# Documento Explicativo JAMES

# 1. Introducción al Sistema

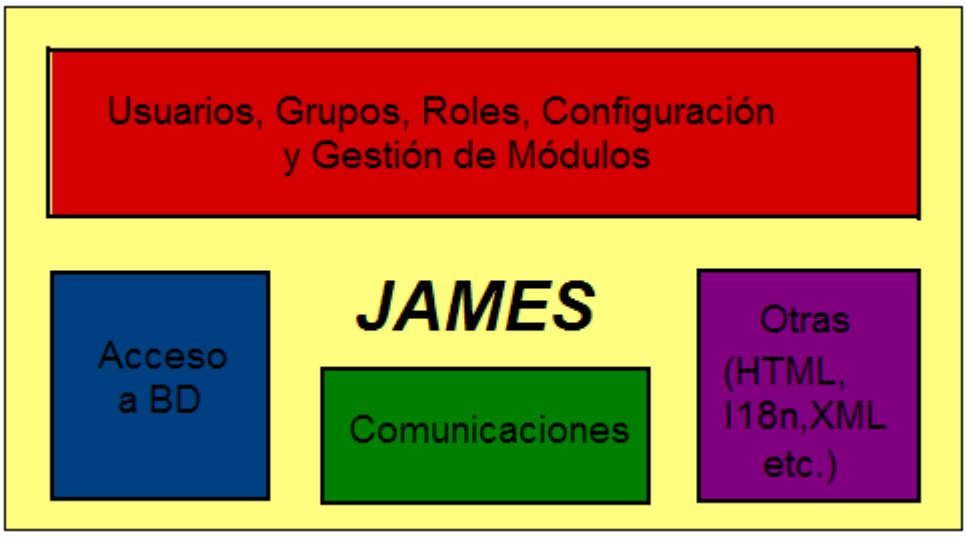
JAMES presenta un conjunto de funcionalidades, que describimos a continuación:

- Autentificación de usuarios
- Gestión de usuarios, permisos y grupos
- Opciones de configuración para los módulos
- Acceso a base de datos
- Comunicaciones con otros módulos
- Ayuda para la internacionalización (i18n)
- Clases de ayuda para generar HTML

Usar un *framework* como JAMES permite olvidarse de todo lo anterior y centrarse en la funcionalidad que queremos aportar a la hora de realizar una aplicación web.
Está diseñado para proporcionar un entorno modular, extensible y adaptable y facilitar el manejo y compartición de la información.

*Entorno*: Apache (en teoría), PHP, MySQL (por defecto).

A modo ilustrativo presentamos el siguiente cuadro:




Explicación de las funcionalidades anteriores, a modo de resumen:

**1.1 Gestión de Usuarios**:

Se encarga de todas las funciones y tareas relacionadas con los usuarios del sistema.

**1.2 Gestión de Grupos**:

Los grupos, como veremos en ejemplos más adelante, tienen asociados módulos.

**1.3 Gestión de Roles**:

Un rol tiene asociado un conjunto de permisos.

**1.4 Gestión de Permisos**:

Los permisos los define cada módulo.

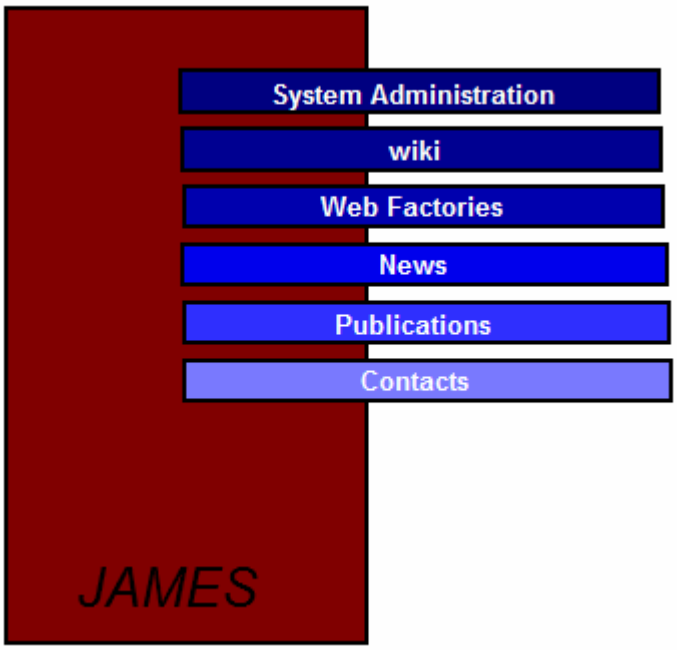
**1.5 Gestión de Módulos**:

Los módulos son cada aplicación independiente.

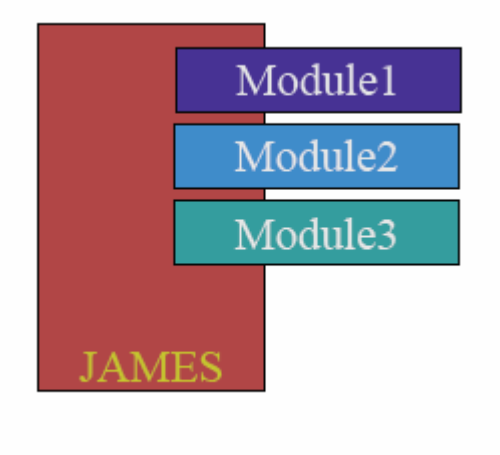
Estructura básica de JAMES:

*JAMES = núcleo + módulos*

Estos son algunos de los módulos que actualmente se encuentran en el sistema:

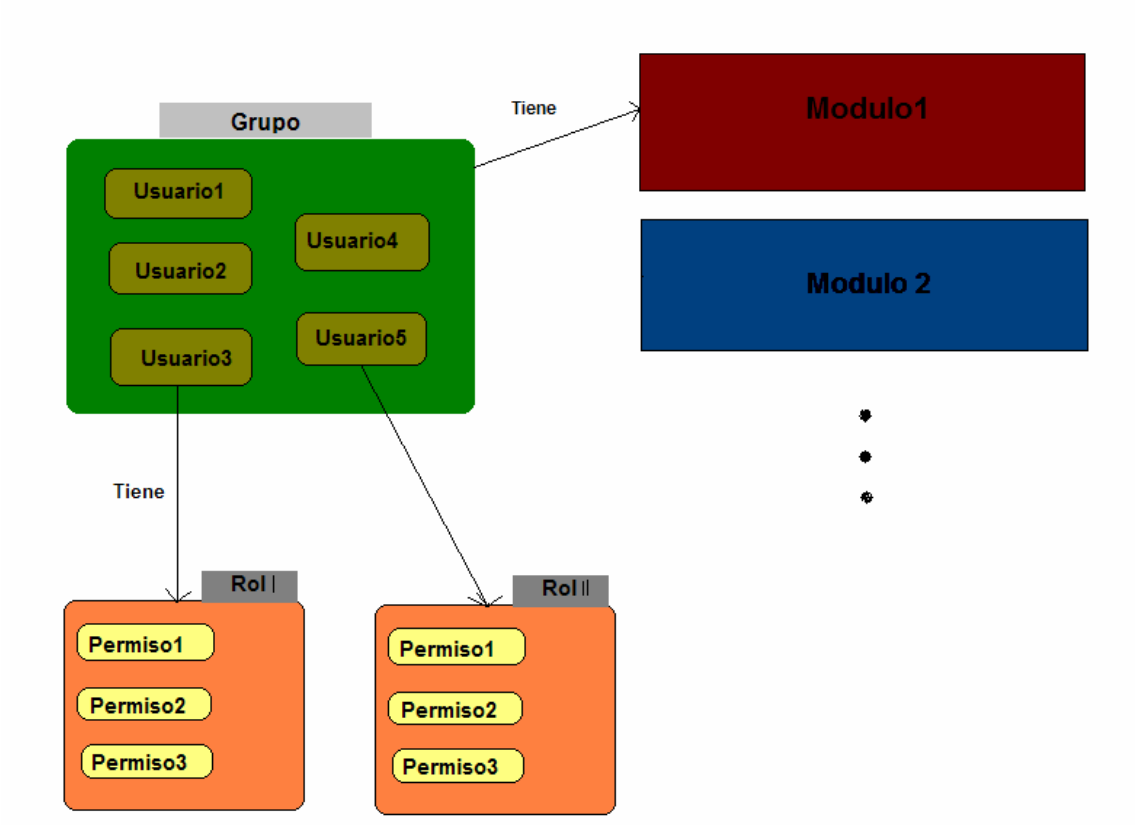



De igual modo, si se desean añadir nuevos módulos (Módule1, Módule2,…etc) a la aplicación, éstos también serán integrados como los anteriores:




Con respecto a la relación, Usuarios/Grupos/Roles/Permisos/Módulos, mostraremos un resumen y un esquema para aclarar aún más estos conceptos.

En conclusión, un usuario está asociado a grupos y en cada grupo tiene un rol (con el rol del usuario en un grupo se definen los permisos que tendrá ese usuario en cada módulo/aplicación del grupo).



## 1.6 Comunicación entre módulos

JAMES permite la comunicación módulo a módulo (y extender las capacidades básicas del sistema) para ello cuenta con una API basada en eventos.
Y, ¿Como se invoca un evento? Pues con una función llamada **ThrowEvent** a la que se le pasa como parámetro el nombre de la función que se quiere llamar, el módulo a la que pertenece y los parámetros que requiera.
Por ejemplo:
Para poder llamar a ThrowEvent tenemos que añadir primero el include de su clase... Esto se traduce en que en el fichero que se vaya a invocar se tiene que poner así: **include_once("core/comms/ThrowEvent.inc.php");**
...
y dentro de tu función:
**$aData = ThrowEvent("getUserXFields","UserAdmExt",$idU);**

donde *$aData* será el array donde se devolveran los valores de la llamada, *getUserXFields* es el nombre de la función que se quiere llamar del módulo, *UserAdmExt* es el nombre del modulo al que pertenece dicha función e *$idU* es el valor que pasas como parámetro (en este ejemplo el usuario).

## 1.7 Abstracción Base de Datos

Utilizaremos la clase *DBJames*, que nos servirá como capa de abstracción para las funciones de acceso a bases de datos de *PHP*, obteniendo de este modo, un acceso a datos centralizado e independencia con la Base de datos:

```
dbObj = new DBJames();
$str = "DELETE FROM Personas WHERE ";
$str .= "nombre = 'pepe'";
$dbObj->Execute($str);
```

## 2. Ejemplos -Vistas del Portal

A partir de ahora se van a mostrar ejemplos de lo explicado en el punto anterior. Usaremos el portal **JSWEB** desarrollado con JAMES, pero tiene que quedar claro que JAMES puede ser utilizado para cualquier entorno que necesite una herramienta corporativa. Ejemplos del Uso de JAMES son los portales de Delegación de Alumnos, Imaginática, portales para Congresos Europeos de Investigación, el portal del grupo de Investigación ISA, etc.

*¡Atención*!: En James, siempre estas logado, es decir, cuando accedes a la aplicación sin logarte, automáticamente te autentica como usuario anónimo.

La siguiente imagen muestra la página inicial de acceso a la aplicación:


Si no nos logamos como administrador del sistema nos aparecerán disponibles las siguientes funciones:

Como se puede observar en los laterales superiores de la imagen aparecen nuevas funciones que con el perfil de anónimo no teníamos, pasemos a explicarlas más a fondo:

A la derecha, aparecen en un recuadro swsseville, Portal Público e ISA, éstos son grupos creados en el sistema, que como vimos tienen asociados módulos.

Si accedemos a Portal Público nos aparecen todos los módulos a los que está asociado dicho grupo: Portal Público, Software, Mi publicación…


Al pinchar sobre un módulo, el usuario accede a ese módulo con los permisos que tenga el rol con el que esté asociado al grupo que pertenece dicho módulo.

En el otro lateral, el izquierdo, aparece una columna que contiene los módulos del grupo que el usuario tiene asignado como Personal (a lo mejor en algún caso lo llaman Global Group en vez de Personal Group). Este grupo es "especial" por diferentes motivos, pero para lo que nos atañe ahora, lo único que nos afecta es:

1. Cada usuario siempre está asociado al menos a un grupo llamado Personal Group.
2. El Personal Group y sus módulos aparecen a la izquierda en vez de a la derecha.

El módulo Users Administration lo vamos a obviar porque en un futuro va a ser eliminado debido a que el módulo Users Administration Ext es una extensión del anterior actualizada.

**Users Administration Ext**:

Este módulo se gestiona las siguientes funciones:

***1)*** *Creación* de nuevos usuarios.

**2)** *Modificación* del password del usuario.

**3)** *Asignación* de grupos y el rol asociado al grupo (incluyendo el Personal Group).

**4)** *Eliminación* de usuarios.

El módulo solo tiene un permiso llamado "System Administration". Si un usuario tiene ese permiso (mediante el rol que tenga en el grupo que tenga el módulo) tiene acceso a todo. Si no lo tiene no puede hacer a nada.

**5)** Permiso para modificar los *datos personales* ("Personal Data Administration")
Si un usuario tiene dicho permiso (“Personal Data Administration”), aunque no tenga el de "System Administration", cuando acceda al módulo verá un formulario para modificar sus propios datos que serán: el password y los campos que se hayan añadido mediante la característica que se describe a continuación:

**6)** Gestión de *Campos personalizables*
Un usuario con permiso "System Administration" puede crear nuevos campos para los usuarios más allá del password. La ventana de administración de campos permite añadir nuevos campos (de tipo text o bool, la diferencia será que a la hora de editar los valores para dichos campos aparecerá un input de tipo texto o un checkbox). El valor para estos campos se podrá editar en la ventana de modificación de datos de cada usuario (ya sea accediendo como administrador a los datos de un usuario o cada usuario a sus datos). Cada campo tiene varios parámetros:

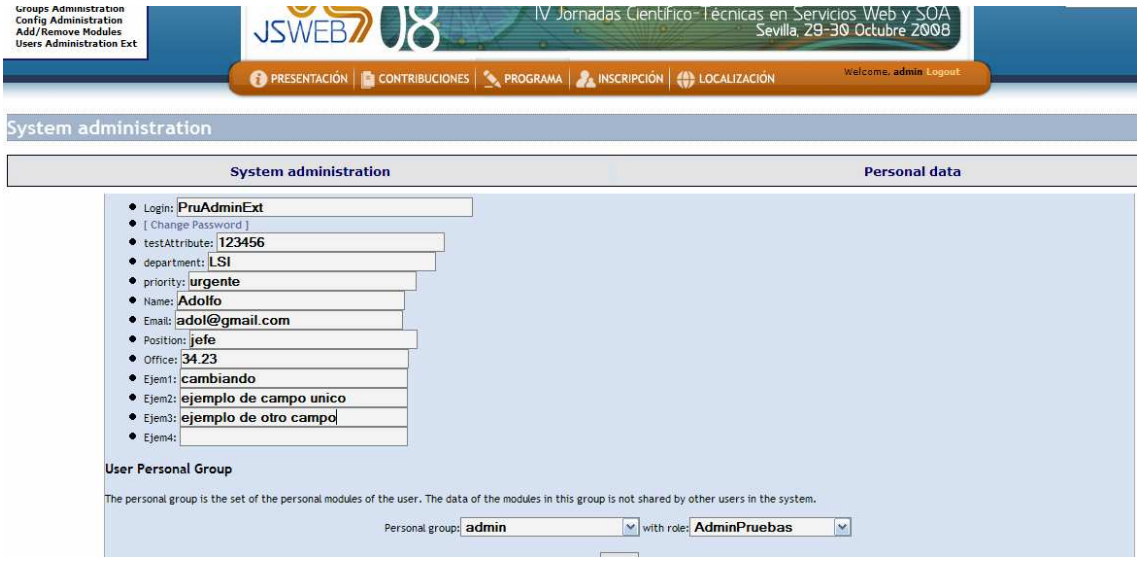
*Friendlyname*, que es el nombre con el que va a aparecer en la ventana de modificación.
*Description*.
Mask, esto vamos a obviarlo.
Cuatro checkbox: R, U, S, H. Son las siglas de "Required", "Unique", "System" y "Hidden" y significan:

- Required: Cuando Aparece un input con borde rojo para avisar de que es un campo requerido y no se guardarán los valores que se pongan al editar si no está relleno.
- Unique: Esto quiere decir que el valor que se ponga para un usuario es UNICO.
- System: Esto quiere decir que no se puede modificar a menos que el usuario tenga permiso de "System Administration".
- Hidden: Esto quiere decir que un usuario no puede ver el valor para este campo a menos que tenga permiso de "System Administration".

**7)** Permiso de *Administración* de *Grupo* (Group Administration es el nombre del permiso, no confundir con el modulo Groups Administration)
Si el módulo está asociado a un grupo "normal" (NO personal) y un usuario de ese grupo tiene el permiso Group Administration, podrá gestionar los usuarios asociados a ese grupo y sus roles. Si el grupo es un Personal Group este permiso no tiene ningún efecto.

Esta es una de las vistas del módulo que se acaba de explicar, en concreto la sección System Administration, donde aparecen todos los usuarios del sistema:

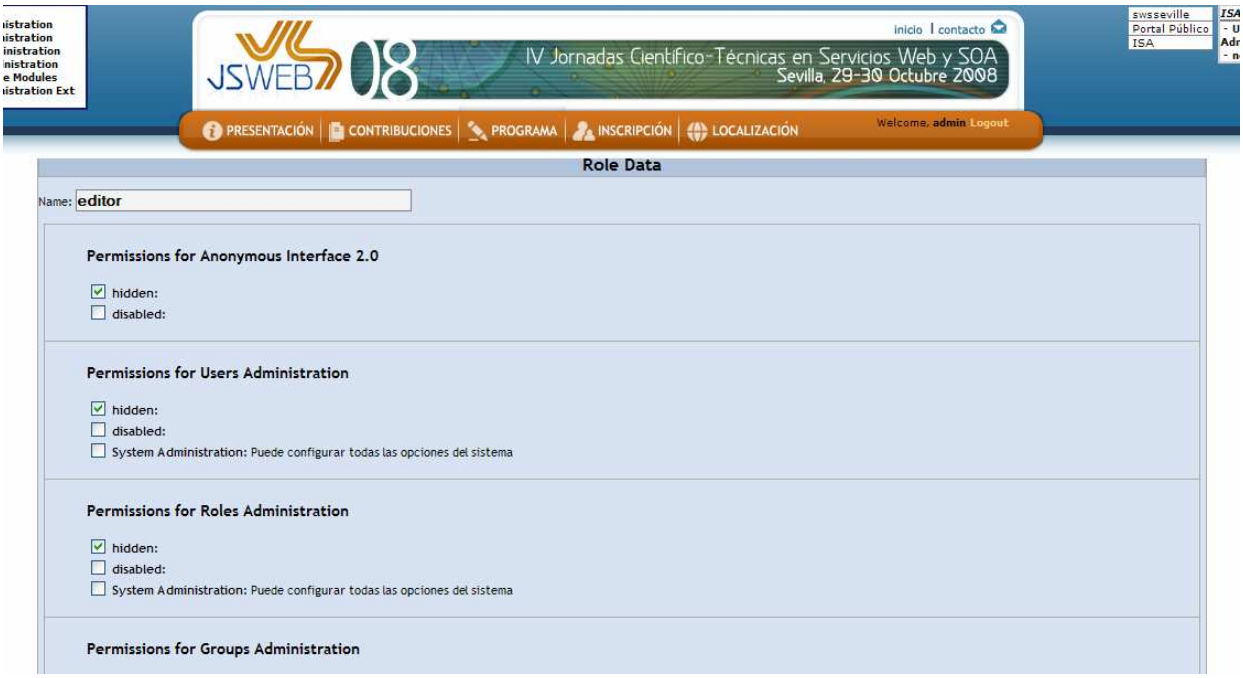

Si pinchamos en Add new user, como su titulo indica añadiremos un nuevo usuario:



Si, por el contrario, pinchamos en Manage Fields, nos aparecerá una interfaz donde podremos gestionar los *Campos personalizables* explicamos anteriormente:


**Módulo Roles Administration:**

Este módulo se encarga de crear nuevos roles, modificarlos y eliminarlos. A modo de aclaración, se muestra la siguiente imagen donde se puede ver como se asignan permisos a un rol en concreto.



El resto de módulos son obvios por su propio nombre, así que no vamos a entrar en más detalle. En el apartado *Instalación de un módulo* haremos referencia al módulo Add/Remove Modules.

# 3. Instalación de un módulo en JAMES

## 3.1 API JAMES

Vayamos al tema que nos concierne, la **creación** de un módulo en JAMES.

Un módulo necesita 3 archivos básicos:

- *init.php:* Página que se carga al lanzar el módulo.
- *config.xml*: Fichero de configuración con información sobre el módulo: nombre, descripción, permisos, etc.
- *MiModulo.class.php:* Clase utilizada para implementar la comunicación entre los módulos, donde *MiModulo* será el nombre del módulo (sensible a mayúsculas).

**Desarrollo del módulo**:

El código del módulo se desarrollará en el archivo *init.php* o en archivos incluidos por éste (código php), mientras que los valores de configuración (como los permisos) que deban ser tenidos en cuenta en el sistema hay que incluirlos en el archivo de configuración (config.xml).

***Usuario Activo***

Para acceder al usuario activo del sistema se hace mediante:

*$GLOBALS[CURRENT_USER]*

Esto devolverá un objeto de la clase *User* representando al usuario activo en el sistema en ese momento.
Algunas funciones de la clase *User* son:

*$GLOBALS[CURRENT_USER]->GetLogin();* // Devuelve el nombre
*$GLOBALS[CURRENT_USER]->GetId();* // Devuelve el Identificador

Ejemplo:

```
<?php
echo “El login es: ”.$GLOBALS[CURRENT_USER]->GetLogin();
echo ”<br>”;
echo “El id es: “ . $GLOBALS[CURRENT_USER]->GetId();
```

```
echo ”<br>”;
?>
```

***Grupo Actual***

La forma de obtener el grupo actual es a través de otra variable del sistema:

*$GLOBALS[CURRENT_GROUP]*

Esto nos devolverá un objeto de la clase *Group*. Algunos métodos de los objetos de esta clase son:

*$GLOBALS[CURRENT_GROUP]->GetName();* // Devuelve el nombre
*$GLOBALS[CURRENT_GROUP]->GetId();* // Devuelve el Identificador

Ejemplo:

```
<?php

echo “El grupo es” . $GLOBALS[CURRENT_GROUP]->GetName();
echo ”<br>”;
echo “El id es: “ . $GLOBALS[CURRENT_GROUP]->GetId();
echo ”<br>”;

?>
```

***Permisos***

Los permisos para que puedan configurarse en el sistema utilizando los roles deben haber sido primero definidos en el archivo de configuración, *config.xml*, antes de instalar el módulo. Veamos un ejemplo de como añadir un permiso a dicho archivo:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<module name="MiModulo">
        <screenname lang="es">Mi Modulo</screenname>
                <permissions>
                        <permission name="miPermiso">
                                <screenname lang="es">Mi permiso</screenname>
                                <description lang="es">
                                        Permiso de prueba para MiModulo
                                </description>
                        </permission>
                </permissions>
</module>
```

*Comentario:*

La etiqueta <*permissions*> indica que vamos a comenzar a definir permisos (uno o varios). Para cada permiso hay que definir como mínimo el nombre interno: <*permission name="miPermiso"*> que será el que se utilice en el desarrollo del módulo.
La etiqueta *screenname* indica el nombre 'amigable' que se mostrará en pantalla y la etiqueta *description*, la descripción que aparecerá en pantalla explicando la utilidad de dicho permiso.

## Consulta de permisos

Para consultar los permisos de un usuario en el código de un módulo utilizaremos la función:
*HasPermission* de la clase *User*. Un ejemplo de uso es el siguiente:

```
<?php

if ($GLOBALS[CURRENT_USER]->HasPermission(“miPermiso”))

        echo “El usuario tiene el permiso 'miPermiso'”;
else
        echo “No tienes el permiso 'miPermiso'”;
?>
```

donde *miPermiso* en la llamada a la función HasPermission será el nombre que tenga el permiso en el archivo *config.xml* (sensible a mayúsculas y minúsculas).

## *Opciones de Configuración*

Un Módulo puede utilizar variables de configuración (por ejemplo, el tamaño máximo de los archivos que se pueden subir a un repositorio). Estas variables se definirán en el archivo *config.xml* (antes de instalar el módulo), se modificarán en el módulo de *Configuración* y se podrán consultar desde el código del módulo.

### Definición de Opciones de Configuración en *config.xml*

```
<module name=“MiModulo”>
        ...
        <config>
        <!-- Primero definiremos una clave de configuración para un valor entero -->

                <configkey name="size" type="integer" level="system">

                        <!-- name=“size“ será el nombre que utilice en el sistema,
                        type=“integer“ indica que es de tipo entero (otros tipos serán
                        “boolean“, “option“ o “string“) y level=“system“ que será una
```

```
                variable configurable para todo el sistema (podría serlo para
                “group“ o “user“) -->

                <screenname lang="es">Tamaño</screenname>
                <description lang=”es”>
                        Tamaño máximo asignado por usuario
                </description>
        </configkey>
        <!-- Ahora vamos a definir una clave de configuración con una lista de
        valores opcionales -->

        <configkey name="format" type="option" level="system">
                <screenname lang="es">Formato</screenname>
                <description lang=”es”>Formato de  envío</description>
                <options>
                        <option name=”text”>
                                <screenname lang=”es”>Texto</screenname>
                        </option>
                        <option name=”html”>
                                <screenname lang=”es”>HTML</screenname>
                        </option>
                </options>
        </configkey>
    </config>
    ...
</module>
```

**Consulta de Valores de Configuración en el Módulo**

Para consultar los valores de configuración utilizaremos la función:

*GetConfig(nombre, valor_por_defecto)* de la clase Module donde *nombre* será el nombre que se le haya dado a la clave en el archivo config.xml (sensible a mayúsculas y minúsculas) y *valor_por_defecto* será un valor que pasemos por defecto a la función por si no se le ha asignado aún un valor a dicha clave en el sistema.

Ejemplo:

```
<?php
    echo “El tamaño asignado es: ”;
    echo $GLOBALS[CURRENT_MODULE]->GetConfig(“size”, 5000);
?>
```

***Obtener Información de otros Usuarios/Grupos distintos al Actual***

Utilizaremos:

**$GLOBALS[JUGRSYSTEM]**

Ejemplos:

```
// Obtener un usuario
$user = $GLOBALS[JUGRSYSTEM]->GetUser("admin");
$user->GetId();

// Obtener un grupo
$group = $GLOBALS[JUGRSYSTEM]->GetGroup(1);
$group->GetName();
```

***Acceso a Base de Datos***

**Funciones y Atributos**

• *Query(miConsulta):* Realiza la consulta miConsulta sobre la base de datos y se posiciona sobre el primer registro.
• *row[miCampo]:* Accede al campo miCampo del registro actual.
• *Next():* Accede al siguiente registro.
• *EOF:* Devuelve un booleano indicando si hemos sobrepasado el último registro.

Ejemplo típico:

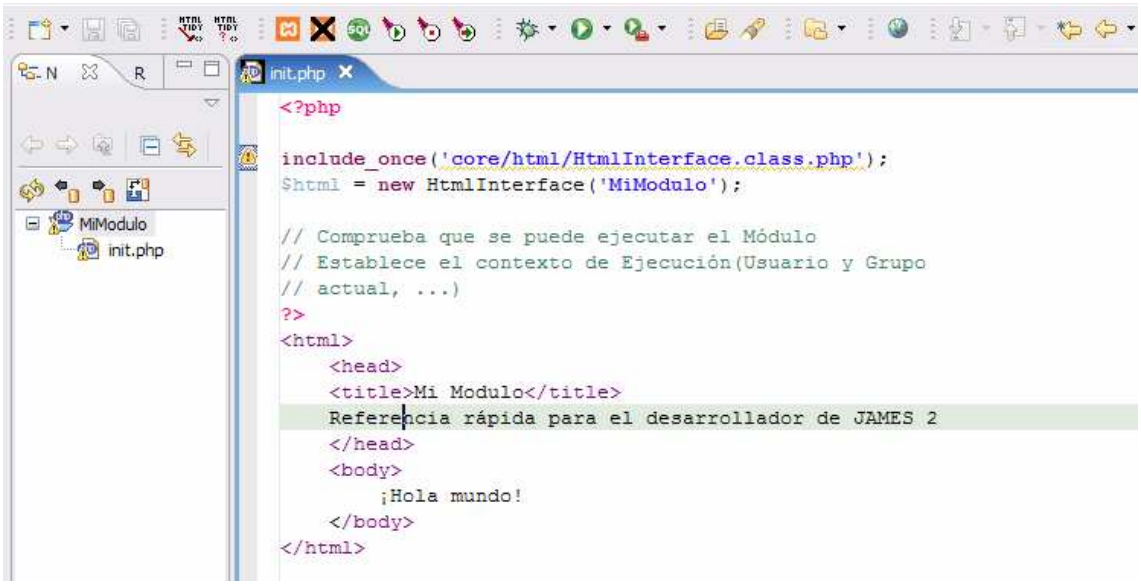
```
...
$dbObj = new DBJames();
$dbObj->Query("SELECT * FROM ModNews ORDER BY date DESC where
idG=2");
while(!$dbObj->EOF)
        {
                echo "<h1>";
                echo $dbObj->row[ "title" ];
                echo "<h1>";
                echo "<hr>";
                echo $dbObj->row[ "body" ];
                echo "<br>";
                $dbObj->Next();
        }
...
```

## 3.2 Instalación de un módulo

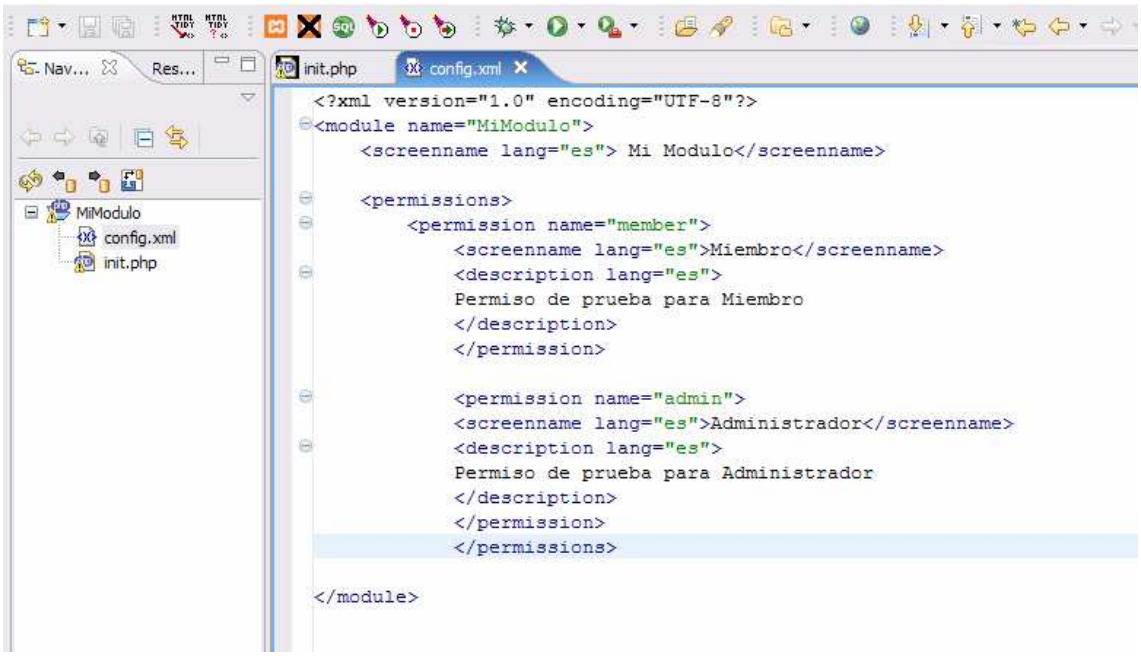
Para terminar de aclarar los conceptos explicados, vamos a poner un ejemplo básico ilustrativo de la creación/instalación de un módulo con JAMES.

Tenemos que crear los tres archivos básicos necesarios.

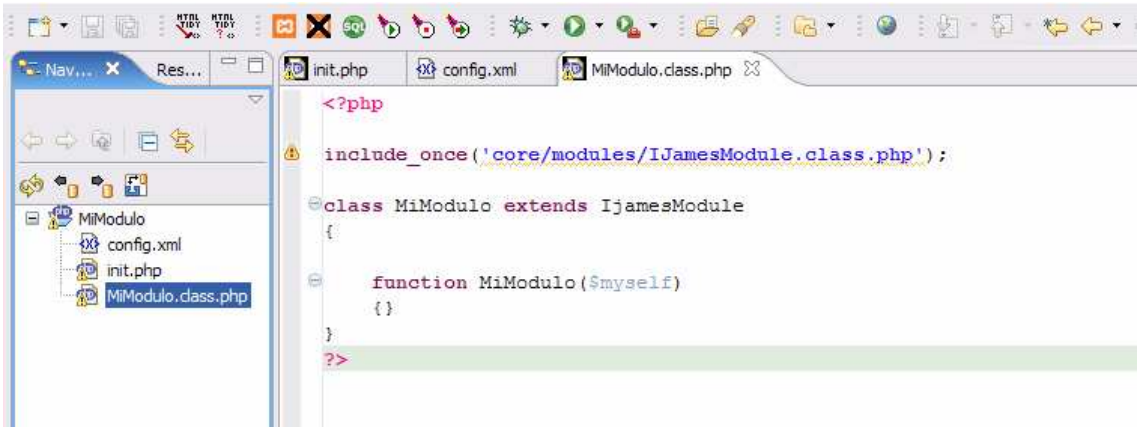
Creamos init.php:




Creamos config.xml:




Creamos MiModulo.class.php:




*Importante*: El nombre de la **carpeta** del modulo, el nombre de la **Clase** utilizada para implementar la comunicación entre los módulos y el nombre del módulo en el fichero **config**.**xml** (<module name="MiModulo">) deben ser el mismo, en nuestro ejemplo seria "MiModulo".

Una vez creada esta estructura básica de un módulo podemos proceder a explicar como sería su instalación.
Todos los ficheros que forman parte del módulo se empaquetan en un archivo comprimido que debe tener el mismo nombre que el módulo que contienen. Cualquier otro archivo que se desee incluir, siempre debe ir dentro del directorio /MiModulo. En el caso de este ejemplo, sería *MiModulo.zip*.

Los módulos se instalan el sistema utilizando el *Módulo de Instalación/Elminación de módulos*. En dicho módulo hay que indicarle el archivo *MiModulo.zip* y seguir los pasos de la instalación.

En el portal que estamos utilizando seria asi:

Entramos en el modulo Add/Remove Modules.


Esto registrará el módulo en el sistema. A partir de este momento ya se puede añadir el módulo a cualquier grupo del sistema y podremos ponerlo en marcha.