SowerPHP
Minimalist Framework for PHP

Esteban De La Fuente Rubio

http://sowerphp.org
https://github.com/SowerPHP

Contenido

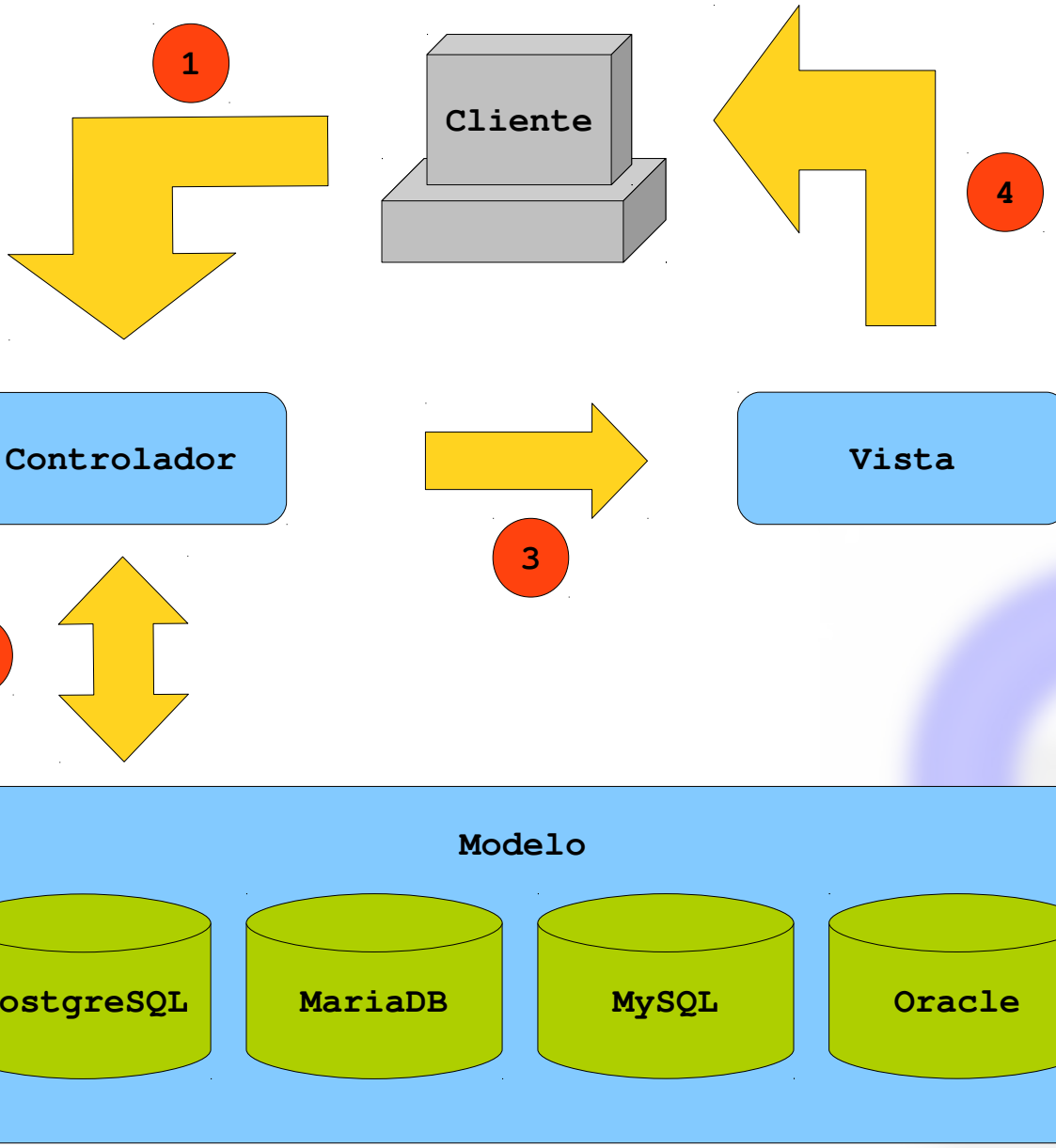
- Introducción
  - ¿Qué es SowerPHP?
  - ¿Cómo funciona?
- Parte “práctica”:
  - Instalación y puesta en marcha
  - Ejemplos de funcionalidades (extensiones y módulos)
- Tareas pendientes/futuras
- Enlaces de interés

SowerPHP

- Framework minimalista para PHP.
- Basado originalmente en CakePHP*.
- Fork del proyecto MiPaGiNa que a su vez fue sucesor del proyecto MiInTrAnEt (pensado para MiPyMEs, pero era complejo y desordenado) y este fue sucesor del proyecto MiSiTiO.
- Utiliza MVC (modelo, vista, controlador).

** Actualmente bastante alejado de la idea original de CakePHP*

Diagrama MVC



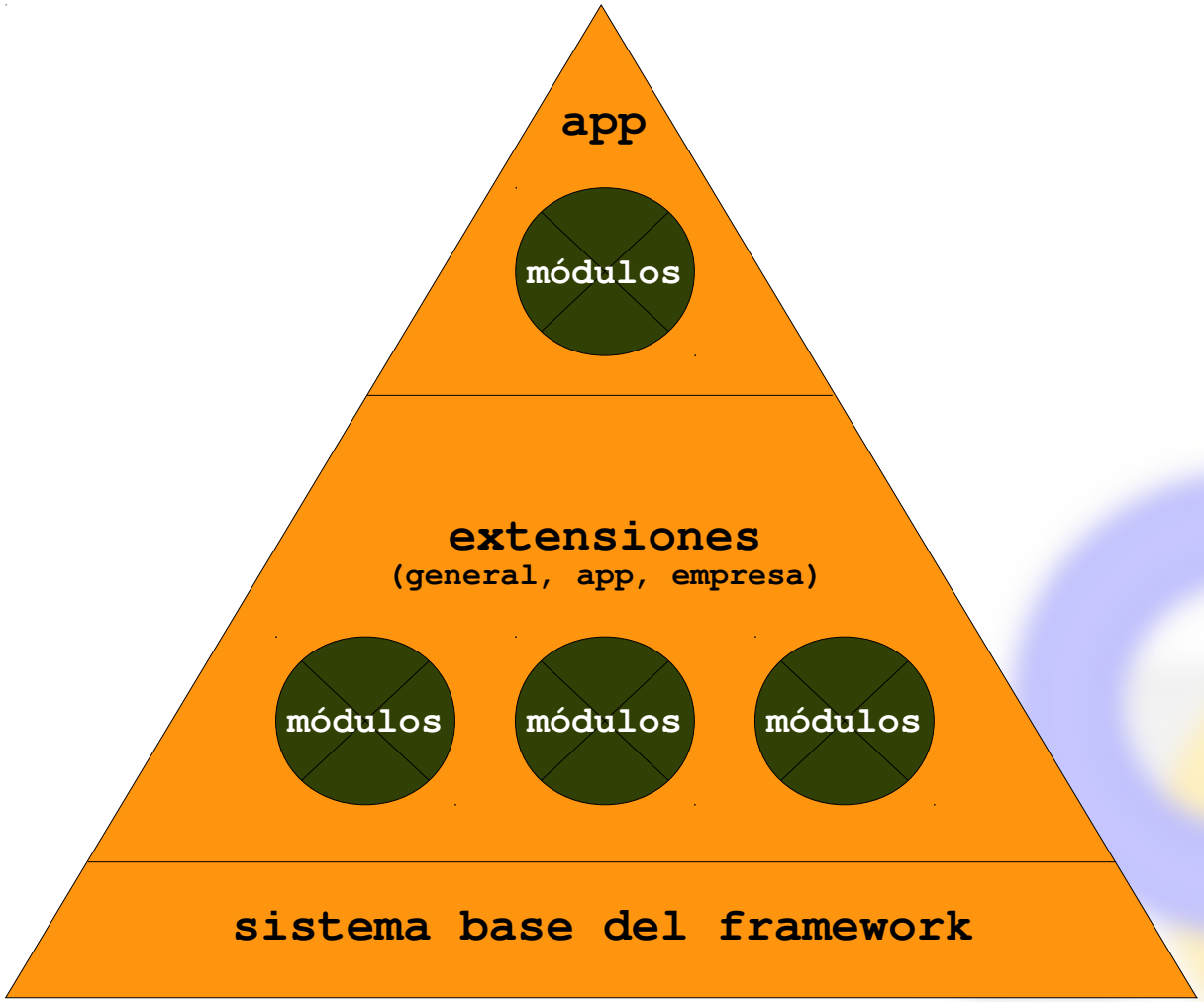
Características base (<500KB)

- Soporte de módulos.
- Soporte de base de datos.
- Creación de scripts de terminal.
- Múltiples layouts (varios en uso al mismo tiempo)
- Soporte de diferentes tipos de páginas para renderizar (por defecto PHP y Markdown).
- Internacionalización (I18n).
- Clases para acciones sencillas (como enviar correo, solicitudes HTTP o manejo de errores).

Extensiones

- Aparte de soportar módulos, el sistema soporta el concepto de extensiones.
- Son una capa que se coloca sobre el sistema base (directorio *lib/sowerphp/core*) u otras extensiones.
- Las extensiones proveen funcionalidades específicas para un tipo de aplicación o página web (ejemplo: empresa).
- Cada extensión puede contener módulos o bien sobreescribir existentes (lo mismo con cada una de las clases, no así las funciones).

Extensiones y módulos



Ejemplo: extensión general, módulo Exportar

Además el módulo permite exportar a códigos de barra 1D y 2D (PDF417 y QR)

¿Por qué un framework nuevo?

- A fines de 2011 se revisó CakePHP y se encontraron cosas “extrañas” o que no cumplían con lo que se esperaba, ejemplos:
  - Uso de una PK por tabla (ID único).
  - Plugins de CakePHP vs extensiones y módulos de SowerPHP.
  - Generación automática de código (mantenedores) buena, pero se podía mejorar (ejemplo: filtros).
  - Framework muy genérico (para muchos escenarios).

¿Por qué un framework nuevo?

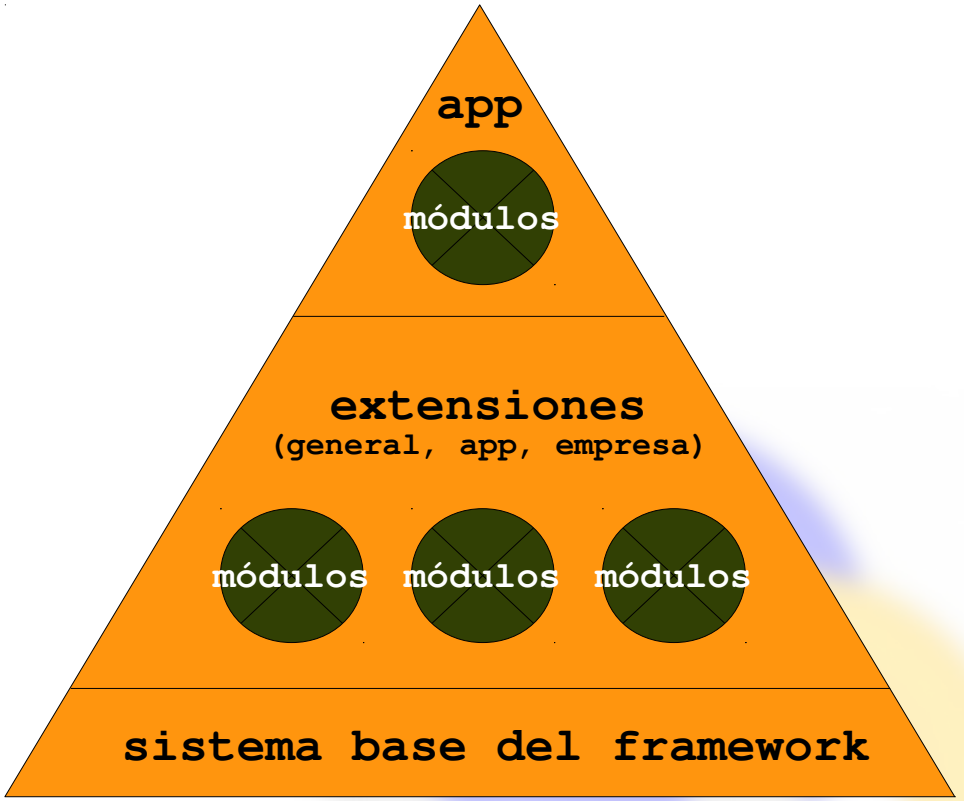
- La idea fue construir un framework que considerará un escenario más acotado (por ejemplo: permitir su uso sólo en Apache y PHP>=5.5). Con esto se sabe que funcionará en menos casos, pero que funcione de forma más eficiente (es la idea...).
- La razón más importante: "porque se puede", por el afán de aprender y entender como se construye un framework.

Directorios del framework

- Framework se compone en su nivel superior de 3 directorios:
  - lib: base del framework (contiene a sowerphp/core).
  - project: directorio para el proyecto que se desarrolla.
  - extensions: directorio donde va cada una de las extensiones que se instalan.
- En *project* el usuario dispondrá, básicamente, solo de los archivos para lanzar una página web (index.php) o lanzar un comando de terminal (shell.php).

¿Cómo se usan los directorios?

- Al buscar archivos (de clases principalemente*) son obtenidos desde la punta de la pirámide hacia abajo.
- El primer archivo que se encuentra es el que se utiliza.
- Lo anterior permite que sea muy simple “sobre escribir” archivos sin alterar los originales.



** Solo en caso que las clases hayan sido instanciadas o llamadas estáticamente sin un namespace, ejemplo: \Clase*

Directorios y *namespaces*

- Directorio base es un *namespace* (sowerphp\core).
- Una extensión es un *namespace* (vendor\extension).
- Directorio project/website es un *namespace* (website).
- Los módulos dentro de los directorios anteriores son *namespaces*:
  - vendor\extension\Modulo
  - vendor\extension\Modulo\Submodulo
  - website\Modulo
  - website\Modulo\Submodulo

Directorios en *lib/sowerphp/core*

- Config: archivos de configuración.
- Controller: controladores.
- Exception: excepciones.
- Model: modelos y acceso a bases de datos.
- Network: clases relacionadas con comunicación por red.

Directorios en *lib/sowerphp/core*

- Routing: procesamiento de url y ejecución de la web.
- Shell: clases para ejecutar comándos por terminal.
- Utility: utilidades varias.
- Vendor: bibliotecas externas (otros proveedores).
- View: clases de las vistas y vistas.
- webroot: archivos “reales” (como imágenes).

Archivos en *lib/sowerphp/core*

Algunos archivos dentro del directorio son:

- basics.php: funciones.
- bootstrap.php: proceso que se ejecuta al inicio.
- Config/core.php: configuración por defecto de la web.
- Config/routes.php: rutas, indican que hacer con las URLs.
- Controller/App.php: controlador que se debe reemplazar en caso de tener funciones en común y debe ser extendido por todo controlador creado.

Archivos en *lib/sowerphp/core*

- Model/App.php: modelo que se debe reemplazar en caso de tener funciones en común y debe ser extendido por los modelos creado que utilicen BD (estos están “obligados”).
- Network/Email.php: clase para envio de correo electrónico.
- Network/Http/Socket.php: clase para hacer consultas a través de métodos de HTTP (como POST).

Archivos en *lib/sowerphp/core*

- Shell/App.php: shell que se debe reemplazar en caso de tener funciones en común y debe ser extendida por todas las shells.
- View/Helper/Pages/*.php: clases que renderizan páginas de diferentes extensiones.

Clases y métodos ejecución web

- Configure:bootstrap(): carga la configuración de la aplicación, cargando archivos core.php y routes.php de todas las extensiones (no módulos).
- Routing_Dispatcher::dispatch(): despacha la página que se esté solicitando.
- Routing_Router::parse(): procesa una URL y determina que es lo que se desea obtener (módulo, controlador, acción y parámetros).

Clases y métodos ejecución web

- Controller::”action”(): acción solicitada al controlador.
- View::render(): método que realiza el renderizado de la vista.

Ejemplos de URLs aceptadas

example.com/pagina_estatica

example.com/controlador/accion

example.com/modulo/controlador/accion

example.com/modulo/submodulo/controlador/accion

Si la acción no se especifica se lanzará la acción index por defecto, además cada acción puede recibir parámetros.
Pueden haber tandos submódulos como se desee.

Ejecución página web*

** Se han omitido ejecuciones de errores (por ejemplo al no encontrar un controlador)*
*** A pesar de ser una página estática igualmente se ejecuta un controlador, este es Pages::render() que renderiza la página estática*
**** La vista en realidad devuelve el control al controlador y este es quien envía la respuesta al usuario o realiza otra acción (como enviar un email)*

Instalación y puesta en marcha

- Requerimientos:
  - Apache 2: con mod_rewrite
  - PHP >=5.5: idealmente con PEAR (para instalar biblioteca para correo), soporte para BD y GD (para generación de gráficos).
  - Composer, git y mercurial (depende de lo que se requiera instalar).
  - Motor de base de datos: PostgreSQL o MariaDB por ejemplo.

En http://sowerphp.org/doc/general/servidor_web encontrará más detalles de la configuración del servidor web

Instalación y puesta en marcha

- Instalar con composer:

```
$ composer create-project sowerphp/sowerphp sower \
    --stability="dev"
```

- O, clonar con git:

```
$ git clone https://github.com/SowerPHP/sowerphp.git
```

En http://sowerphp.org/doc/paso_a_paso/instalacion encontrará más detalles sobre la instalación y puesta en marcha del framework

Instalación y puesta en marcha

Este es el nuevo layout por defecto. Sin embargo los ejemplos que se mostrarán fueron hechos con el layout antiguo (de MiPaGiNa).

Composer

- El framework utiliza composer* para la administración de bibliotecas externas (*vendors*) e instalación de extensiones.
- El sistema base permite utilizar Markdown, por lo cual si se requiere se debe instalar la biblioteca mediante composer:

```
$ cd /ruta/a/sowerphp/lib/sowerphp/core
$ composer install
```

- Verificar otras dependencias en composer.json

** Composer es una herramienta que debe ser instalada por separado, y podría requerir otros paquetes (como git o mercurial)*

Ejemplos

- Se mostrarán diferentes ejemplos construídos con el framework.
- La idea es mostrar como lograr diferentes cosas con las funcionalidades disponibles.
- Finalmente se mostrará como es la construcción inicial de una aplicación web con un sistema de autenticación.
- Algunos ejemplos no usarán MVC, se trabajará solo con las páginas “estáticas” (¿contradicción?).

Ejemplo: “Hola Mundo”

- Con la instalación antes mencionada se crea el archivo View/Pages/inicio.php con el siguiente contenido:

```
<h1>Hola mundo!</h1>

<p>Bienvenid@ a mi página</p>
```

- También se podría haber creado el archivo View/Pages/inicio.md con el contenido:

```
Hola mundo!
===========


Bienvenid@ a mi página
```

Ejemplo: “Hola Mundo”

MiPaGiNa

Inicio

Hola mundo!

Bienvenid@ a mi página

powered by MiPaGiNa

Ejemplo: otra página de inicio

- Agregar a Config/core.php:

```
\sowerphp\core\Configure::write('homepage', 'otra');
```

- Con esto ahora nuestra página de inicio será el archivo View/Pages/otra.php (o otra.md).
- No olvidar crear el nuevo archivo o habrá error:

```
# echo Otra pág > /var/www/pagina/View/Pages/otra.php
```

- A través del archivo Config/core.php se pueden cambiar layouts, textos y otras configuraciones.

Ejemplo: enviar un email

- Se debe configurar la cuenta de correo que enviará el correo (se utiliza SMTP para el envio). Esto se configura en Config/core.php:

```
\sowerphp\core\Configure::write('email.default', array(
  'type' => 'smtp',
  'host' => 'ssl://smtp.gmail.com',
  'port' => 465,
  'user' => 'no-reply@example.com',
  'pass' => 'XXXXXX',
));
```

Además se puede definir el indice 'to' que es necesaria para recibir correo (formulario de contacto de la extensión general).

Ejemplo: enviar un email

- Crear un archivo View/Pages/email.php:

```
<?php
$email = new \sowerphp\core\Network_Email();
$email->to('destinatario@example');
$email->subject('Mensaje de prueba');
$email->send('Probando envio de correo');
\sowerphp\core\Model_Datasource_Session::message(
  'Correo electrónico ha sido enviado'
);
```

- Ingresar vía web a la página: example.com/email

\sowerphp\core\Model_Datasource_Session::message() escribe un mensaje para ser mostrado al renderizar la página.

Ejemplo: enviar un email

MiPaGiNa

Inicio

Correo electrónico ha sido enviado

powered by MiPaGiNa

Mensaje de prueba Recibidos x

SASCO <no-reply@sasco.cl> 1:26 (hace 3 minutos)

para

Probando envio de correo

Instalación de extensiones

- Las extensiones existen en GitHUB como repositorios *extension-unaextension*. Pero además son provistos a través de:
  - https://packagist.org/packages/SowerPHP
- Instalar usando composer (ejemplo extensión layouts):

```
$ composer require sowerphp/layouts dev-master
```

- La otra alternativa es clonar el repositorio y dejar las extensiones en el directorio que corresponda (hasta ahora esto es obligatorio para extensión general, ¿bug?).
- Se pueden instalar globalmente o por proyecto.

Uso de extensión general

- Los siguientes ejemplos hacen uso de la extensión general, por lo cual se debe habilitar (al menos) para el archivo webroot/index.php dejando la variable de extensiones de la forma:

```
$_EXTENSIONS = array('sowerphp/general');
```

- Instalar bibliotecas de *vendors* con composer.
- Con esto la extensión ya estará operativa.

Ejemplo: página de contacto

- Al habilitar la extensión y configurar el correo electrónico se habilita automáticamente el formulario de contacto en example.com/contacto:

Si se cargan los archivos js/__.js y js/form.js el formulario es validado antes de ser enviado (estos archivos se cargan solos con la extensión app).

Ejemplo: modificar menú web

- Tenemos formulario de contacto, pero no hay link para acceder, editamos Config/core.php para agregar el enlace al menú:

```
\sowerphp\core\Configure::write('nav.website', array(
  '/inicio'=>'Inicio',
  '/contacto'=>'Contacto',
));
```

- Adicionalmente, si utilizamos la extensión app existe la configuración nav.app que permite modificar el menú de la aplicación.

Ejemplo: modificar menú web

MiPaGiNa

Inicio Contacto

Contacto

Por favor enviar su mensaje a través del siguiente formulario, será contactado a la brevedad.

Nombre

Correo electrónico

Mensaje

Enviar mensaje

powered by MiPaGiNa

Módulo Exportar

- La extensión general trae el módulo Exportar mencionado antes. Para habilitarlo editar Config/core.php:

```
\sowerphp\core\Module::uses(array(
    'Exportar'
));
```

Ejemplo: exportar a código QR

- Ir a la URL:

```
http://example.com/exportar/qrcode/TWlQYUdpTmEK
```

- El string que se desea codificar es pasado en base64.

- Para exportar a código de barras de 1D y PDF417 se utiliza barcode y pdf417 respectivamente en la URL.

Ejemplo: TableHelper

- Para generar fácilmente una tabla en HTML utilizar:

```
<?php
new \sowerphp\general\View_Helper_Table (array(
  array('f1c1', 'f1c2', 'f1c3'),
  array('f2c1', 'f2c2', 'f2c3'),
  array('f3c1', 'f3c2', 'f3c3'),
  array('f4c1', 'f4c2', 'f4c3'),
));
```

Ejemplo: TableHelper

MiPaGiNa

Inicio Contacto

f1c1	f1c2	f1c3
f2c1	f2c2	f2c3
f3c1	f3c2	f3c3
f4c1	f4c2	f4c3

powered by MiPaGiNa

¿y? ¿por qué no generarla directamente con HTML?

Ejemplo: TableHelper 2

- Las tablas pueden ser “automágicamente” exportadas a diferentes formatos:

```
<?php
new \sowerphp\general\View_Helper_Table (array(
  array('f1c1', 'f1c2', 'f1c3'),
  array('f2c1', 'f2c2', 'f2c3'),
  array('f3c1', 'f3c2', 'f3c3'),
  array('f4c1', 'f4c2', 'f4c3'),
), 'tabla', true);
```

Ejemplo: TableHelper 2

MiPaGiNa

Inicio Contacto

f1c1	f1c2	f1c3
f2c1	f2c2	f2c3
f3c1	f3c2	f3c3
f4c1	f4c2	f4c3

powered by MiPaGiNa

Signos más y menos aparecen pero no funcionan, se requiere js/jquery.js. Más adelante se mostrarán ejemplos donde estos funciona ok.

Ejemplo: TableHelper 2

```
1 "f1c1","f1c2","f1c3"
2 "f2c1","f2c2","f2c3"
3 "f3c1","f3c2","f3c3"
4 "f4c1","f4c2","f4c3"
5
```

	A	B	C
1	f1c1	f1c2	f1c3
2	f2c1	f2c2	f2c3
3	f3c1	f3c2	f3c3
4	f4c1	f4c2	f4c3
5			

This XML file does not appear to

```
▼<tabla>
  <script/>
 ▼<tabla>
    <f1c1>f2c1</f1c1>
    <f1c2>f2c2</f1c2>
    <f1c3>f2c3</f1c3>
  </tabla>
 ▼<tabla>
    <f1c1>f3c1</f1c1>
    <f1c2>f3c2</f1c2>
    <f1c3>f3c3</f1c3>
  </tabla>
 ▼<tabla>
    <f1c1>f4c1</f1c1>
    <f1c2>f4c2</f1c2>
    <f1c3>f4c3</f1c3>
  </tabla>
</tabla>
```

MiPaGiNa
Tabla: tabla

f1c1	f1c2	f1c3
f2c1	f2c2	f2c3
f3c1	f3c2	f3c3
f4c1	f4c2	f4c3

```
[["f1c1","f1c2","f1c3"],["f2c1","f2c2","f2c3"],["f3c1","f3c2","f3c3"],["f4c1","f4c2","f4c3"]]
```

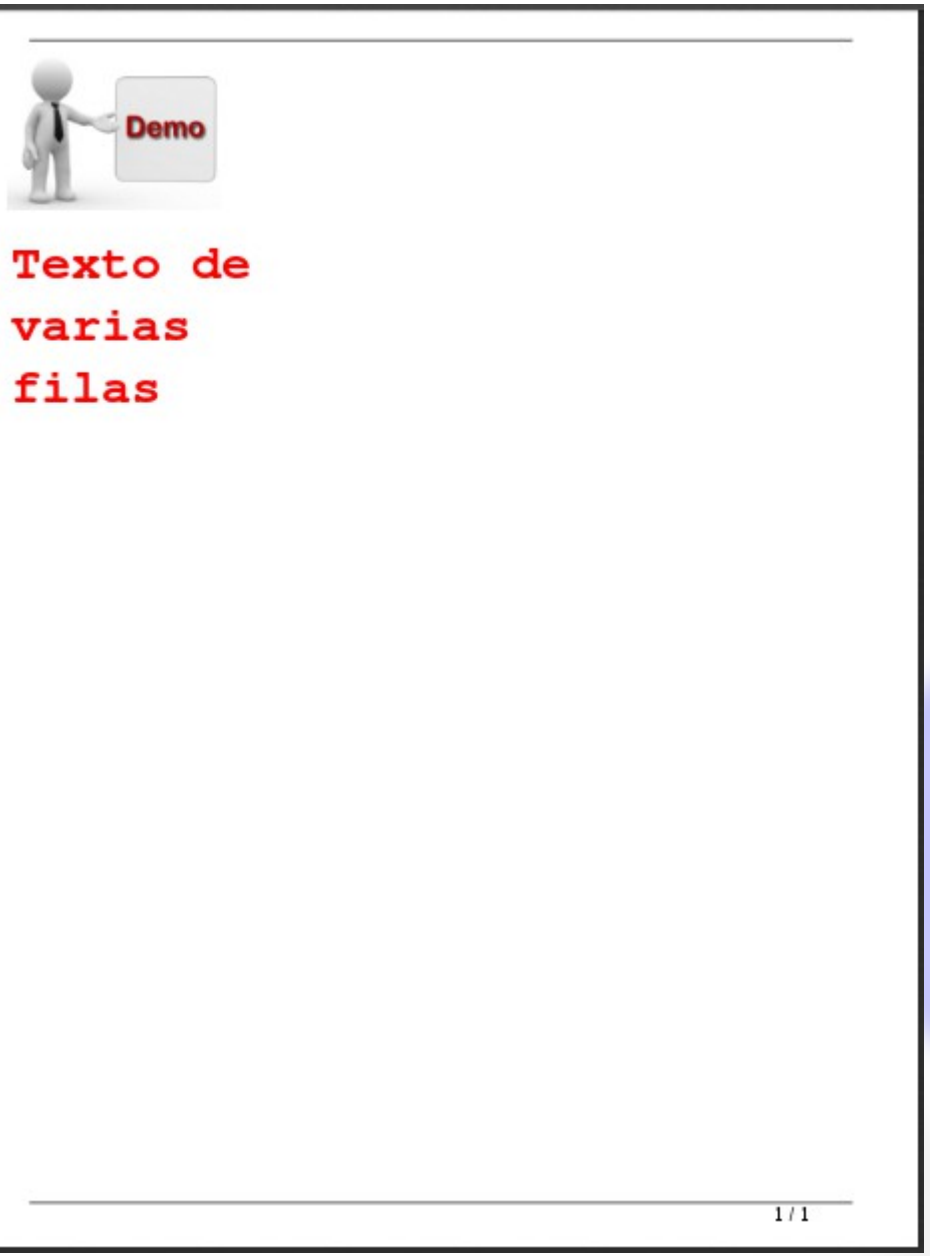
Ejemplo: PDFHelper

- Para generar un PDF utilizar:

```
<?php
$pdf = new \sowerphp\general\View_Helper_PDF();
$pdf->setFont ('freemono', 'B', 32);
$pdf->AddPage();
$logo = DIR_WEBSITE.'/webroot/img/logo.png';
$pdf->Image($logo, 10, 10, 0, 0, 'PNG', 'http://example.com');
$pdf->SetTextColorArray(array(255, 0, 0));
$pdf->MultiTexto (10, 50, 'Texto de varias filas', 'L', 60);
$pdf->Output('archivo.pdf');
exit(0);
```

- El helper hereda de TCPDF.

Ejemplo: PDFHelper



Ejemplo: ChartHelper

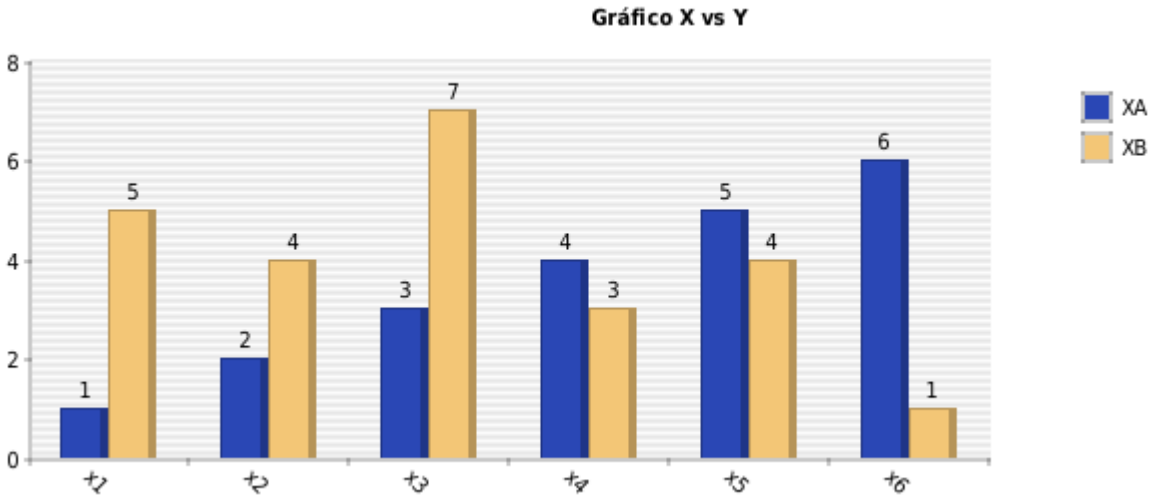
- Para generar un gráfico de barras verticales utilizar:

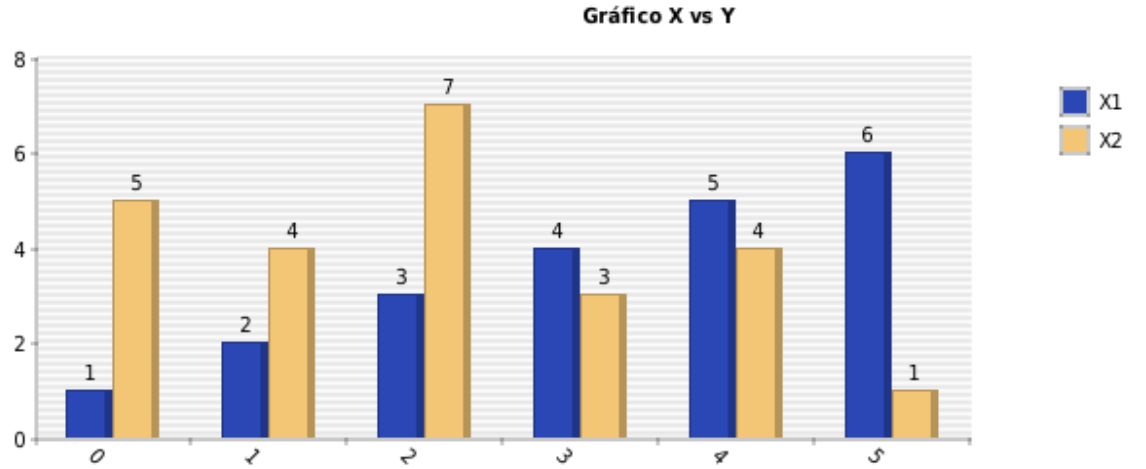
```
<?php
$chart = new \sowerphp\general\View_Helper_Chart ();
$chart->vertical_bar ('Gráfico X vs Y', array(
  'XA' =>
        array('x1'=>1,'x2'=>2,'x3'=>3,'x4'=>4,'x5'=>5,'x6'=>6),
  'XB' =>
        array('x1'=>5,'x2'=>4,'x3'=>7,'x4'=>3,'x5'=>4,'x6'=>1),
));
```

- El helper hace uso de libchart*.

** Una vez instalado libchart con composer se debe ingresar al directorio Vendor/libchart/libchart/ y ejecutar "composer install"*

Ejemplo: ChartHelper





En el gráfico inferior se omitieron los índices asociados a los datos (x1, x2, x3...) al generar el gráfico

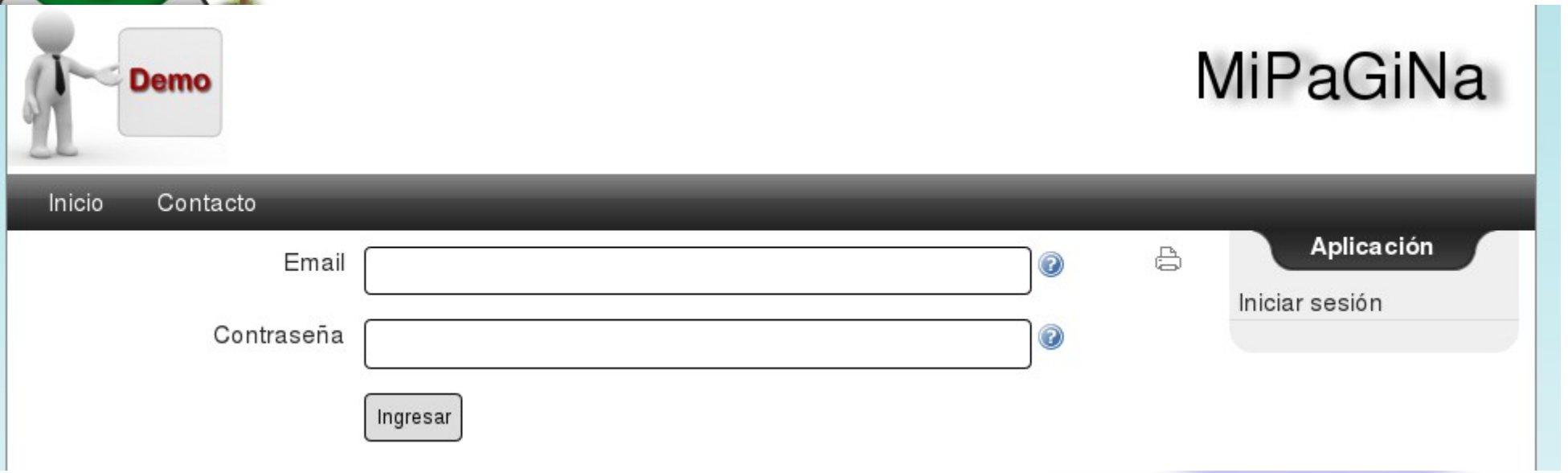
Ejemplo: FormHelper

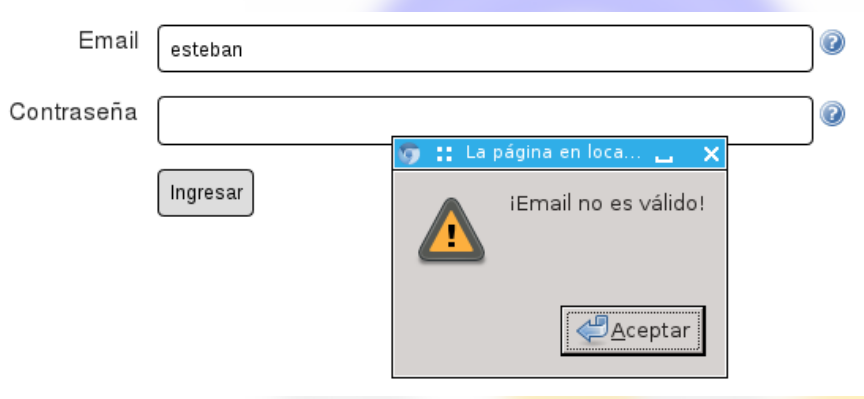
- Para generar un formulario HTML utilizar:

```
<?php
$f = new \sowerphp\general\View_Helper_Form ();
echo $f->begin(array('focus'=>'email', 'onsubmit'=>'Form.check()'));
echo $f->input(array(
  'name'=>'email', 'label'=>'Email',
  'check'=>'notempty email', 'help'=>'Ingrese su email'
));
echo $f->input(array(
  'type'=>'password', 'name'=>'contrasenia', 'label'=>'Contraseña',
  'check'=>'notempty', 'help'=>'Ingrese su contraseña',
));
echo $f->end('Ingresar');
```

Se debe incluir js/jquery.js y js/form.js o bien usar la extensión app con su layout para que funcionen los chequeos en el formulario.

Ejemplo: FormHelper





Se está utilizando extensión app que ya incluye los scripts js/__.js y js/form.js

Ejemplo: FormHelper

Tipos de inputs soportados:

- hidden
- text
- password
- textarea
- checkbox
- checkboxes
- date
- file
- select
- radios
- js
- tablecheck

Ejemplo: FormHelper 2

- Para generar campos de forma dinámica utilizar tipo js de la siguiente forma:

```
echo $f->begin();
echo $f->input(array(
  'type' => 'js',
  'id' => 'items',
  'label' => 'Items',
  'titles' => array ('ID', 'Descripción'),
  'inputs' => array (
    array('name'=>'id'), array('name'=>'descripcion')
  )
));
echo $f->end();
```

Ejemplo: FormHelper 2

Items

ID	Descripción	

Enviar

Se está utilizando extensión app que ya incluye los scripts js/__.js y js/form.js

Ejemplo: FormHelper 3

- Para generar una tabla con elementos y seleccionar uno o varios utilizar tipo tablecheck de la siguiente forma:

```
echo $f->begin();
echo $f->input (array(
  'type'=>'tablecheck',
  'name'=>'items',
  'label'=>'Items',
  'titles'=>array('ID', 'Descripción'),
  'table'=>array(array(1, 'D1'), array(2, 'D2'), array(3, 'D3')),
));
echo $f->end('Seleccionar');
```

Ejemplo: FormHelper 3

Items

ID	Descripción	☑
1	D1	☑
2	D2	☑
3	D3	☑

Seleccionar

Se está utilizando extensión app que ya incluye los scripts js/jquery.js y js/form.js

Uso de extensión app

- Los siguientes ejemplos hacen uso de la extensión app, por lo cual se debe habilitar (al menos) para el archivo webroot/index.php dejando la variable de extensiones de la forma:

```
$_EXTENSIONS = array('sowerphp/app', 'sowerphp/general');
```

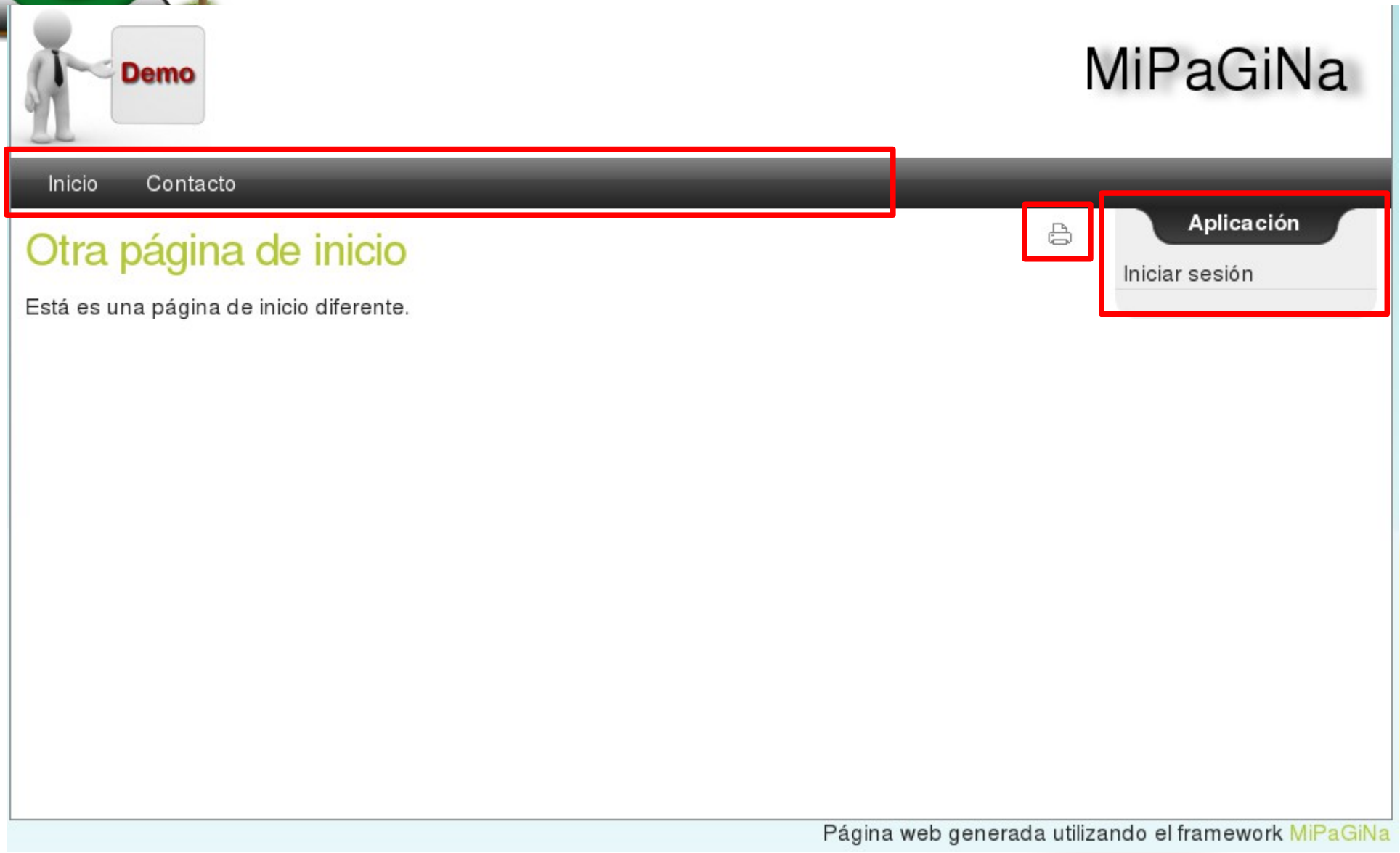
- Extensión app tiene como requisito la extensión general (orden es importante, debe ir primero app).
- Editar Config/core.php y seleccionar layout App*:

```
\sowerphp\core\Configure::write('page.layout', 'App');
```

- Con esto la extensión ya estará operativa.

** Si el layout no cambia es porque se asigna con la sesión, borrar datos de la web o asignar con example.com/session/config/page.layout/App*

Uso de extensión app



Ejemplo: uso de base de datos

- Supongamos que tenemos una base de datos en PostgreSQL llamada **demo** con la siguiente tabla:

```
CREATE TABLE parametro (
  modulo VARCHAR (20) NOT NULL,
  parametro VARCHAR (20) NOT NULL,
  valor VARCHAR (20) NOT NULL,
  CONSTRAINT parametro_pk PRIMARY KEY
    (modulo, parametro)
);
```

Ejemplo: uso de base de datos

- Debemos configurar los datos de acceso a la base de datos en Config/core.php usando:

```
\sowerphp\core\Configure::write('database.default', array(
  'type' => 'PostgreSQL',
  'user' => 'usuario_bd',
  'pass' => 'XXXXXX',
  'name' => 'demo',
));
```

Ejemplo: uso de base de datos

- Debemos recuperar el objeto que representa la conexión a la base de datos:

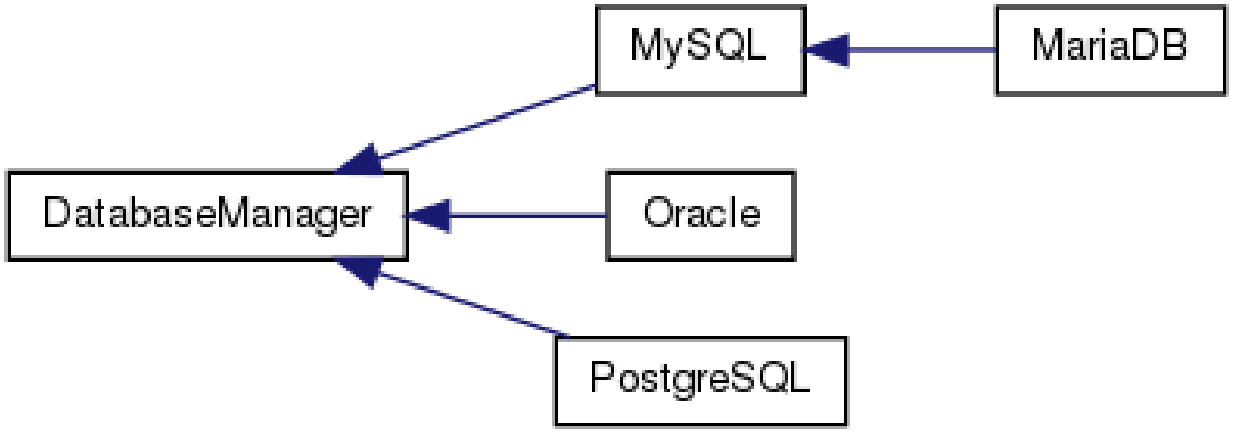
```
$db = &\sowerphp\core\Model_Datasource_Database::get();
```

- Lo anterior es equivalente a:

```
$db = &\sowerphp\core\Model_Datasource_Database::get(
        'default');
```

- ***default*** es el nombre utilizado para la configuración de la BD en Config/core.php
- A través de los métodos de ***$db*** interactuamos con la BD (query, getTable, getValue, getCol, getRow, etc).

Ejemplo: uso de base de datos



** Imagen no representa los nombres reales de las clases, pero la idea es la misma*

Ejemplo: uso de base de datos

- Insertar datos:

```
$db->query('
  INSERT INTO parametro VALUES
    (\'mod1\', \'par1\', \'val1\'),
    (\'mod1\', \'par2\', \'val2\'),
    (\'mod2\', \'par1\', \'val1\'),
    (\'mod2\', \'par2\', \'val2\'),
    (\'mod3\', \'par1\', \'val1\')
  ;
');
```

Ejemplo: uso de base de datos

- Consultar datos insertados:

```
new \sowerphp\general\View_Helper_Table (
  $db->getTableWithColsNames('
    SELECT * FROM parametro ORDER BY modulo, parametro
  '), 'parametros', true);
```

modulo	parametro	valor
mod1	par1	val1
mod1	par2	val2
mod2	par1	val1
mod2	par2	val2
mod3	par1	val1

Ejemplo: uso de base de datos

MiPaGiNa
Tabla: parametros

modulo	parametro	valor
mod1	par1	val1
mod1	par2	val2
mod2	par1	val1
mod2	par2	val2
mod3	par1	val1

	A	B	
1	modulo	parametro	valor
2	mod1	par1	val1
3	mod1	par2	val2
4	mod2	par1	val1
5	mod2	par2	val2
6	mod3	par1	val1

```
▼<parametros>
  <script/>
  ▼<parametro>
    <modulo>mod1</modulo>
    <parametro>par1</parametro>
    <valor>val1</valor>
  </parametro>
  ▼<parametro>
    <modulo>mod1</modulo>
    <parametro>par2</parametro>
    <valor>val2</valor>
  </parametro>
```

Ejemplo: uso de base de datos

Métodos principales del objeto ***$db***:

- query(): usado para INSERT, DELETE y UPDATE.
- getTable(): recupera datos en una tabla.
- getValue(): recupera un valor único.
- getCol(): recupera una columna.
- getRow(): recupera una fila (un registro).

Ejemplo: autenticación

- Utilizando la base de datos demo se carga el esquema (y datos) disponible en la extensión app para autenticación:

```
$ cd sowerphp/app/Module/Sistema/Module/Usuarios/Model/Sql/PostgreSQL
$ psql demo < usuarios.sql
```

- Lo anterior creará las tablas:
  - auth: asigna permisos de un grupo sobre los recursos.
  - grupo: grupos del sistema.
  - usuario: usuarios del sistema.
  - usuario_grupo: relación entre usuarios y sus grupos.

Ejemplo: autenticación

- Listo! Si, eso es todo.

Iniciar sesión

Usuario admin

Contraseña •••••

Ingresar

¿No recuerda su usuario o contraseña?, click aquí para recuperar.

Usuario *admin* ha iniciado su sesión
Último ingreso fue el *2014-02-27 15:07:05* desde *::1*

Por defecto usuario admin con clave admin

Perfil de usuario

Perfil del usuario *admin*

Datos personales

Nombre: Administrador

Usuario: admin

Email: admin@example.com

Hash: t7dr5B1ujphds043WMMEFWwFLeyWYqMU

Guardar cambios

Cambiar contraseña

Contraseña:

Repetir contraseña:

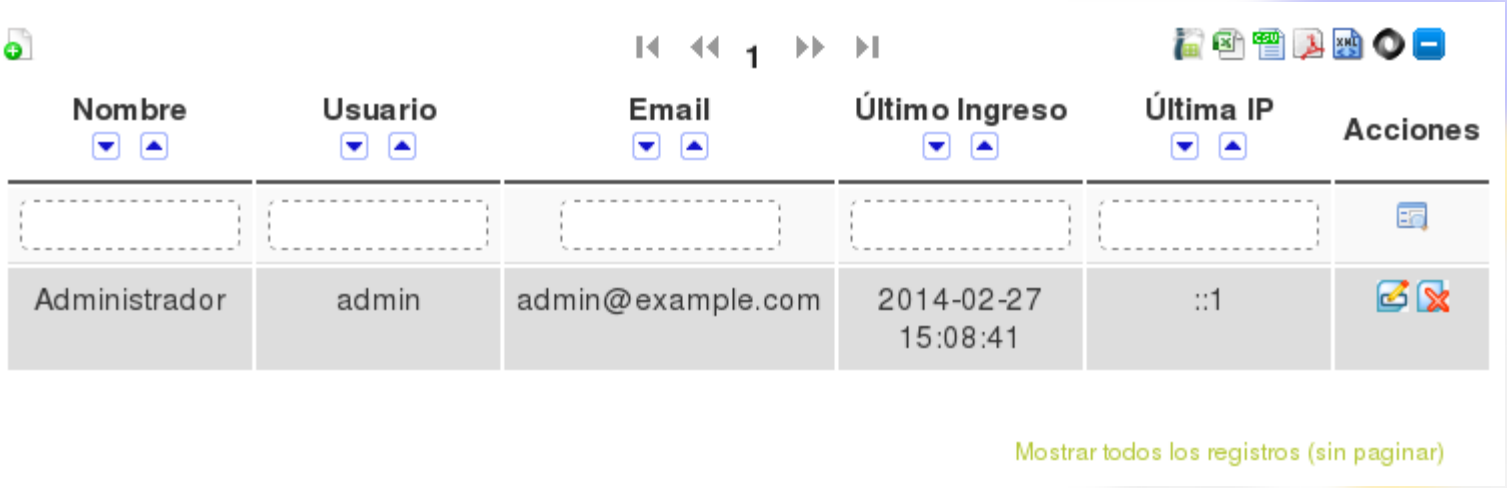
Cambiar contraseña

Aplicación

Sistema

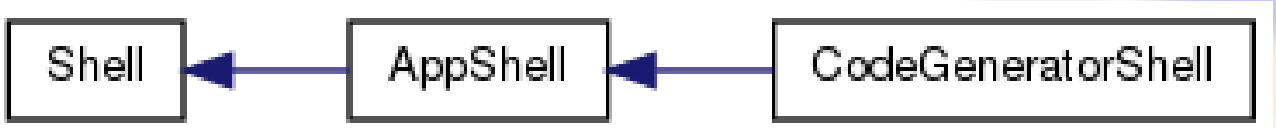
Mantenedor de usuarios

Sistema » Usuarios



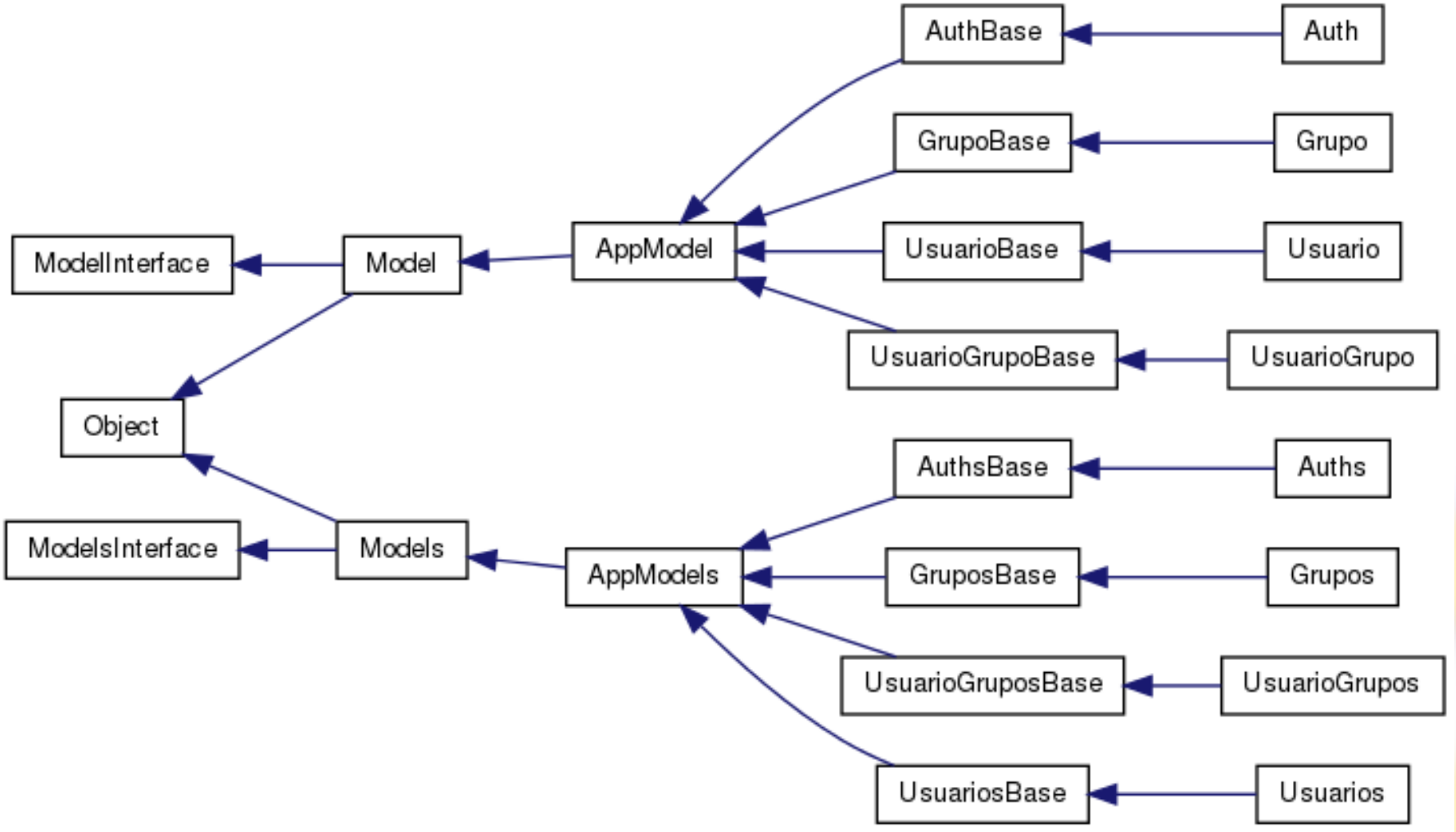
Ejemplo: shell codeGenerator

- Queremos crear el mantenedor (que use MVC) para una o más tablas de la base de datos.
- El comando de shell codeGenerator permite generar código que en la mayoría de los casos servirá "*out of the box" para mantenedores*.
- En realidad se genera muy poco código, mucho se autodetecta a partir de los datos de las columnas de la tabla asociada al modelo en la BD.



** Imagen no representa los nombres reales de las clases, pero la idea es la misma*

Ejemplo: shell codeGenerator



** Imagen no representa los nombres reales de las clases, pero la idea es la misma*

Ejemplo: shell codeGenerator

- Crearemos en el módulo Sistema el mantenedor para la tabla **parametro**, creada antes para la base de datos **demo**.
- Lo primero es crear el módulo Sistema en nuestra aplicación:

```
$ mkdir -p Module/Sistema
```

- Ahora debemos crear el menú del módulo. Esto “sobre escribirá” el menú que viene por defecto en la extensión app para el módulo (recordar pirámide).

No es necesario decir que se usará el módulo Sistema, puesto que la extensión app lo hace de forma automática.

Ejemplo: shell codeGenerator

- Creamos el archivo Module/Sistema/Config/core.php con lo siguiente:

```
\sowerphp\core\Configure::write('nav.module', array(
  '/usuarios' => array(
    'name' => 'Usuarios',
    'desc' => 'Mantenedor de usuarios y grupos del sistema',
    'imag' => '/sistema/usuarios/img/icons/48x48/grupo.png',
  ),
  '/parametros/listar' => array(
    'name' => 'Parámetros',
    'desc' => 'Mantenedor de parámetros',
    'imag' => '/img/icons/48x48/configuracion.png',
  ),
));
```

Ejemplo: shell codeGenerator

- Ahora generamos el código utilizando la shell:

```
$ ./Shell/shell.php codeGenerator
```

- Preguntará por:
  - Base de datos (si hay más de una)
  - Tablas a las que generar código (o todas).
  - Directorio donde generar (principal, módulo o submódulos).

Ejemplo: shell codeGenerator

```
delaf@trunks:~/www/localhost/dev/pages/example$ ./Shell/shell.php codeGenerator
Seleccionando base de datos para generar código
Tablas disponibles: auth, grupo, parametro, usuario, usuario_grupo
Ingresar las tablas que desea procesar [*]: parametro
Recuperando información de las tablas 100%
Directorios de destino disponibles:
1.- Directorio base de la aplicación
2.- Módulo Sistema
Seleccionar un directorio para guardar archivos generados: 2
Generando base para modelos
Generando modelos
Generando base para controladores
Generando controladores
Generando vistas (en directorio tmp/View)
```

Imagen referencial, ahora solo se generan modelos y controladores

Ejemplo: shell codeGenerator

- Al ingresar al módulo sistema ahora existirán dos iconos disponibles: usuarios y parámetros.

Ejemplo: shell codeGenerator

Listado de Parametros

1

Modulo	Parametro	Valor	Acciones
mod1	par1	val1	
mod1	par2	val2	
mod2	par1	val1	
mod2	par2	val2	
mod3	par1	val1	

Mostrar todos los registros (sin paginar)

Si se usan comentarios en las tablas y columnas de las mismas se usan en los mantenedores (en este caso no se usaron).

Ejemplo: shell codeGenerator

Crear Parametro

Modulo

Parametro

Valor

Guardar

Editar Parametro

Modulo mod1

Parametro par2

Valor val2

Guardar

Si se usan comentarios en las tablas y columnas de las mismas se usan en los mantenedores (en este caso no se usaron).

Ejemplo: shell codeGenerator

1

Modulo	Parametro	Valor	Acciones
mod2			
mod2	par1	val1	
mod2	par2	val2	

1

Modulo	Parametro	Valor	Acciones
mod3	par1	val1	
mod2	par2	val2	
mod2	par1	val1	
mod1	par2	val2	
mod1	par1	val1	

Mostrar todos los registros (sin paginar)

Ejemplo: documentación

- Ahora que existen clases, será interesante generar su documentación. Para esto el directorio principal contiene un archivo Doxyfile ya preconfigurado.
- Nos situamos en la raíz de la aplicación y ejecutamos*:

```
$ doxygen
```

- Ahora a través de example.com/doc/html tendremos disponible la documentación generada.

** Doxygen es un paquete aparte que debe ser instalado si se desea generar la documentación mediante este.*

Extensión general (más)

- Clase Utility_Spreadsheet::read(): permite leer una planilla de cálculo (CSV, ODS, XLS, XLSX, XLSM).
- Clase Utility_File: permite realizar operaciones con archivos (tamaño, subir, extraer, descomprimir ZIP y empaquetar con TGZ).
- Clase Utility_Image: trabajo con imágenes (detectar rostros si existe face_detect, miniaturas, rotar).
- Módulo Multimedia: permite generar una galería de imágenes.

Tareas pendientes/futuras

- Completar soporte de base de datos.
- Listado de TODOs, ejemplo:
  - ¿Métodos vacíos?.
  - Depurar código (reemplezar código por más eficiente).
  - Solución problema de caché (por entregar archivos con PHP).
- Acercarse lo más posible a estándares (como PSR).
- Desarrollo de nuevas funcionalidades, ejemplos:
  - Módulo de administración de Bind10 (iniciado, pero incompleto).
  - Factura electrónica para extensión empresa.
  - Implementación de webservices usando REST.
  - Implementación de llamadas a código en Java.

Enlaces de interés

- Página web del proyecto:
  - http://sowerphp.org
- Proyecto en GitHUB:
  - https://github.com/SowerPHP
- Página del ejemplo de la presentación:
  - http://demo.sowerphp.org
- Proyecto (código) del ejemplo de la presentación:
  - http://sowerphp.org/archivos/demo.tar.gz