi:include href="usuario_formulario.xml"
                    xpointer="xpointer(/formulario/campos/campo[dato='contrasenia2'])"/ ↩
                        >
    </campos>
    <codigoExtra>
        <xi:include href="usuario_cambioclave_formulario.xml"
                    xpointer="xpointer(/formulario/codigoExtra/codigo[@tipo='PHP' and  ↩
                        @ubicacion='despues_inicializacion'])"/>
        <xi:include href="usuario_formulario.xml"
                    xpointer="xpointer(/formulario/codigoExtra/codigo[@tipo='JAVASCRIPT ↩
                        ' and @ubicacion='fin_body'])"/>
    </codigoExtra>
</formulario>
```

Código extra PHP en `despues_inicializacion`

Definido en usuario_cambioclave_formulario.xml.

Código extra JavaScript en `fin_body`

Definido en usuario_formulario.xml.

usuario_formulario.xml

Formulario de alta y modificación de usuario.

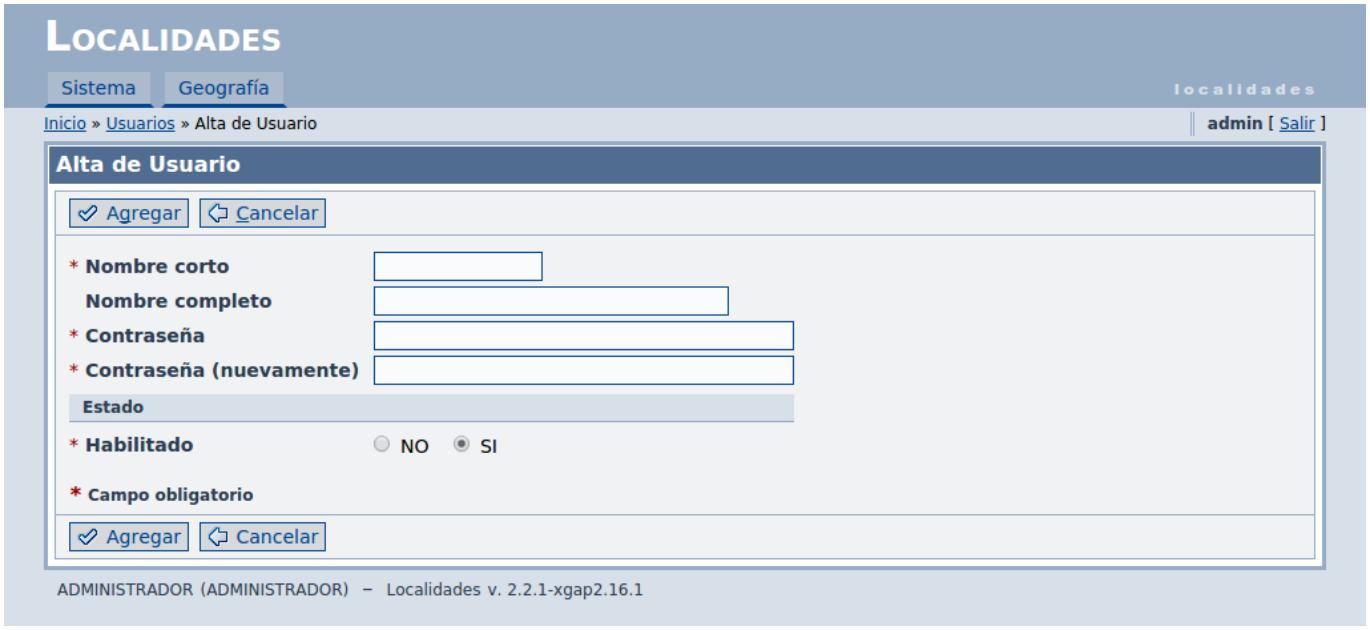


Figura 9.32: Captura de pantalla de usuario_formulario.php en la aplicación “Localidades”

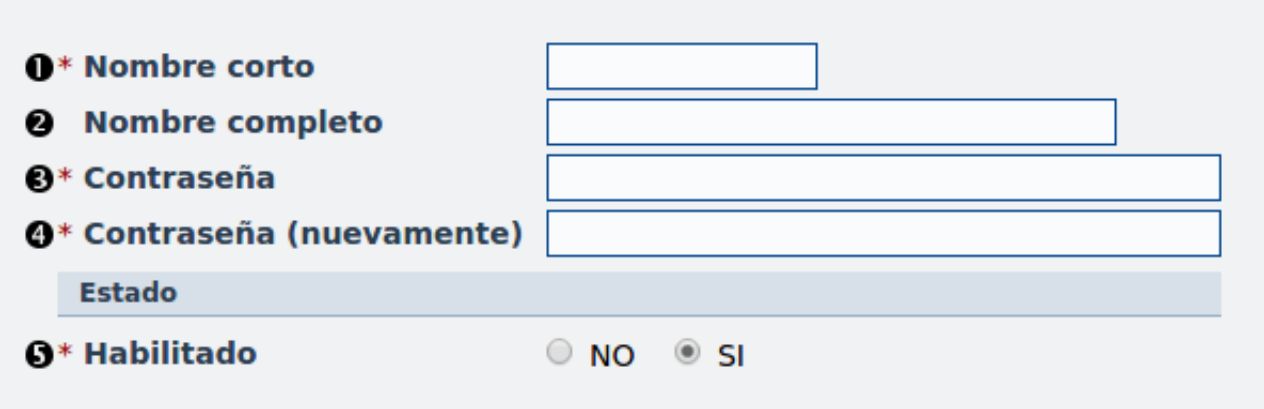
Definición de campos

Los campos del formulario están definidos como sigue:

```
<tabla nombre="usuario" esquema="seguridad"/>
...
<campos>
    <campo mayusculas="false">                                          ❶
        <titulo>Nombre corto</titulo>
        <dato>nombre</dato>
        <tip>Nombre para el ingreso al sistema. Sólo debe usar letras, números o ↩
            guiones bajos.</tip>
        <validacion>alphanum</validacion>
    </campo>
    <campo>                                                             ❷
        <titulo>Nombre completo</titulo>
        <dato>nombre_ext</dato>
        <tip>Apellido y nombre completos</tip>
    </campo>
    <campo tipo="Clave" ancho="40" requerido="true">                    ❸
        <titulo>Contraseña</titulo>
        <dato>contrasenia</dato>
    </campo>
    <campo tipo="Clave" ancho="40" requerido="true">                    ❹
        <titulo>Contraseña (nuevamente)</titulo>
        <dato>contrasenia2</dato>
    </campo>
    <separador_grupo titulo="Estado" />
    <campo tipo="RadioBD">                                              ❺
        <titulo>Habilitado</titulo>
        <dato>habilitado</dato>
        <consulta>
```

```
                <tabla>sistema.traduccionboolean</tabla>
                <valor>bvalor</valor>
                <etiqueta>vvalortraducido</etiqueta>
                <where>vnombre = 'SINO'</where>
            </consulta>
        </campo>
    </campos>
```

1. `seguridad.usuario.nombre character(10).`
2. `seguridad.usuario.nombre_ext character varying(30)`
3. `seguridad.usuario.contrasenia character varying(40)`
4. No corresponde a una columna en la base de datos. Sólo se utiliza para confirmar la contraseña ingresada.
5. Valor: `seguridad.usuario.habilitado boolean NOT NULL`. Opciones: `sistema.traduccionboolean.bvalor boolean` para los valores; `sistema.traduccionboolean.vvalortraducido character varying(50)` para las etiquetas.



Código extra PHP en `xgap_cargado`

Establece un valor por defecto para el usuario, cuando el formulario se presenta en modo de modificación. Ver comentario en el código.

```
<?php
$agregado = param2boolean(Request::obtener(RequestXgap::PARAMETRO_FORM_ALTA ↩
    , false));
if (!$agregado && !Request::existe('id_usuario')
        && ($id_usuario = Contexto::obtener('ident_usuario', '')) != '') {
    /*
     * Si es un formulario de modificación y no se indica el id_usuario en  ↩
        el request,
     * se carga el correspondiente al usuario actual
     */
    Request::almacenarGet('id_usuario', $id_usuario, true);
}
```

Código extra JavaScript en `fin_body`

Agrega una función de pos-validación para impedir que se guarden los cambios si las dos contraseñas ingresadas no son iguales.

```
v.f_pos_validation = function () {
    if (obtener_elemento('contrasenia').value != obtener_elemento(' ↩
        contrasenia2').value) {
        alert('La contraseña ingresada difiere de su confirmación.  Por  ↩
            favor inténtelo nuevamente.');
        return false;
    }
```

```
                else return true;
            };
```

usuario_listado.xml

Listado de usuarios.

Figura 9.33: Captura de pantalla de usuario_listado.php en la aplicación “Localidades”

Definición de columnas

Las columnas del listado están definidas como sigue:

```
    <columnas>
        <columna llevaALink="true">                                    ❶
            <titulo>Nombre corto</titulo>
            <dato>nombre</dato>
            <link>usuario_sinclave_formulario.php</link>
        </columna>
        <columna>                                                      ❷
            <titulo>Nombre completo</titulo>
            <dato>nombre_ext</dato>
        </columna>
        <columna>                                                      ❸
            <titulo>Hab?</titulo>
            <tip>¿Habilitado?</tip>
            <condicion>
                <![CDATA[
                return !$GLOBALS['ocultar_borrados'];
                ]]>
            </condicion>
            <dato>vhabilitado</dato>
        </columna>
        <acciones>                                                     ❹
            <titulo></titulo>
            <condicion>
                <![CDATA[
                return $seguridad->puedeVerPagina('usuario_cambioclaveadmin_formulario. ↩
                    php');
                ]]>
```

```
            </condicion>
            <accion>
                <destino pasarParametros="true">
                    <url>usuario_cambioclaveadmin_formulario.php</url>
                </destino>
                <etiqueta>Cambiar contraseña</etiqueta>
            </accion>
        </acciones>
    </columnas>
```

1. `seguridad.usuario.nombre character(10)`. La definición de la columna incluye el elemento `columna/link` para especificar un formulario distinto que el predeterminado; en este caso, el link lleva al formulario de usuario sin cambio de contraseña, dado que ésta se modifica por separado.
2. `seguridad.usuario.nombre_ext character varying(30)`.
3. `seguridad.usuario.habilitado boolean`. La condición aplicada a esta columna hace que sólo sea visible cuando se están mostrando los usuarios deshabilitados, de acuerdo al estado del checkbox que se genera por la definición de borrado lógico incluída en el listado:

```
    <borrado-logico>
        <columna>habilitado</columna>
        <valor>false</valor>
        <con-control etiqueta="Sólo habilitados" generar="true"/>
    </borrado-logico>
```

La condición de la columna utiliza la variable booleana global `$ocultar_borrados`, definida automáticamente por XGAP, que indica el estado de presentación de items con borrado lógico:

```
<?php
return !$GLOBALS['ocultar_borrados'];
```

4. Se incluye una columna de acciones, con una única acción definida que lleva al formulario de cambio de contraseña del usuario correspondiente.

   La condición de la columna asegura que no sea visible si el usuario no tiene permiso para acceder al destino de la acción.

```
<?php
return $seguridad->puedeVerPagina('
    usuario_cambioclaveadmin_formulario.php');
```

usuario_sinclave_formulario.xml

Formulario de modificación de usuario, sin cambio de contraseña.

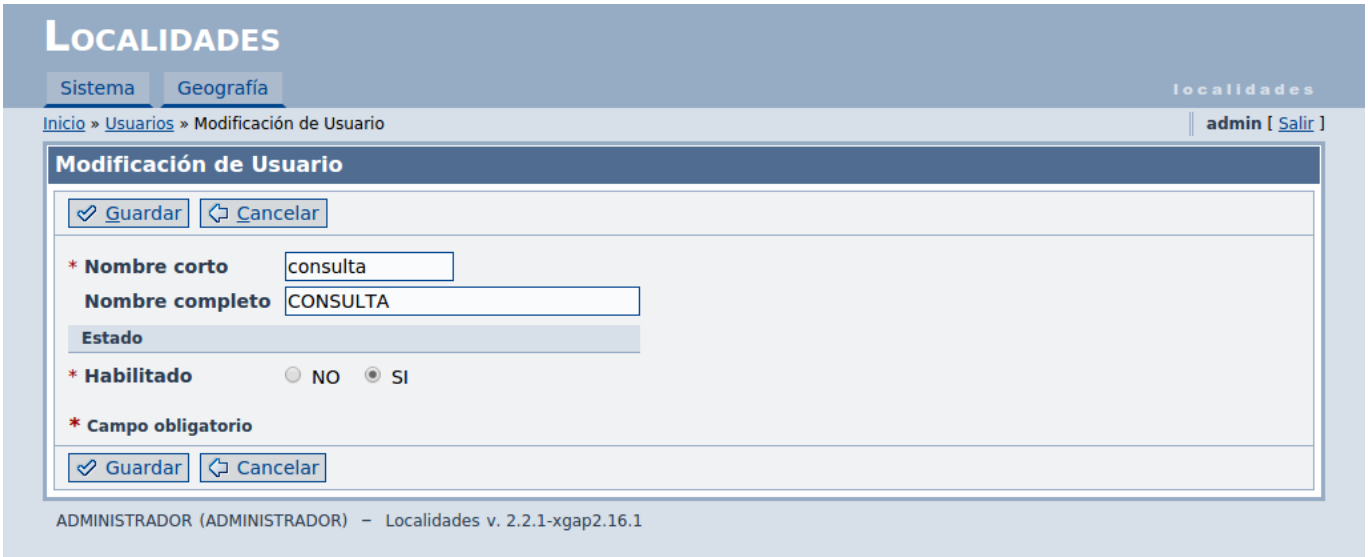


Figura 9.34: Captura de pantalla de usuario_sinclave_formulario.php en la aplicación “Localidades”

Es similar al formulario completo de usuario, con dos diferencias:

- No incluye los dos campos destinados al ingreso de contraseña.
- Se impide su uso para altas (debido al punto anterior).

Uso de XInclude

Para aprovechar la similitud mencionada, la mayor parte del código de `usuario_sinclave_formulario.xml` se obtiene de `usuario_formulario.xml` , además de un elemento de código extra de `usuario_cambioclave_formulario.xml`, por medio de elementos XInclude:

```
<formulario
            xmlns:xi="http://www.w3.org/2001/XInclude"
            >
    <!-- ... -->
    <xi:include href="usuario_formulario.xml"
                xpointer="xpointer(/formulario/configuracion)"/>
    <xi:include href="usuario_formulario.xml"
                xpointer="xpointer(/formulario/tabla)"/>
    <xi:include href="usuario_formulario.xml"
                xpointer="xpointer(/formulario/clave)"/>
    <campos>
        <xi:include href="usuario_formulario.xml"
                    xpointer="xpointer(/formulario/campos/campo[dato='nombre'])"/>
        <xi:include href="usuario_formulario.xml"
                    xpointer="xpointer(/formulario/campos/campo[dato='nombre_ext'])"/ ↩
                        >
        <xi:include href="usuario_formulario.xml"
                    xpointer="xpointer(/formulario/campos/separador_grupo[1])"/>
        <xi:include href="usuario_formulario.xml"
                    xpointer="xpointer(/formulario/campos/campo[dato='habilitado'])"/ ↩
                        >
    </campos>
    <codigoExtra>
```

```
        <xi:include href="usuario_formulario.xml"
                    xpointer="xpointer(/formulario/codigoExtra/codigo[@tipo='PHP' and ↩
                        @ubicacion='xgap_cargado'])"/>
        <xi:include href="usuario_cambioclave_formulario.xml"
                    xpointer="xpointer(/formulario/codigoExtra/codigo[@tipo='PHP' and ↩
                        @ubicacion='despues_inicializacion'])"/>
    </codigoExtra>
</formulario>
```

Código extra PHP en `xgap_cargado`
Definido en usuario_formulario.xml.

Código extra PHP en `despues_inicializacion`
Definido en usuario_cambioclave_formulario.xml.

Cambios al modelo básico de aplicación predefinido por XGAP

Los scripts SQL y páginas iniciales provistos por XGAP definen un modelo básico de aplicación. En esta sección se detallan los cambios que tiene la aplicación "Localidades" respecto a ese modelo.

Cambios en funcionalidad

- Posibilidad de definir una página de inicio diferente para cada rol funcional.
- Los usuarios se pueden deshabilitar.

Cambios en el esquema de base de datos

- Se agregan tablas y vistas correspondientes al modelo propio de la aplicación, junto con otras tablas auxiliares.
- Se agrega columna `pagina_inicio` a la tabla `seguridad.rolf`.
- Se agrega columna `habilitado` a la tabla `seguridad.usuario`.

Cambios en las páginas predefinidas

`login_contenido.xml`

- Se impide el ingreso de usuarios deshabilitados.

`login_contenido.xml` y `seleccionarrol_contenido.xml`

- Se tiene en cuenta la página de inicio que está definida para el rol funcional del usuario, para establecer el destino de la redirección final que realizan ambas páginas.
- Se almacena la página de inicio en la sesión.

rolf_formulario.xml

- Se agrega un campo correspondiente a la columna `seguridad.rolf.pagina_inicio`.

rolf_listado.xml

- Se agrega una columna para mostrar el valor de `seguridad.rolf.pagina_inicio`.

usuario_formulario.xml y usuario_sinclave_formulario.xml

- Se agrega un campo correspondiente a la columna `seguridad.usuario.habilitado`.

usuario_listado.xml

- Se agrega uso de borrado lógico, de acuerdo al valor de `seguridad.usuario.habilitado`, y una columna para mostrar el estado de habilitación del usuario.

Parte V

Apéndices

Apéndice A

Software requerido para convertir la Guía a otros formatos

Los fuentes de esta guía están escritos en AsciiDoc, un lenguage de marcado ligero. Para convertir los fuentes a otros formatos, se requiere el siguiente software:

General

- AsciiDoc
- Graphviz
- PlantUML
- Shaape
- Python, requerido por AsciiDoc
- Java JRE, requerido por PlantUML
- cairo con enlaces Python, requerido por Shaape
- pango, requerido por Shaape

Salida HTML

- GNU source-highlight o Highlight o Pygments

Salida PDF

- xsltproc
- DocBook XSL Stylesheets
- dblatex o FOP
- LaTeX, sólo si se usa dblatex en lugar de FOP.