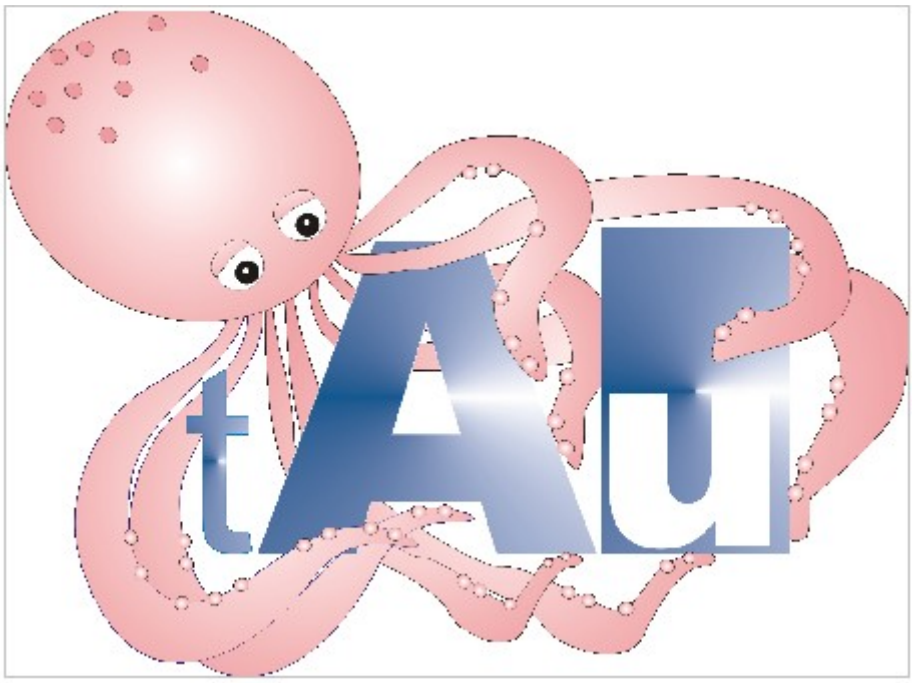
http://tauproject.sourceforge.net

MANUAL DE ARQUITECTURA Y PROGRAMACION



Proyecto TAU

06 de diciembre de 2008

Versión 1.0 del 06/12/2008

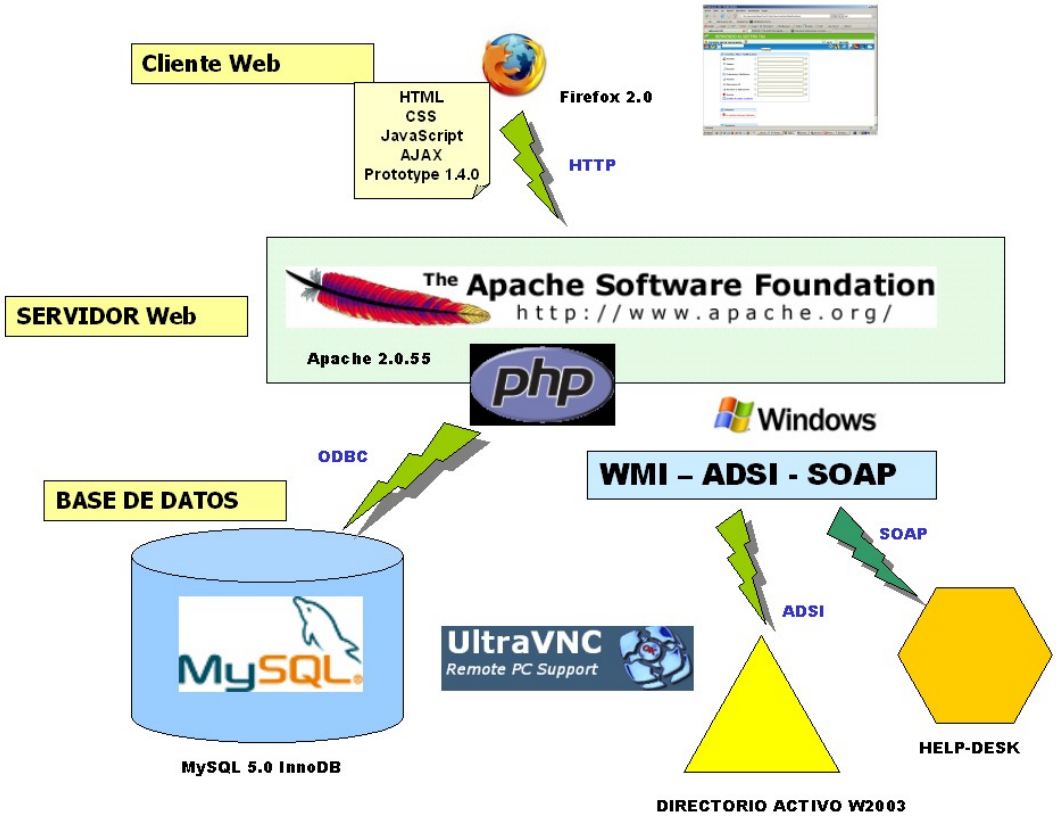
1.OBJETO

El presente documento, tiene como objeto, describir de una forma breve el entorno tecnológico de la aplicación TAU, el software en el que se basa, la arquitectura externa, así como las grandes líneas de codificación interna. No pretende constituir una herramienta de programación exhaustiva.

1.ARQUITECTURA EXTERNA

El proyecto **TAU** se compone de una aplicación de 3 niveles:

- Cliente WEB
- Servidor WEB
- Base de datos



Los **clientes** están basados en **navegadores de internet**, con el fin de que no se precise de instalación, y actualizaciones en el puesto de trabajo final (ya que casi todos los Sistemas Operativos traen un navegador web de serie). Además, la interfaz web mostrada es **ligera** ya que se basa en **HTML** y **JavaScript**, sin ser preciso tecnología más pesadas como **Flash**, o **Java**.(en el caso de algunos módulos nuevos es preciso instalar el JRE de Java) Para ofrecer en una interfaz web acciones semejantes a una aplicación de escritorio, se hace uso de la tecnología **AJAX** a través de la librería **Prototype 1.4.0**. El único cliente web validado en la aplicación es **Firefox** aunque se ha comprobado el funcionamiento con otros navegadores

con motor interno **Mozilla**.

El **Servidor WEB** es el encargado de atender las peticiones de los clientes, y actuar como intermediario con otros sistemas como la base de datos, el directorio activo, la interfaz WMI, etc. Como servidor WEB se ha utilizado el servidor **Apache 2.2.9**, dada su probada estabilidad y rendimiento, así como la integración con módulos como PERL, PHP, autenticación, proxy, etc. El servidor web ofrece su contenido a través de páginas generadas mediante **scripting PHP** en el lado del servidor. La utilización de PHP respecto a otras tecnologías fue debida principalmente, a su rendimiento, economía de código (respecto a otros como java), y multitud de módulos externos (librerías matemáticas, de XML, de generación en PDF, servicios SOAP, interfaz COM, etc.). La **versión de PHP que se utiliza es la 5.2.6.** Mediante PHP se hace uso de librerías de interfaz de Windows **WMI** (para obtención de datos de equipos remotos), **ADSI** (para su integración con el directorio activo), **SOAP** (para la integración con el sistema de HELP-DESK), y con ejecución remota de comando mediante la utilidad **BeyondLogic RmtExec Server versión 2.05.** Por otra parte, para el control remoto de los clientes se utiliza la aplicación **UltraVNC v 1.0.2 modificada** El sistema operativo sobre el que va instalado el servidor WEB es **W2003 Server. NO SE HA PROBADO EL SOFTWARE SOBRE PLATAFORMAS LINUX, debido a que no se dispone de interfaz nativa ADSI o WMI que es necesario para algunos módulos**

La **Base de datos**, es el repositorio de información donde se almacena la información de los distintos módulos, pudiendo residir en un servido independiente. La conexión entre el servidor WEB y la base de datos se realiza mediante **conexiones ODBC**. El motor de base de datos está basado en **MySQL 5.0 en modo transaccional**, con el fin de poder realizar transacciones en las operaciones de los módulos, mediante el tipo de tablas **InnoDB**. El modelo de datos interno es simple, y se ha evitado el uso de vistas, o procedimientos almacenados.

Los **entes externos**, son otros componentes que forman parte del TAU tales como el directorio activo: utilizado para la autenticación, cambio de contraseñas,etc. Su acceso se realiza mediante interfaz **ADSI**. La interfaz **WMI**, que nos permite obtener distintos datos de equipos remotos tales como procesos, aplicaciones instaladas, especificación de hardware, etc. El software de control remoto **UltraVNC 1.0.2** para el control de equipos, y el software de ejecución remota **BeyondLogic RmtExec Server versión 2.05.** A parte se utiliza la tecnología **SOAP** para la integración del inventario con el Help-Desk de Unicenter.

2.ARQUITECTURA INTERNA

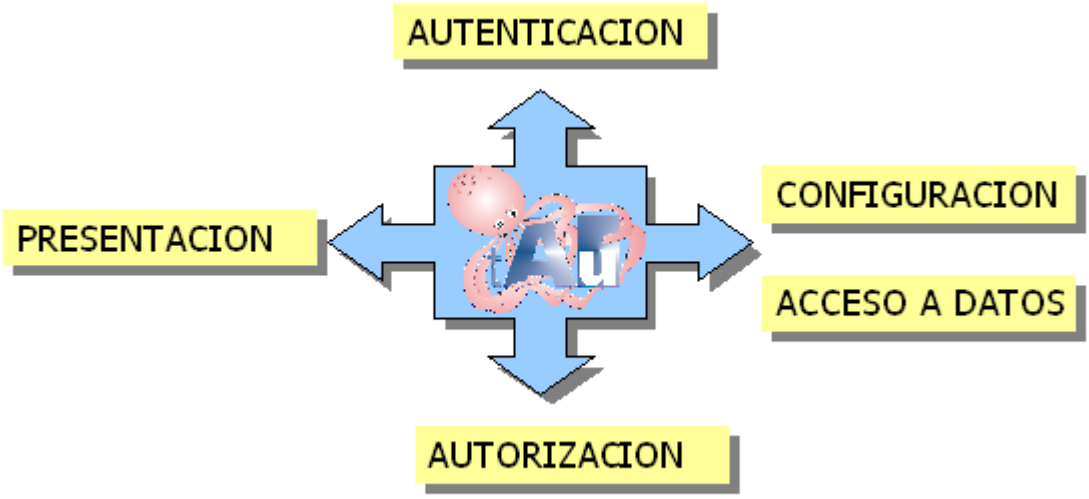
A continuación se detalla la arquitectura interna del sistema TAU, sus distintos componentes, la estructura de ficheros, y los principios de diseño

COMPONENTES DEL TAU

El TAU no es más que un **portal** de **miniaplicaciones** o también llamados **módulos**, siendo alguna de éstas, tal y como el **Inventario**, bastante complejas, y otras más sencillas, como el **Reloj**. Su misión es ofrecer una infraestructura común a todas estas mini aplicaciones. Los servicios que ofrece el portal TAU son los siguientes:

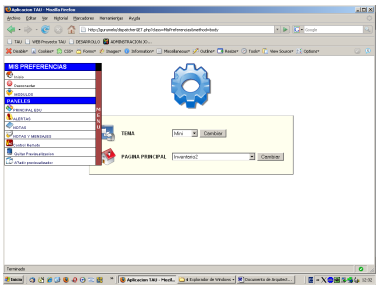
- **Autenticación**: el TAU es responsable de autenticar al usuario y verificar sus credenciales y si tiene acceso al portal
- **Autorización**: determina que miniaplicaciones, y que operaciones dentro de la miniaplicación un determinado usuario autenticado puede realizar
- **Presentación**: El TAU ofrece una interfaz uniforme de presentación de los módulos, creando áreas en pantalla predefinidas, con las que las distintas aplicaciones pueden interactuar.
- **Configuración**: Los distintos módulos o miniaplicaciones pueden registrar ciertos parámetros de configuración mediante la interfaz ofrecida por el TAU
- **Acceso a datos**: Si alguna miniaplicación precisa del acceso a fuentes de datos externas, el portal TAU, ofrece un conjunto de conexiones para ésta.

Cualquiera de estos servicios pueden ser sustituidos sin afectar al resto de componentes. Así por ejemplo el servicio de Autenticación puede realizarse pidiendo usuario y contraseña de un LDAP, o desde una base de datos.



Todo el sistema TAU está basado en **módulos**. Cada **módulo** es lo que denominamos una **miniaplicación** . Los servicios ofrecidos por el TAU (autenticación, autorización, presentación, etc.), son implementados y ofrecidos también a través de módulos. Es decir, tanto los servicios del portal TAU, como las miniaplicaciones, todos son módulos. De hecho, los servicios del TAU, al ser comunes para todas las miniaplicaciones, son considerados **módulos KERNEL**, y el resto simplemente **módulos**.

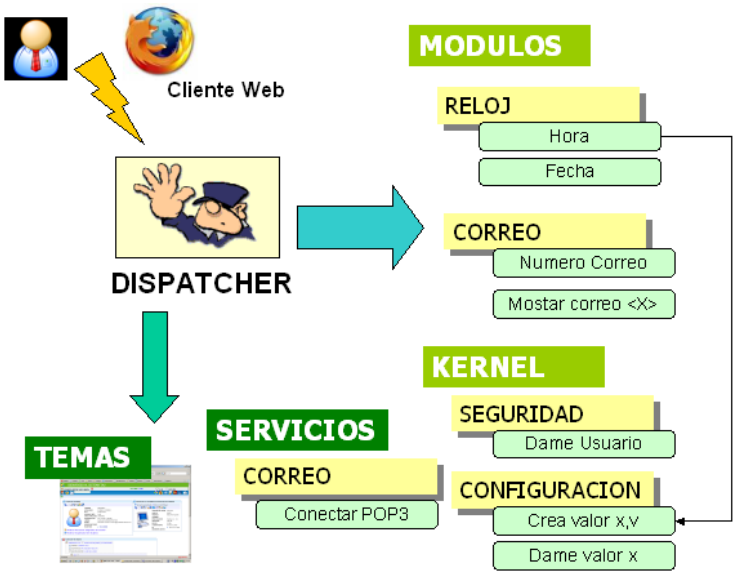
En cuanto al servicio de **Presentación**, este es ofrecido a través de **TEMAS** (que se implementan mediante un módulo especial para cada TEMA y hojas de estilo). De esta forma, la apariencia de la infraestructura del TAU puede cambiar, dependiendo del TEMA seleccionado por el usuario.



Por otra parte, existen los denominados **SERVICIOS**. Estos son utilidades, ejecutables, utilizados por un determinado módulo (de hecho, el nombre del servicio debe ser el mismo que el del módulo), pero que por su naturaleza, no debe permanecer dentro del árbol de acceso del servidor web, sino ser invocado externamente por el módulo tras una petición web.

Todo el control de acceso a las distintas operaciones de los módulos se realiza a través de un **DISPATCHER**, que es el encargado de autenticar al usuario, autorizarlo, y ofrecer la presentación según el

tema elegido por el usuario. Es decir, el **DISPATCHER**, es el '**guardia policía**' encargado de controlar el acceso, y proporcionar la respuesta devuelta por la invocación del módulo, formateada según la apariencia del **TEMA** elegido por el usuario.



El portal TAU requiere una configuración mínima global para todos los módulos, que es configurada en el fichero de configuración **config.inc.php** del directorio raiz. Los parámetros de configuración global son básicamente definición de PATH, y parámetros de conexión con la base de datos.

ESTRUCTURA DE ARCHIVOS

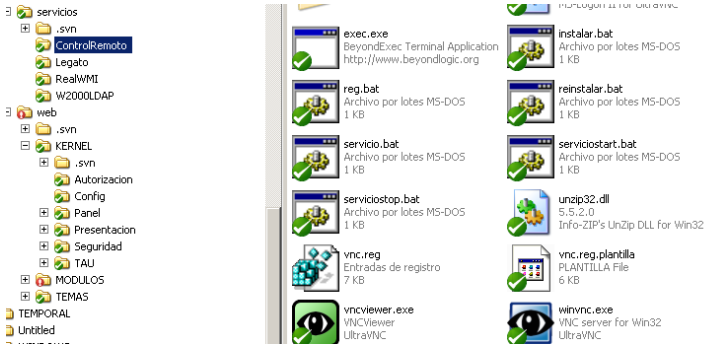
Toda este conjunto de componentes (módulos, módulos kernel, temas, servicios, dispatcher, configuración global), se organiza en la siguiente estructura de ficheros:

Toda la aplicación está contenido dentro de un directorio, que se configura mediante la variable de entorno **TAU_HOME**. (En este caso particular la carpeta es **C:\TAU**). Dentro de dicha carpeta, aparecen dos directorios:

- **Web**: que es el directorio raíz del servidor web, y por tanto, todo su contenido es accesible mediante la invocación a una URL concreta.
- **Servicios**: que es donde se encuentra los ejecutables o programas externos que, por seguridad, no deben ser accesibles mediante una URL web, sino ser invocados indirectamente por los módulos web.
- **Herramientas**: son pequeños scripts o utilidades (TAUPhone) que deben ser utilizados conjuntamente con algún módulo para ofrecer cierta funcionalidad

La parte de **SERVICIOS** se organiza en carpetas, cada una, con el nombre del **MODULO** que hace uso de las utilidades contenidas en la carpeta. Por ejemplo, en la estructura presentada, aparece la carpeta

ControlRemoto, donde se contienen los ejecutables del **UltraVNC** para poder instalarlos en la máquina remota si es preciso, tras una solicitud al método **instalar** del **MODULO ControlRemoto**



La parte **WEB** constituye todo el contenido accesible desde el servidor web, y que puede ser invocado por el cliente mediante una URL específica. Dentro de esta carpeta existen otras tres que son:

- **KERNEL**: En el aparece una carpeta por cada módulo que implementa uno de los servicios del portal TAU, tal y como Autenticación, Autorización, Presentación, Configuración, etc.
- **MODULOS**: Aparece una carpeta por cada miniaplicación que se quiera integrar en el portal TAU (por ejemplo, ControlRemoto, CorreoCorporativo, Mensajería, Help-Desk, Inventario, etc.)
- **TEMAS**: Existe una carpeta por cada aspecto diferente que se quiera proporcionar visualmente al portal TAU.

Por otra parte, en el directorio raíz **WEB** están dos de los componentes fundamentales del portal:

- **DISPATCHER**: Es el único punto de entrada a la invocación a las distintas operaciones que ofrecen las miniaplicaciones. **Es imposible** invocar la funcionalidad de una miniaplicación, sin tener que pasar por el DISPATCHER. Existe una versión para GET y otra para método POST.
- **CONFIG.INC.PHP**: Fichero de configuración de variables globales del portal TAU.

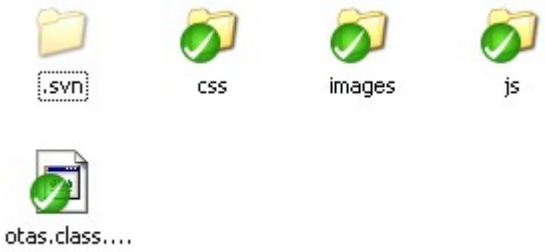
Dentro de la carpeta **TAU/WEB/KERNEL** aparece una carpeta por cada servicio ofrecido por el portal TAU:

- **ADMINISTRACION**: Módulo que nos permite gestionar los usuarios, módulos y configuración del sistema.
- **AUTORIZACIÓN**: Determina que operaciones de que módulo puede invocar un determinado usuario

- **CONFIG**: Posibilidad de que cada módulo registre variables de configuración globales, o particulares de un usuario, con persistencia o no.
- **PANEL**: Gestión de Paneles en la Presentación. (componente PREVIEW). Ver guía de uso del TAU para más información.
- **PRESENTACIÓN**: Encargado de aplicar el TEMA seleccionado por el usuario a la salida proporcionada por la invocación de un método de un módulo
- **SEGURIDAD**: implementa la autenticación del usuario.
- **TAU**: Implementa la conectividad con base de datos, y otras utilidades genéricas.

Dentro de la carpeta **TAU/WEB/MODULOS** aparece una carpeta por cada miniaplicación integrada dentro del portal TAU. La estructura de cada MODULO es la siguiente:

- **CLASE PRINCIPAL**: es un fichero con el *Nombre_de_la_clase.class.php*. Es el punto de entrada principal de acceso a las operaciones que ofrece el módulo. Al ser una clase de PHP, este no puede ser invocada directamente mediante URL, sino sólo a través del DISPATCHER.
- **CARPETA CSS**: en el se almacenan las hojas de estilo de dicho módulo, para su copia en la carpeta de TEMAS, y posterior modificación y se desea cambiar el aspecto visual.
- **CARPETA IMAGES**: Contiene las imágenes utilizadas por el módulo
- **CARPETA JS**: Contiene el código javascript utilizado por el módulo



Dentro de la carpeta **TAU/WEB/TEMAS** aparece una carpeta por cada TEMA que se puede utilizar dentro del portal TAU. La estructura de cada TEMA es la siguiente:

- **CLASE PRINCIPAL**: es un fichero con el *Nombre_de_la_clase.class.php*. Es el punto de entrada principal de acceso a las operaciones que ofrece el TEMA. Todo tema debe implementar una interfaz que unifique la forma de presentar los distintos elementos del portal TAU.

- **CARPETA CSS**: en el se almacenan la hoja de estilo del propio TEMA, además de las modificación de las hojas de estilo de cada módulo que se desee para personalizar su aspecto.
- **CARPETA IMAGES**: Contiene las imágenes utilizadas por el tema
- **CARPETA JS**: Contiene el código javascript utilizado por el tema

3.ESTRUCTURA DE CODIGO

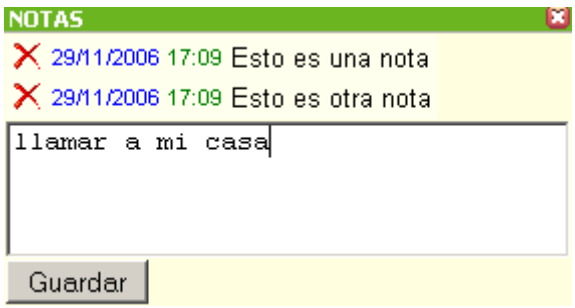
Tres son los componentes principales del TAU a nivel del código:

- **MODULOS**: Implementa la funcionalidad de una miniaplicación
- **TEMAS**: Definen el aspecto visual del portal
- **DISPATCHER**: Punto de control único de acceso a la invocación de los métodos de cada módulo

Vamos a estudiar la estructura de cada uno y como se materializa a nivel de código:

MODULOS

Son el **componente principal del TAU** (casi todo en el TAU es un módulo). Un modulo se implementa como una **clase en PHP**. Cada **método público** de la clase constituye una **operación invocable mediante URL** por el cliente, es decir, una operación. A dicha operación se le pueden pasar **parámetros** (en forma de array asociativo de forma que el número de parámetros puede ser variable). Por ejemplo, sea un módulo **Notas** que permite crear pequeñas notas para recordar:



Las operaciones de este módulo NOTAS serían:

- Ver notas
- Inserta Nota (texto_nota)
- Eliminar Nota (id_nota)

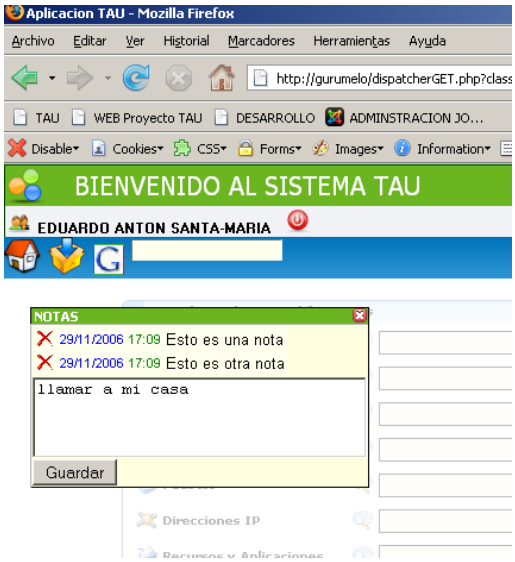
Por lo tanto un ejemplo de código de implementación del módulo **Notas** sería:

```
class Notas {
        var $name = __CLASS__;
        var $desc = "Notas personales";
        private $vista = "";

        // Muetra las notas
        public function verNotas() {
        …
        }

        public  function insertaNota( $param ) {
        …
        }

        public function eliminarNota ( $param ) {
        …
        }
}
```

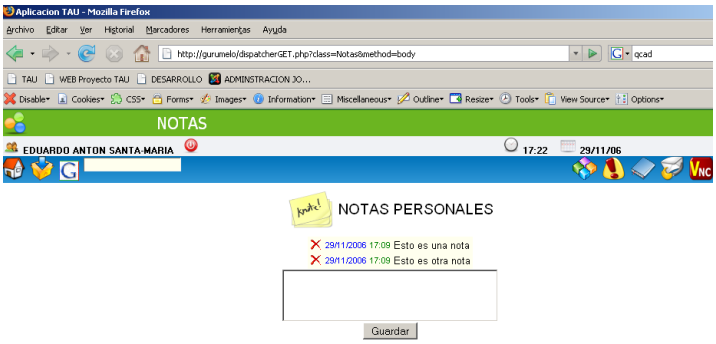
Por otra parte, tal y como está pensado el portal TAU, toda aplicación debe ofrecer al menos dos vistas:

- Vista en panel, llamada **PREVIEW**
- Vista en modo normal, y entrada principal, llamado **BODY**

La vista **PREVIEW** permite ver la miniaplicación como una ventanita independiente en un área de paneles cuando éste se activa.



Por otra parte, la vista **BODY** hace que la aplicación que se está utilizando en el portal TAU sea ahora la principal, y esta sea su página de inicio.



Por lo tanto, en la **clase PHP**, además de los métodos de la propia clase, es preciso la implementación de los métodos **preview** (para la vista en panel), y **body** (para la vista en principal como página de entrada de inicio). Así el código quedaría como sigue:

```
class Notas {
        var $name = __CLASS__;
        var $desc = "Notas personales";
        private $vista = "";

        // Muetra las notas

        public function preview(){
        echo  $this->verNotas();
        }

        public function body(){
        echo  TAU::cabecera('NOTAS PARTICULARES');
        ...
        echo $this->verNotas();
        }

        public function verNotas() {
        …
        }

        public  function insertaNota( $param ) {
        …
        }

        public function eliminarNota ( $param ) {
        …
        }

}
```

Internamente cada **MODULO** puede utilizar las imágenes, hojas de estilo, y código javascript, que se almacena respectivamente en su subcarpeta **images**, **css**, y **js**. No es preciso que se devuelva código

HTML como salida, sino que perfectamente podría tratarse de código XML, que sirviese como entrega para un servicio SOAP.

Por otra parte, cada módulo debe poder ser configurado por el módulo de Administración. Para ello, se precisa que se implemente dos métodos más en la clase del módulo

- **Configurar**: Permite configurar el módulo mediante pares de párametro-valor
- **Configuracion**: Devuelve los valores de configuración almacenados.

Así por ejemplo, para el módulo Nagios (donde hay que indicar la URL de acceso a la instalación del Nagios), una posible implementación sería la siguiente:

```
public function Configuracion(){
        $conf = array ();
        $conf[] = array ('Nagios_Nagios-Server','Servidor Nagios','Direccion IP o nombre del servidor donde se encuentra instalado el NAGIOS', Config::get( Config::getModuleID("Nagios") ,"Nagios_Nagios-Server",0,"http://nagios@shrek/nagios/"));
        return $conf;
}

public function Configurar( $param ){
        Config::put( Config::getModuleID("Nagios") , "Nagios_Nagios-Server", $param['Nagios_Nagios-Server'], 0 );
}
```

El formato de la configuración es un array, en la que cada entrada es un array con los siguientes valores:

- Nombre único de variable de configuración. Por ejemplo: Nagios_Nagios-Server
- Nombre descriptivo de variable. Por ejemplo: Servidor Nagios
- Descripción de variables. Por ejemplo: Direcció IP o nombre del servidor donde se encuentra instalado el NAGIOS
- Valor actual de la variable.

Para el almacenamiento de la configuracíón se utiliza el módulo Config del KERNEL.

La seguridad de los módulos viene dada del hecho de que una clase en PHP no ejecuta ningún código al invocarla, sino que es preciso la instanciación de ésta, y esto solo ocurre en el DISPATCHER. Así por ejemplo, para invocar el método **eliminarNota** del módulo **NOTAS**, la URL a invocar sería:

http://servidor/dispatcherGET.php?class=NOTAS&method=eliminarNota&idnota=8

- Ver el DISPATCHER para entender el funcionamiento de la URL.

TEMAS

Son el **componente del TAU** que permiten **personalizar el aspecto visual del portal**. Es decir, permite mostrar la misma información, con otros colores, otras imágenes, e incluso con otra organización.

Tal y como se ha visto en la **Guía de uso del portal TAU**, éste se compone de varios elementos visuales:

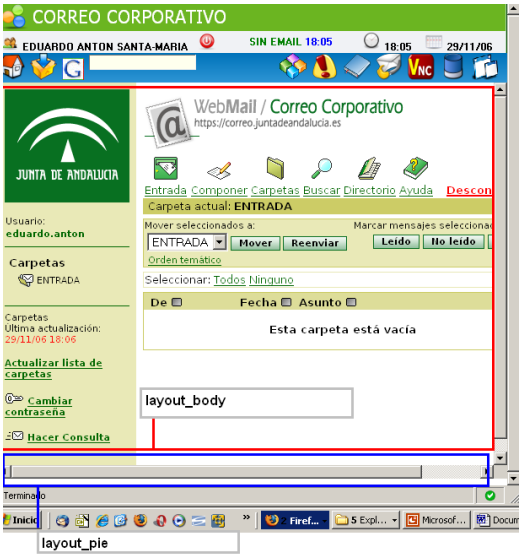
- Cabecera (con logo y título)
- Menu (botón inicio, selector de módulos, región de espera, área de paneles)
- Área de aplicación
- Pie de aplicación
- Previews

Cada área se implementa mediante **DIV HTML** con un **ID** único, según la siguiente correspondencia:

- Cabecera → layout_cabecera
  - Logo → layout_cabecera_logo
  - Titulo → layout_cabecera_titulo

- Menu → layout_menu
  - Botón inicio → layout_menu_home
  - Selector de módulos → layout_menu_selector
  - Región de espera → espera
  - Área de paneles → layout_menu_paneles
  - Paneles → panel

- Área de aplicación → layout_body
- Área fin de aplicación → layout_pie



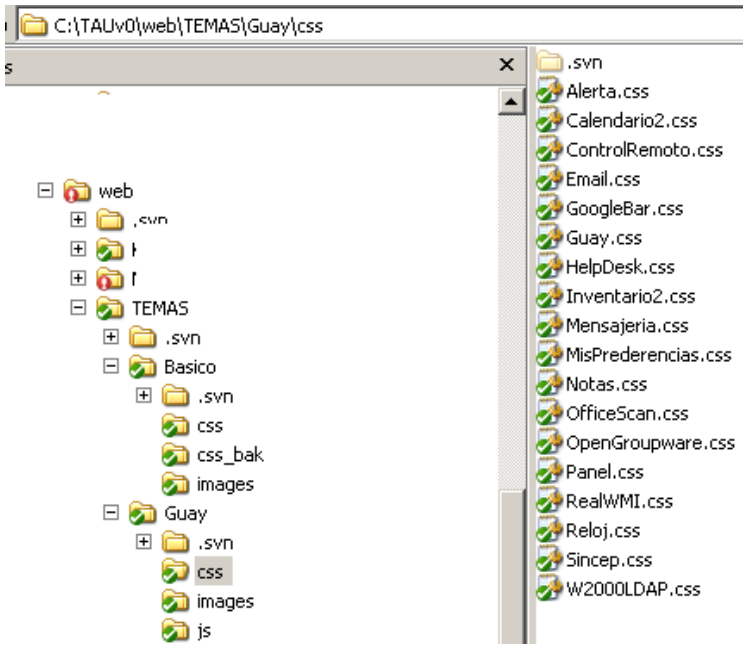
- Modulo
  - Cabecera → cabecera-MODULO
  - Cuerpo → cuerpo-MODULO

El **TEMA** se implementa como una clase PHP, que debe crear todos los DIV contenedores arriba descritos, y organizarlos y decorarlos como se desee en el TEMA. (cambiar la imagen y posición del botón inicio, hacer un área de paneles flotante, etc.). Para ello, cada TEMA debe implementar los siguientes métodos de interfaz

- **inserta_cabecera:** Se encarga de generar **layout_cabecera** y su contenido
- **Inserta_menu**: Genera el **layout_menu** con su contenido.
- **Inserta_body_pre:** Genera la decoración de marco del **layout_body**
- **Inserta_body_post:** Cierra la decoración externa del marco **layout_body**
- **Inserta_pie** Genera el **layout_pie**
- **Inserta_modulo_pre:** Genera la decoración del preview **cabecera-MODULO, cuerpo-MODULO**
- **Inserta_modulo_post:** Cierra la decoración del preview

Dichos métodos son invocados por el **módulo KERNEL PRESENTACIÓN** a medida que va construyendo la página de salida.

Internamente cada **TEMA** puede utilizar las imágenes, hojas de estilo, y código javascript, que se almacena respectivamente en su subcarpeta **images**, **css**, y **js**. El TEMA tendrá en la carpeta **css** una hoja de estilos con el mismo nombre del TEMA (que es la que sirve para dar formato a los elementos generados por el tema), así como una hoja de estilo por cada MODULO integrado en el portal. De esta manera, se puede personalizar la hoja de estilo de un determinado módulo, para este tema en concreto.


C:\TAUv0\web\TEMAS\Guay\css
web
.svn
TEMAS
.svn
Basico
.svn
css
css_bak
images
Guay
.svn
css
images
js
.svn
Alerta.css
Calendario2.css
ControlRemoto.css
Email.css
GoogleBar.css
Guay.css
HelpDesk.css
Inventario2.css
Mensajeria.css
MisPrederencias.css
Notas.css
OfficeScan.css
OpenGroupware.css
Panel.css
RealWMI.css
Reloj.css
Sincep.css
W2000LDAP.css

DISPATCHER

Es el componente de **control de acceso** a las distintas **operaciones** que ofrece cada **módulo o miniaplicación**. Es el encargado de:

- Autenticar al usuario
- Autorizar al usuario a invocar la operación sobre dicho módulo
- Instanciar la clase del módulo invocado, y llamar al método indicado para realizar la operación.
- A dar formato según el TEMA predefinido por el usuario

A.- Autenticar al usuario

Lo primero que realiza el DISPATCHER es comprobar si el usuario está autorizado a utilizar el portal TAU, es decir, si está autenticado en el sistema correctamente. Para ello, instancia al módulo **Seguridad** e invoca el método público **Comprueba()**. Este función debe devolver el **id de usuario** si la autenticación es correcta, o redirigir la página, a una de petición de credenciales en caso de fallo de autenticación.

B.- Autorizar al usuario

Tras saber el id de usuario, la miniaplicación que se desea invocar, y el método (e incluso los parámetros), éstos son pasados a una instancia del módulo **Autorización**, que en base a una matriz de comprobación, nos dirá si el usuario está autorizado o no a realizar tal operación, mediante la llamada al método **EstaAutorizado()** En caso de no estar autorizado se muestra el mensaje pertinente.

C.- Instanciar la clase del módulo invocado

Esta tarea se realiza a través del módulo **Presentación**, mediante la llamada al método **construye_pagina.** (el porqué esta tarea no es realizada directamente por el dispatcher, es debido, a que el contenido a mostrar de la aplicación invocada, va incrustada dentro de un marco de decoración, y es preciso generar dicho marco de decoración antes que la salida proporciona por el resultado de ejecutar el método del módulo). Lo que hace el dispatcher es crear una instancia del módulo invocado **$inst = new $param[‘class’]** y después ejecutar el método solicitado pasándole los parámetros **$inst->$param[‘method’]($_GET)**

D.- Dar formato según el tema seleccionado por el usuario

Esta tarea también se delega en el módulo **Presentación** .Esta instancia una clase del **TEMA** preferido del usuario, y va llamando a los métodos de interfaz **inserta_cabecera, inserta_menu, etc..**

Un esquema básico del **dispatcher** es como sigue:

```
include_once "config.inc.php";  // Inclusión de configuración global
 $auth = new Seguridad();
$usuario =  $auth->Comprueba(); // Comprueba si el usuario está autenticado correctamente
$aut = new Autorización();
$aut->EstaAurorizado( $usuario, $_GET ); // Comprueba si el usuario está autorizado
$p = new Presentación();
$p->construye_pagina();// Construye la página
```

Y parte del trabajo del dispatcher, que es realizado por la clase **Presentación**, el código someramente sería así:

```
function __construct( $disparam ){
        $this->param = $disparam;
if ( ! array_key_exists( 'layout', $disparam ) ){ // Seleccionamos el TEMA preferido del usuario
                $layout = Config::get(0, "tema.preferido", Config::getUser(), "Basico" );
        } else {
                $layout = $disparam['layout'];
        }
        if ( array_key_exists( 'ajax', $disparam ) ){
                $this->ajax = true;
        }
        include_once PATH_TEMAS . $layout . "/" . $layout . ".class.php";
        $this->layoutClass = new $layout(); // Instanciamos el TEMA
        $this->estilo = $layout;

}

public function construye_pagina(){
        //ob_start();
        if ( ! $this->ajax ){
                $this->comienzo_de_pagina();// Construimos la cabecera
                $this->layoutClass->inserta_cabecera();
                $this->layoutClass->inserta_menu();
                $this->layoutClass->inserta_body_pre();
        }
        if ( $this->param['class'] == 'Panel' ){
                include_once PATH_KERNEL . $this->param['class'] . "/" . $this->param['class'] . ".class.php";

        } else {
                include_once   PATH_MODULOS   .   $this->param['class']   .   "/"   .   $this->param['class']   .
".class.php";
        }
        $m = new $this->param['class'](); // Instanciamos la APLICACION INVOCADA

        //print_r($this->param); // Llamamos al método de  la APLICACION INVOCADA

        $m->{$this->param['method']}($this->param);

        if ( ! $this->ajax );// Construimos el pie de página
```

```
                $this->layoutClass->inserta_body_post();
                $this->layoutClass->inserta_pie();
                $this->fin_de_pagina();
            }
            ob_end_flush();
    }
```

Como se habrá podido observar la **Presentación** se genera condicionalmente según existe el parámetro **ajax=true**. En el caso de no existir dicho parámetro, el dispatcher generará una página del portal TAU completa, con su cabecera, su menu, y el cuerpo de la aplicación. Pero debido a que en determinadas ocasiones hacemos uso de la tecnología AJAX, no queremos que al pedir un método a una miniaplicación, ésta nos devuelva la salida tal cual, sin ningún marco de decoración (es decir, sin menu, sin cabecera). Por ejemplo, por AJAX, se puede insertar una nota en el módulo NOTAS, sin necesidad de regenerar la página completa. Por ello, al realizar la operación de **InsertarNota** hay que pasar como parámetro a la petición **ajax=true** con el fin de que la clase **Presentación** devuelva el contenido tal cual, sin ninguna decoración.

FORMA DE INVOCAR AL DISPATCHER

Tal y como se ha comentado, toda operación sobre alguna **MINIAPLICACION** o **MODULO** debe pasar obligatoriamente por el DISPATCHER. La forma de invocar una determinado operación sobre un determinado módulo es mediante una URL. Se puede utilizar tecnología **GET** o **POST** según sea preciso. Los parámetros de invocación son:

- **class**: Define el nombre del MODULO-MINIAPLICACION que se desea invocar
- **method**: Define el método del MODULO a invocar
- **ajax**: Si esta presente indica a la clase Presentación que no muestre decoración y devuelva el contenido tal cual.
- **Layout**: Nombre del TEMA a utilizar para devolver la página si es distinto al preferido por el usuario.
- **Parámetros**: opcionalmente pueden pasarse varios parámetros.

Así por ejemplo, si queremos borra la nota 7 dentro de la aplicación NOTAS, la URL a invocar sería:

http://SERVIDOR/dispatcherGET.php?class=Notas&method=borrarNota&id=7

Otro ejemplo, invocamos al módulo NOTAS para insertar una nueva NOTA, pero lo hacemos por AJAX, de manera que no se regenera la página, sino sólo una porción del área de visualización.

http://SERVIDOR/dispatcherGET.php?class=Notas&method=insertarNota&ajax=true&texto=Esto%20es%20una%20prueba