Relación 8: Sesiones, seguridad.

Se creará un nuevo proyecto llamado relacion8. Hay que definir nuestro sitio virtual www.relacion8.es en apache para que apunte a la carpeta creada.

Este proyecto tendrá la estructura indicada en la ESTRUCTURA DE LA APLICACIÓN. Todas las clases que se creen tienen deshabilitada la sobrecarga dinámica de propiedades.

Se deben de tener en cuenta todas las restricciones indicadas en prácticas anteriores como barra de ubicación, menú, subir a Sitio Web, etc

1.- Para personalizar la presentación del sitio web se van a usar cookies. Para ello, crear la página /aplicacion/personalizar/personalizar.php que nos permitirá seleccionar el color de fondo (combo con mínimo de cinco valores posibles: blanco, rojo, verde, azul, cyan, …) y el color del texto (combo con cuatro valores mínimos: negro, azul, blanco, rojo, …). Por defecto, el color de fondo será blanco y el del texto negro. Para consultar los colores posibles se crearán dos constantes (COLORESTEXTO y COLORESFONDO) en cabecera.php cada una con un array de los colores posibles para el texto y el fondo. Con cada color guardaremos una cadena que representa su valor en CSS (nombre en inglés o función RGB)

Se modificará la plantilla para que se inicialicen las cookies con los valores por defecto. Además, en la función inicio_cuerpo se deben consultar los valores de los colores que tenemos en las cookies y se establecerán adecuadamente esos colores para la página (por ejemplo, con propiedad style).

Además, en el fichero index.php se mostrará un contador que indique el número de veces que el usuario ha accedido a la página (almacenado mediante cookies).

2.- Definir la clase RegistroTexto (guardada en el fichero /aplicacion/clases/RegistroTexto.php). Esta clase guardará una cadena y una fecha/hora (datetime). Estas propiedades son privadas. Al constructor se indicará una cadena y utilizará la hora del sistema actual para rellenar el campo de fecha/hora. Las propiedades son accesibles mediante métodos públicos getXXXX (no se definen métodos setXXXX porque no se quieren cambiar los valores una vez creada)

Se creará el fichero /aplicacion/texto/verTextos.php donde el usuario escribirá distintos textos que quedarán registrados con la clase anterior y que podremos consultar en llamadas consecutivas. Para ello, definir un formulario con un text donde se indica el texto a registrar. Aparecerá también un textarea en el que se mostrarán todos los textos indicados por el usuario junto con la fecha/hora en la que lo ha introducido y un botón Limpiar para borrar todos los textos.

Se trabajará con el array $textos en donde se irán guardando objetos de tipo RegistroTexto. Este array se usará en todo lo anterior.

Para mantener los textos entre llamadas consecutivas, se guardará en sesión una copia del array $textos trabajando de la siguiente forma:

- Al inicio del script se recogerán los textos que tengamos en la sesión y se guardan en el array $textos.
- Se trabaja con el array $textos en el fichero
- Cuando se añada o borre un texto, se actualizará el array. Además, se actualizará la sesión para que estén los textos correctos para llamadas posteriores.

  Nota en la operación limpiar se actualiza la información de la sesión borrando los

textos. No se borra la sesión ya que podemos borrar otros elementos en sesión necesarios.

La sesión, al usarla en diferentes páginas se creará en cabecera.php.

Dentro de /index.php tendremos un enlace a la página /textos/verTextos.php.

3.- Para restringir el acceso a la página vamos a validar quien se conecta generando para ello la clase abstracta ACLBase con nombre ACLBase.php. Esta clase junto con las siguientes se reutilizarán en futuros proyectos por lo que la colocamos /scripts/clases. (garantizar la autocarga).

En la ACL se gestionarán dos elementos:

- Roles: Distintos roles del sistema. De un role se sabe el nombre, un código y 10 permisos posibles (permiso1, permiso2, …..permiso10) usados para controlar diferentes aspectos en la aplicación en donde se use.
- Usuarios: Distintos usuarios en el sistema. De un usuario se sabe código (único, autonumérico). el nombre, el Nick (único, identifica al usuario), la contraseña, código de role. Además se tiene un borrado lógico de los usuarios

En esta clase se definirá la siguiente interfaz (funciones abstractas):

- anadirRole(string $nombre, array $permisos=array()): bool. Permite añadir un nuevo role con el nombre y los permisos indicados. Los permisos se establecen con un array de hasta 10 elementos. No es necesario establecer valor para todos los permisos (array(1=>true, 2=>false, 5=>true). Devuelve true si se ha podido añadir el role y false en caso de que no (ya existe otro role con el mismo nombre). Si no se indica un permiso concreto, se asignará a false. Se asigna un código de role en secuencia (autonumérico).
- getCodRole(string $nombre): int. Devuelve el código de role correspondiente al $nombre o false si no lo encuentra.
- existeRole(int $codRole):bool. Devuelve true si existe un role con el código indicado.
- getPermisosRole(int $codRole): array|false. Devuelve los permisos asignados a un role.
- getPermisoRole(int $codRole, int $numero): bool. Devuelve el permiso $numero del role.
- anadirUsuario(string $nombre, string $nick, string $contrasena, int $codRole):bool. Añade un nuevo usuario con los datos indicados. Devuelve true si lo ha podido crear o falso en caso de error (Nick repetido, codRole que no existe, etc). El Nick debe ser único. Se asignará un código autonumérico único.
- getCodUsuario(string $nick):int|false : Devuelve el código de usuario correspondiente a un Nick o false si no lo encuentra.
- existeCodUsuario(int $codUsuario):bool: Devuelve true si existe un usuario con el código indicado. False en caso contrario.
- existeUsuario(string $nick):bool. Devuelve true si existe un usuario con el Nick indicado. False en caso contrario.

- esValido(string $nick, string $contrasena):bool: Comprueba dentro de la ACL disponible si el $nick (usuario) existe y coincide su contraseña. Devuelve true si es válido y false en otro caso.
- getPermiso(int $codUsuario, int $numero):bool. Devuelve si tiene o no el permiso numero $numero el usuario indicado.
- getPermisos(int $codUsuario):array. Devuelve un array de todos los permisos que puede tener el usuario
- getNombre(int $codUsuario):string|false. Función que devuelve el nombre dado un código de usuario o false si no lo encuentra.
- getBorrado(int $codUsuario):bool: Función que devuelve si el usuario (con código dado) está borrado o no.
- getUsuarioRole(int $codUsuario):int|false. Función que devuelve el código del role para el usuario indicado o false si no lo encuentra
- setNombre(int $codUsuario, string $nombre):bool. Función que permite cambiar el $nombre de un usuario identificado por el codigo. Devuelve si se ha podido hacer
- setContrasenia(int $codUsuario, string $contra):bool. Función que permite establecer la contraseña para el usuario identificado por el código. Devuelve si se ha podido hacer.
- setBorrado(int $codUsuario, bool $borrado):bool. Función que permite establecer si el usuario esta borrado o no. Devuelve si se ha podido hacer
- setUsuarioRole(int $codUsuario, int $role): Función que permite cambiar el role a un usuario concreto. Devuelve si se ha podido hacer
- dameUsuarios(): array. Esta función devuelve un array asociativo donde el indice es el código de usuario y el valor es el nick.
- dameRoles(): array. Esta función devuelve un array indexado en el que el índice es el código de role y el valor el nombre.

4.- Crear la clase ACLArray (fichero /scripts/clases/ACLArray.php) que hereda de ACLBase que se encargará de gestionar una ACL usando para su almacenamiento arrays.

Para definir la ACL se tendrá

- Array privado $_roles que contiene los distintos roles en el sistema. Inicialmente se tienen dos roles: normales (permiso 1) y administradores (permiso 1 y 2).

- Array privado $_usuarios que contiene todos los usuarios en el sistema. En principio se tendrán como usuarios: alumno (role normales), profesor (role administradores) y XXXX (vosotros como usuario, role administradores).

La contraseña se guardará cifrada. Usaremos la función password_hash con el método PASSWORD_BCRYPT

Los arrays se pueden definir asociativos usando para ello el elemento identificativo. Esto nos facilitará nuestro trabajo posterior.

5.- Una vez que se tiene el mecanismo de ACL, el siguiente paso es mantener la información de si hay usuario registrado en la aplicación (se ha validado dando su Nick y contraseña) y sus datos.

Para mantener la información del usuario registrado se tendrá la clase Acceso (fichero

/scripts/clases/Acceso.php). Esta clase se encargará de guardar la información de acceso del usuario en la aplicación de forma que sea accesible en todas las páginas de la misma. Esto representa un estado y como tal se guarda normalmente usando las sesiones.

Se guardará si hay un usuario validado (validado), el Nick del usuario (cadena, único), el nombre (cadena) y los permisos que tiene (array indexado de 1 a 10 con valor true/false: si tiene o no el permiso X).

Se trabajará teniendo en la clase una serie de propiedades privadas (son las que realmente se consultan) y guardando estos datos en sesión para mantenerla de forma permanente (para que mantengamos el estado).

Esta clase tendrá los siguientes métodos:

- Propiedades privadas (todas las que estimes necesarias).
- constructor. Se encarga de recoger de la sesión los diferentes valores almacenados y asignarlos a las propiedades privadas apropiadas.
- registrarUsuario(string $nick, string $nombre, array $permisos):bool. Sirve para registrar un usuario en la aplicación. Almacena los valores en las propiedades apropiadas y en la sesión para guardar la información del usuario validado.
- quitarRegistroUsuario():bool . Hace que no haya ningún usuario registrado en el sistema
- hayUsuario():bool. Devuelve true si hay un usuario validado y false en caso contrario.
- puedePermiso($numero):bool : Devuelve true si tiene el permiso $numero.
- getNick():string , getNombre():string : funciones que devuelven respectivamente el Nick y el nombre del usuario registrado.

Esta clase es independiente de la ACL. En una aplicación se trabajaría de la siguiente forma:

- Se tiene un objeto de clase ACL y otro de la clase Acceso.
- Cuando se quiere comprobar si existe un usuario o no, crear un usuario, crear un role, obtener la información de un usuario, se usa el objeto de la clase ACL.
- Cuando se valida un usuario (introduce su Nick y contraseña), primero se comprueba con ACL y después se guarda su información en la sesión con el objeto de la clase Acceso.

En la aplicación se tiene que dar soporte al control de usuario de la siguiente forma:

- Un usuario se tiene que validar antes de poder ver cualquier página del sitio (normalmente excepto la página index). Si no está validado se muestra la petición de login/contraseña. El login se hará mediante la página /aplicacion/acceso/login.php.
- Una vez validado la primera vez, no tiene que validarse cuando visite las siguientes páginas.
- Una vez validado se debe mostrar en todas las páginas el nombre del usuario. Además, debe haber algún mecanismo para cerrar la sesión (enlace, botón. etc). Si un usuario cierra la sesión pasa a estar no validado.
- Un usuario validado podrá acceder a las distintas páginas si tiene permiso 1. Si no tiene permiso se debe mostrar una página de error apropiada (ver función paginaError en apartado 4.1 del tema 3).
- Un usuario podrá configurar los colores sólo si tiene permiso 2.

Para implementarlo se aconseja crear en el fichero cabecera.php una instancia de ACLArray y otra de Acceso. En cada página tener en cuenta si el usuario está o no validado (en caso de no estar redireccionar al login) y en cada página comprobar los permisos de acceso/configuración. (ver apartado 4.2 del tema 3)