DESARROLLO DE APLICACIONES WEB CON PHP

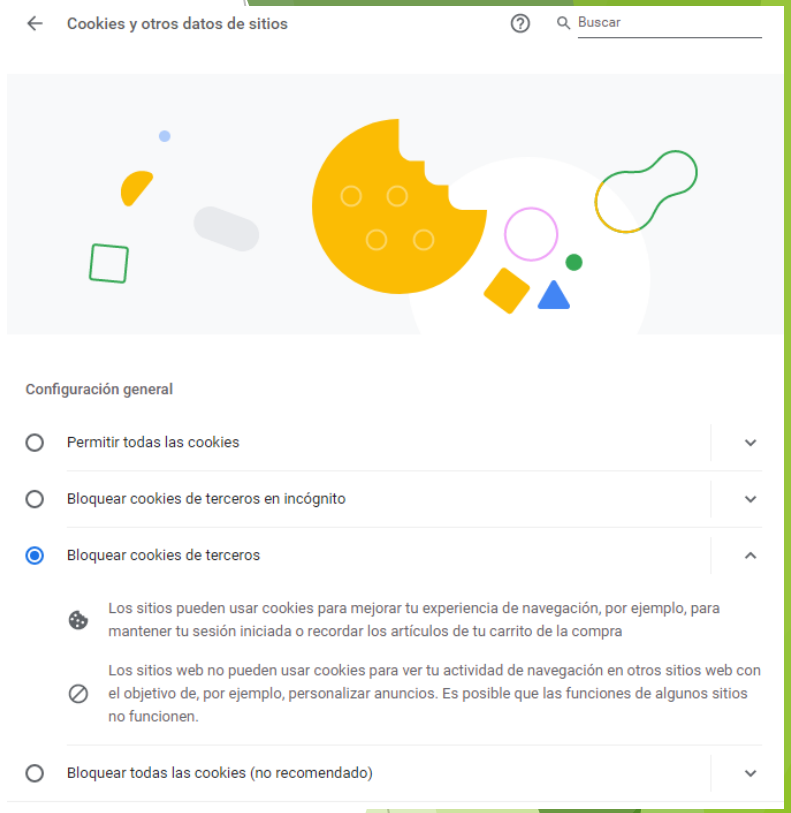
UNIDAD 3

1 COOKIES

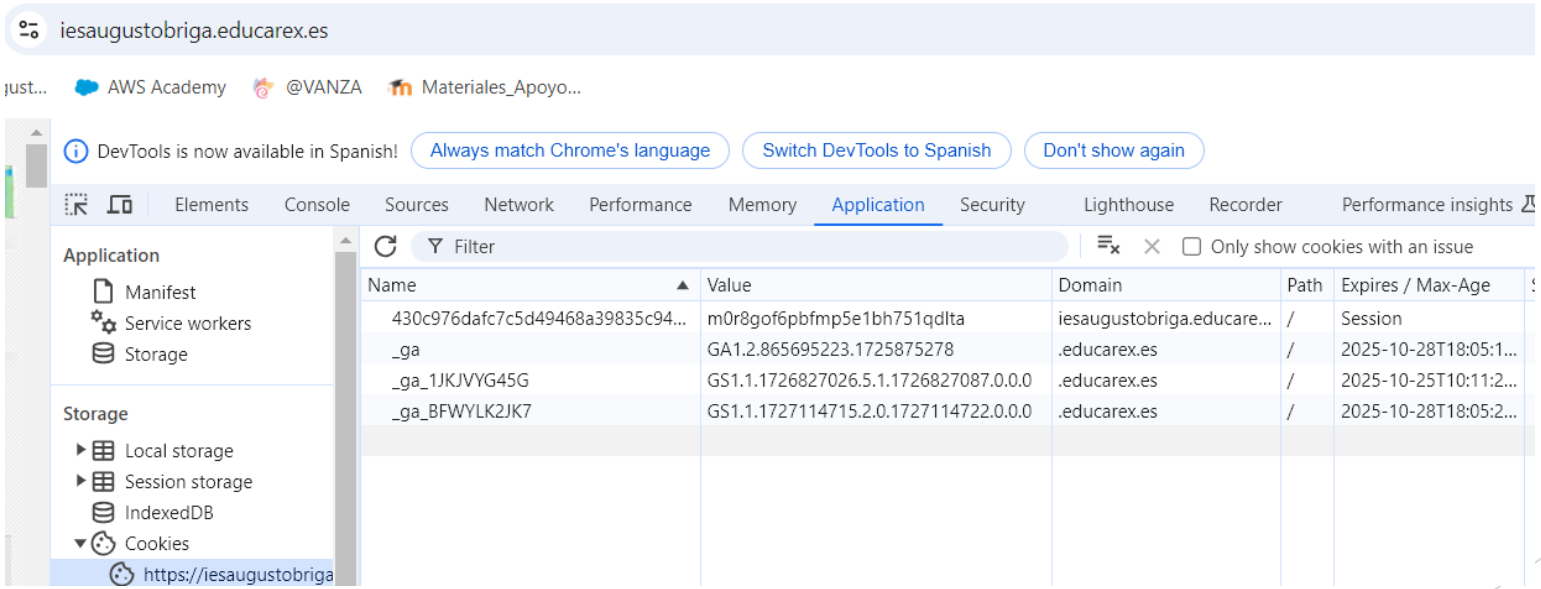
- Fichero que un sitio web guarda en el cliente (depende del navegador).
- Estructura
  - Identificador
  - Valor
  - Caducidad: Tiempo de validez
  - Dominio: Dominio en el cuál es válida
  - Ruta: Directorio en el que tiene validez
  - Seguro: Sólo para HTTPS
- Se usar para compartir información entre distintas conexiones:
  - Recordar el idioma
  - Recordar credenciales
  - Fines comerciales (cookies de 3º)
- Siempre que utilices cookies en una aplicación web, debes tener en cuenta que en última instancia su disponibilidad está controlada por el cliente. Por ejemplo, algunos usuarios deshabilitan las cookies en el navegador porque piensan que la información que almacenan puede suponer un potencial problema de seguridad. O la información que almacenan puede llegar a perderse porque el usuario decide formatear el equipo o simplemente eliminarlas de su sistema.

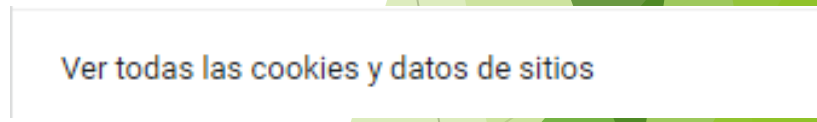
2 COOKIES EN EL NAVEGADOR

- Chrome: Configuración/Privacidad y Seguridad/Cookies y otros datos de sitio.



- Si queremos ver que contienen las cookies que tenemos almacenadas en el navegador, se puede comprobar su valor en *Dev Tools → Application → Storage*



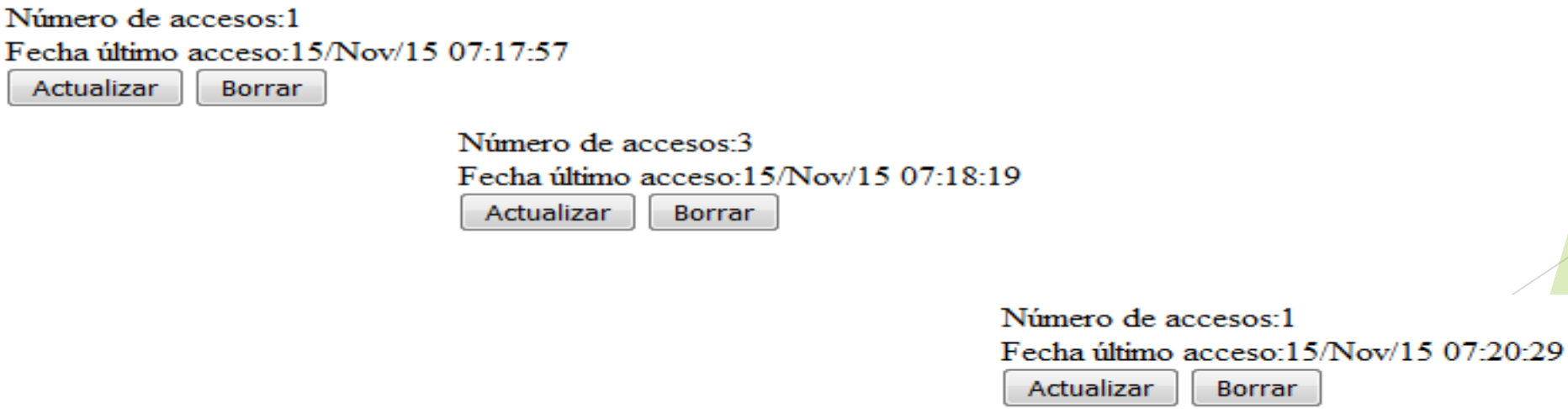


3 GESTIÓN DE UNA COOKIE

- Creación:
  - setcookie(nombre_cookie, valor, caducidad, ruta, dominio, modo_seguro, solo_http)
  - Si no se pasa *caducidad*, la duración será la duración de la sesión.
- Eliminación
  - setcookie(nombre_cookie,””,time()-1): Al tener la fecha de caducidad negativa desaparece.
- Obtención
  - isset($_COOKIE[“nombre_cookie"]): Para comprobar si está definida la cookie
  - $_COOKIE[“nombre_cookie"]: para trabajar con ella
- Existe una limitación de 20 cookies por dominio y 300 en total en el navegador
- Destacar que el nombre no puede contener espacios ni el caracter ;
- Respecto al contenido de la cookie, no puede superar los 4 KB.
- Buenas prácticas de Desarrollo: Comprobar el usuario acepta las Cookies. Para ello puedes crear una cookie y comprobar el valor.

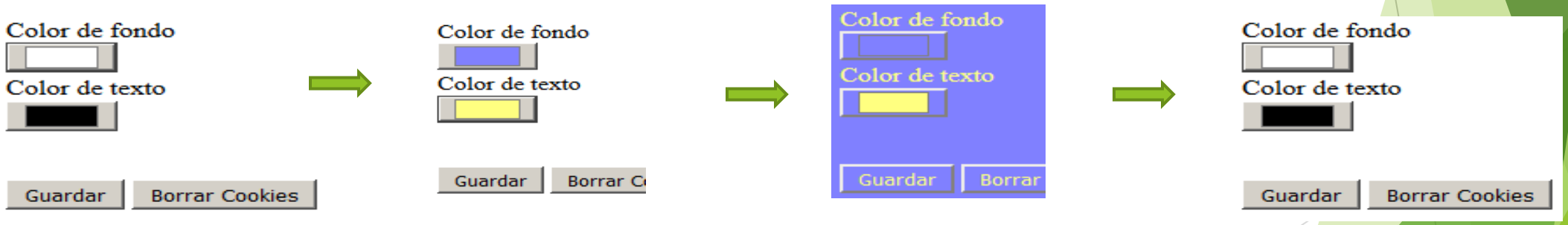
Ejercicio 1

- Realiza una página web que, utilizando cookies, muestre el número de visitas que ha realizado el usuario a la página y la fecha y la hora de la última visita.
- La fecha de caducidad de las cookies será de un año.



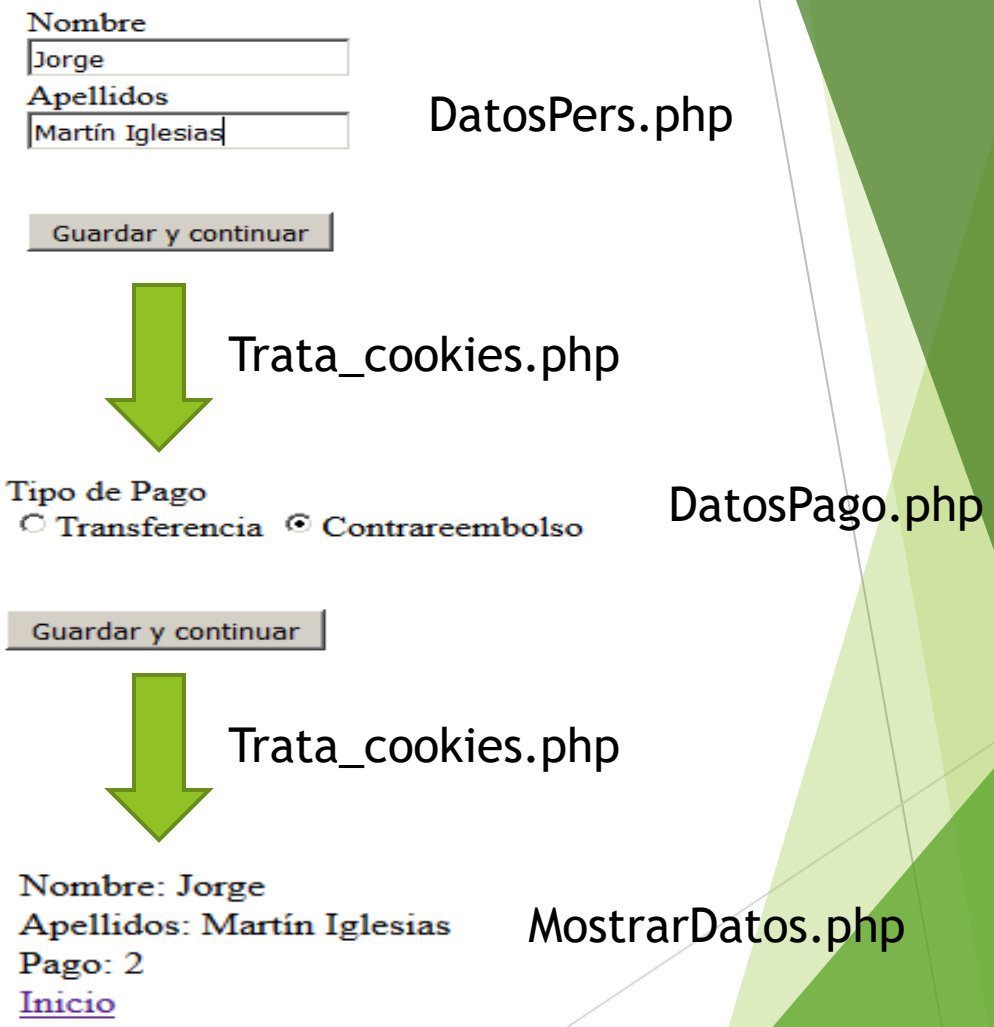
Ejercicio 2

- Realiza una página en la que el usuario pueda seleccionar el color de fondo de la página y el color de fondo del texto. Estos colores se guardarán en una cookie y se utilizarán cada vez que se acceda a la página. Se debe permitir borrar las cookies.



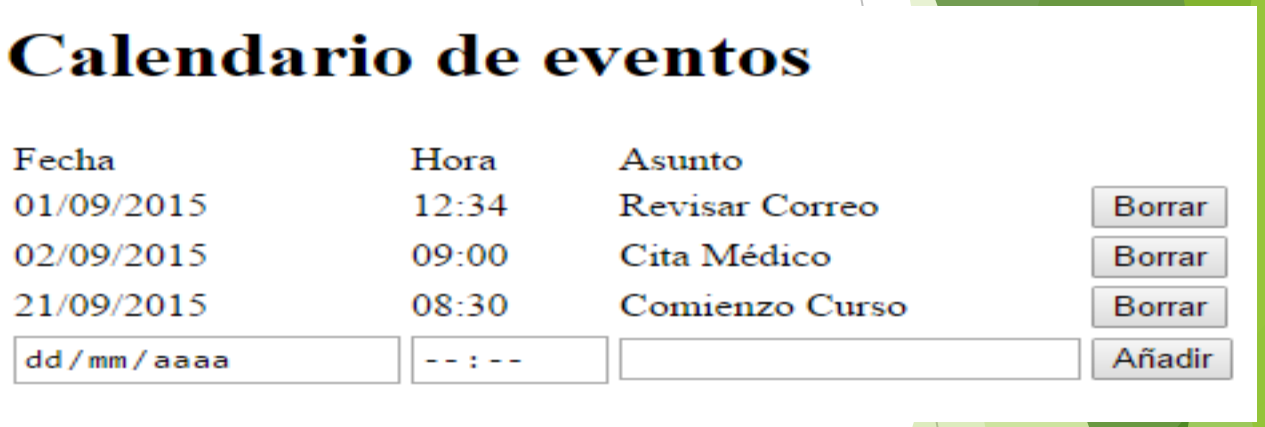
Ejercicio 3

- Realiza un conjunto de páginas que vayan solicitando información a los usuarios.
- La primera página solicitará el nombre y los apellidos esta información se guardará en cookies y cuando se vuelva a acceder a está página se recuperará.
- La segunda página nos solicitará el tipo de pago y funcionará exactamente igual que la anterior.
- Por último la tercera página muestra la información introducida.
- El script que guarda los datos en cookies debe ser único y nos debe redireccionar a la página que corresponda.



Ejercicio 4

- Realiza usando cookies una aplicación Web en PHP que permite llevar una lista de eventos.
- Cada evento tiene una fecha, hora, y asunto. Se pueden agregar nuevos eventos o eliminar los existentes.
- Utiliza funciones en la medida de lo posible.
- Para guardar un array en una cookie utiliza la función **serialize(array)** y para recuperarlo **unserialize($_COOKIE…)**.
- Cada vez que haya que modificar el array hay que desempaquetarlo, modificarlo y volverlo a empaquetar para guardarlo en el cookie.
- Para reindexar un array después de haber borrado algún elemento, utiliza la función **array_values(array)**.
- Para borrar un elemento del array usa la función unset.



4 SESIONES

- Cookies del lado del servidor. Guardan información de los usuarios del sitio web en el servidor. Permiten compartir información entre las distintas páginas del sitio web. Ejemplo: Credenciales de acceso.
- Cada usuario tiene 1 sesión identificada por el ***SID***. A partir de él se puede obtener la información guardada en esa sesión.
- Para mantener el SID entre las páginas de un sitio web que vista un usuario (lo hace automáticamente PHP):
  - Crear una cookie con el SID
  - Pasar el SID por parámetro en la url
- Para cada script que utilice los datos de la sesión hay que recuperar la sesión.
- La sesión se cierra al cerrar el navegador o si lleva un tiempo configurable sin actividad.

5 CONFIGURACION SESIÓN(PHP.INI)

- **session.use_cookies**: Indica si se deben usar cookies (1) o propagación en la URL (0) para almacenar el SID.
- **session.use_only_cookies**: No se reconoce el SID aunque se pasen en la URL.
- **session.save_handler:** Dónde se almacenan los datos de la sesión *file*, *mm*, *sqlite*, *user*. File por defecto.
- **session.name**: Nombre de la cookie que guarda el SID. PHPSESSID por defecto.
- **session.auto_start**: Inicio automática de sesión, no hace falta llamar a la función session_start. 0 por defecto.
- **session.cookie_lifetime**:Tiempo es segundos que debe durar la cookie una vez que se cierre el navegador.
- **session.gc_maxlifetime**:Indica el tiempo en segundos que se debe mantener activa la sesión, aunque no haya ninguna actividad por parte del usuario. Pasado ese tiempo desde la última actividad, la sesión se desconecta automáticamente.

6 MANEJO DE SESIÓN

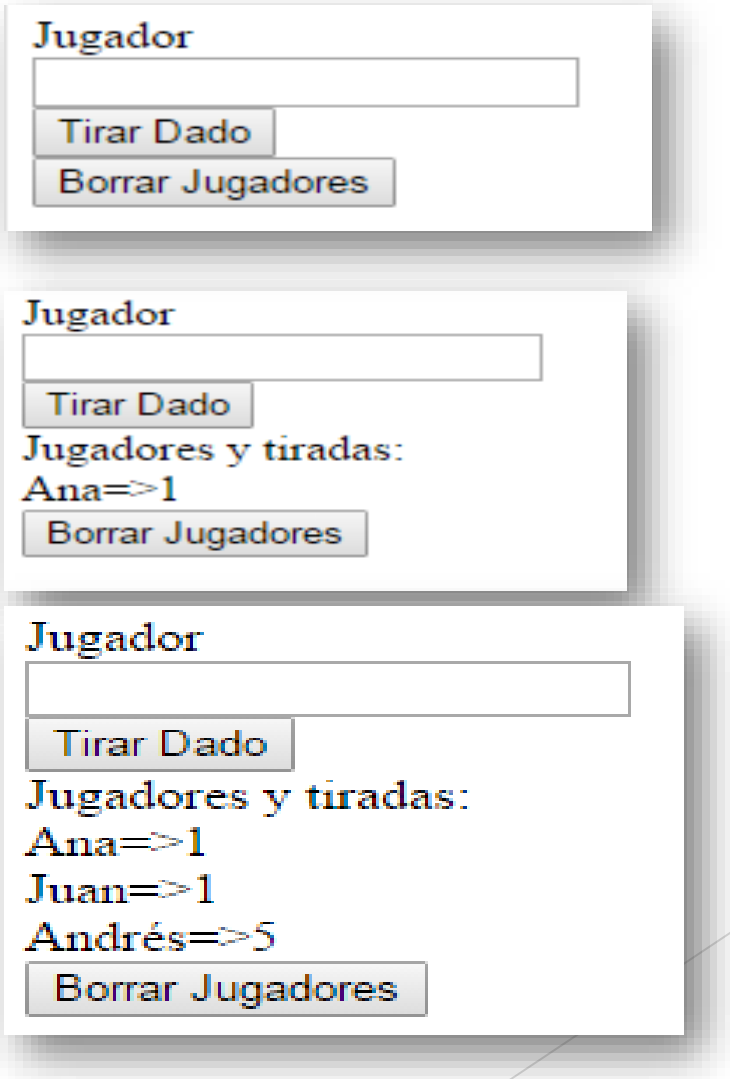
- Inicio/Reanudación
  - session_start(): Devuelve false si no se puede. Es necesario llamar a esta función para poder trabajar con la sesión.
- Establecer/Obtener SID
  - session_id(["id"]): Si no se pasa id, se asigna uno automáticamente.
- Establecer/Obtener Nombre
  - Session_name(([“nombre”]): Si no se pasa nombre, se pone el de por defecto.
- Variables de sesión
  - Array $_SESSION
- Eliminar todas las variables de una sesión pero no la sesión.
  - session_unset(): No elimina las variables de $_SESSION
- Eliminar completamente una sesión.
  - Cerrar el navegador
  - session_destroy()
    - No elimina las variables de $_SESSION
    - Si se usa cookie para almacenar el SID, hay que borrar la cookie (***setcookie(session_name(),””,time()-1000***))

Ejercicio 5

- Crea una página que almacene en la sesión de usuario los instantes de todas sus visitas. Si es su primera visita, muestra un mensaje de bienvenida y el SID. En caso contrario, muestra la fecha y hora de todas sus visitas anteriores (Se debe guardar un array en $_SESSION). Añade un botón a la página que permita cerrar la sesión.

Ejercicio 6

- Crea un sitio web que permita tirar un dado a varios jugadores. El número de jugadores es ilimitado. Cuando se introduzca el nombre del jugador y se pulse en Tirar Dado, se generará un número aleatorio entre 1 y 6. A continuación se mostrarán los usuarios y sus tiradas. Se podrá borrar los jugadores. Debes hacerlo creando un array como variable de sesión en el que los nombres de los usuarios sean las claves del array y para cada uno se guarde el valor del dado:



7 ACCESO A BD RELACIONALES

- Extensiones
  - MySQLi: Solo para MySQL. Se puede trabajar con programación orientada a objetos o con funciones.
  - PDO (PHP Data Objects): Es independiente del SGBD. Solamente ofrece la posibilidad de trabajar con programación orientada a objetos. No es necesario usar un ORM.
- Permiten:
  - Establecer conexiones.
  - Operaciones CRUD.
  - Emplear transacciones.
  - Ejecutar procedimientos almacenados.
  - Gestionar los errores que se produzcan durante la conexión o en el establecimiento de la misma.

8 PDO

- Utiliza un controlador específico para el tipo de SGBD que se utilice (consultar en phpinfo()).

PDO

PDO support	enabled
PDO drivers	mysql, sqlite

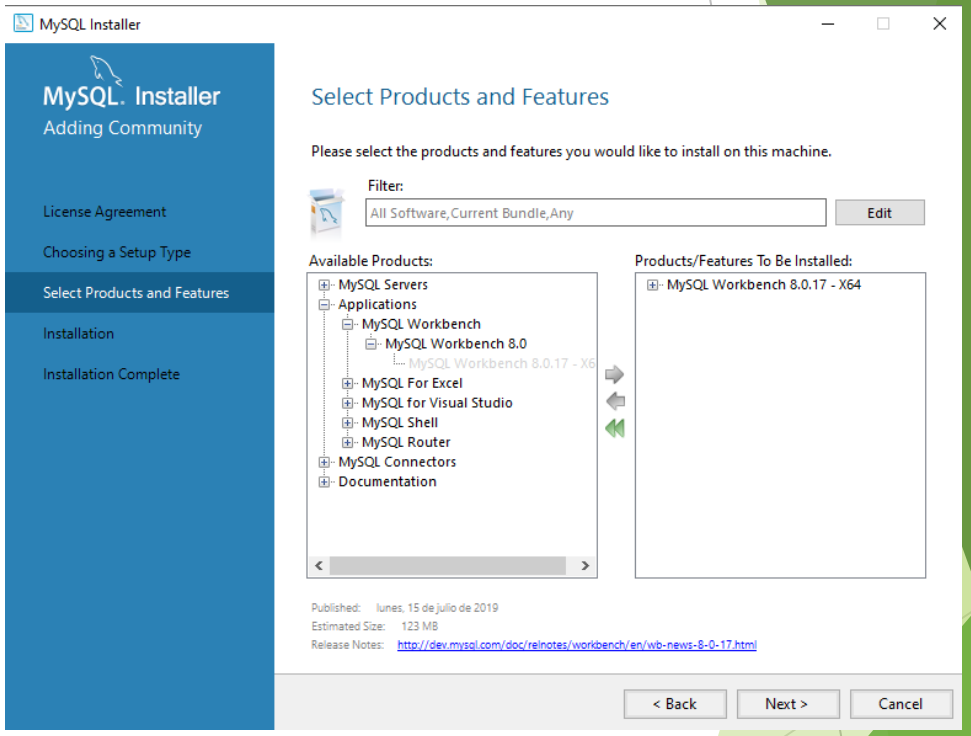
- Control de excepciones: PDOException

```
try {

} catch (PDOException $e) {
    echo "<br/>".$e->getMessage();
}
```

9 INSTALACIÓN SGBD y CLIENTE

- Servidor MySql de XAMPP
- Cliente
  - Phpmyadmin
  - **MySQLWokbench**

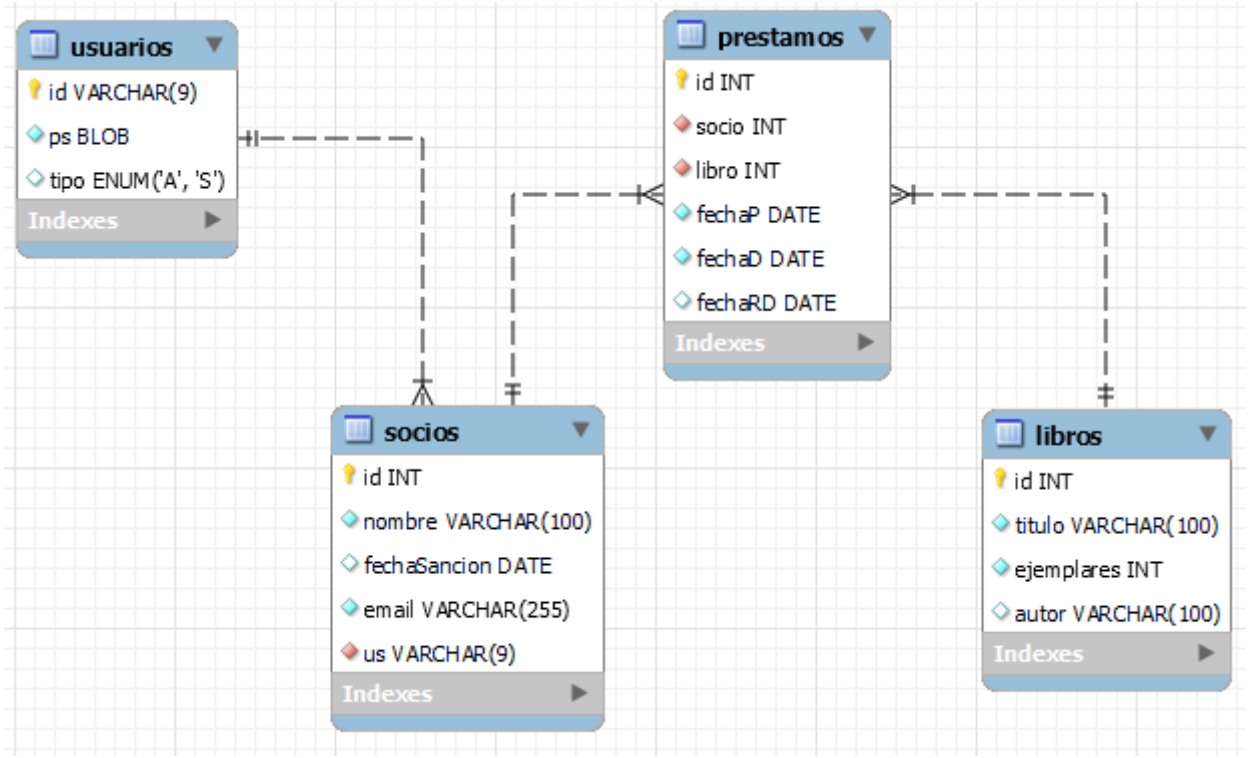


10 ESTABLECIMIENTO DE LA CONEXIÓN

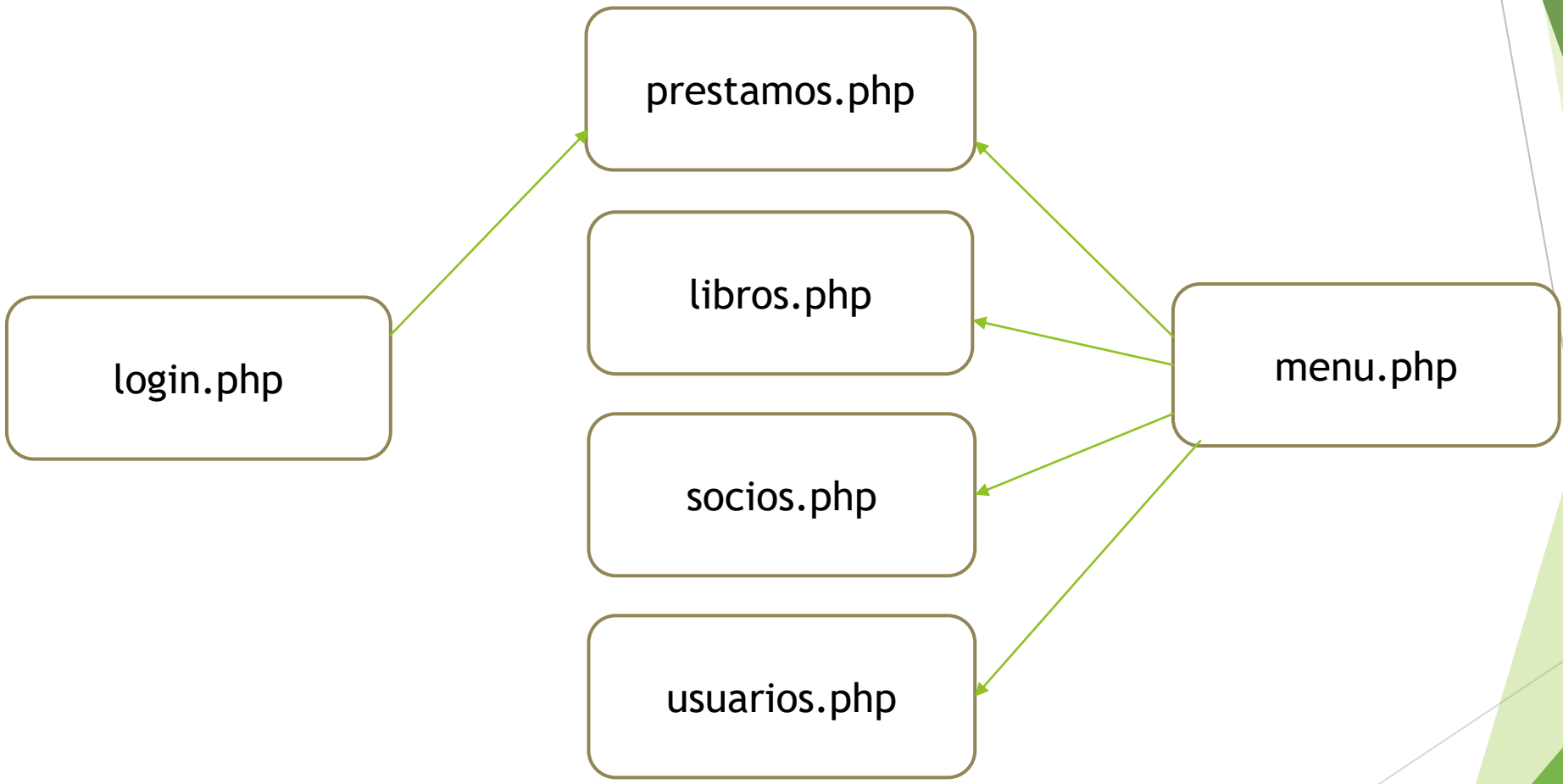
- Instanciar un objeto de la clase PDO pasándole los siguientes parámetros:
  - **Nombre del Origen de datos (DSN)**. Es una cadena de texto que indica qué controlador se va a utilizar y a continuación, separadas por el carácter dos puntos, los parámetros específicos necesarios por el controlador separados por ;. Los parámetros que se suelen indicar son:
    - Nombre o ip del servidor
    - Puerto en el que escucha el servidor: Por defecto, mysql utiliza el puerto 3306.
    - Nombre de la BD
  - **Nombre de usuario con permisos para establecer la conexión**.
  - **Contraseña del usuario**.
  - **Opciones de conexión, almacenadas en forma de array**.

```
$conexion=new PDO("mysql:host=localhost;port=3306;dbname=nombreBD",
    ["usuario"], ["contraseña"],[$array_opciones]);
```

11 BD BIBLIOTECA(biblioteca.sql)



11 BIBLIOTECA (Aplicación)



12 CONSULTAS SELECT

Sin parámetros

- Se ejecutan con **$conexion->query($consulta)** que devuelve un objeto de la clase PDOStatement.

```
$textoConsulta = 'select * from libros';
$resultadoConsulta = $conexion->query($textoConsulta);
while($fila= $resultadoConsulta->fetch()){
    echo $fila['atributo1'],..., $fila['atributoN']
}
```

- Recuperación del resultado: Se usa el método **fetch()** de la clase PDOStatement. Devuelve:
  - array asociativo con la información de un resultado de la consulta
  - False si no quedan más registros por recorrer.
- Por defecto, el método fetch genera y devuelve, a partir de cada registro, un array con claves numéricas y asociativas. Para cambiar su comportamiento, admite un parámetro opcional que puede tomar uno de los siguientes valores:
  - PDO::FETCH_ASSOC. Devuelve solo un array asociativo.
  - PDO::FETCH_NUM. Devuelve solo un array con claves numéricas.
  - PDO::FETCH_BOTH. Devuelve un array con claves numéricas y asociativas. Es el comportamiento por defecto.
  - PDO::FETCH_OBJ. Devuelve un objeto cuyas propiedades se corresponden con los campos del registro.

Con parámetros

- Tres pasos: Preparar consulta, rellenar parámetros y ejecutar
- Opción 1

```
$consulta=$conexion->prepare('select * from libros where id =:param1 and titulo= :param2');
$parametros= array(':param1'=>valor, 'param2'=>valor);
$resultado = $consulta->execute($parametros);
if($resultado){
        if( $fila=$consulta ->fetch()){
            echo $fila['atributo1'],..., $fila['atributoN'];
        }
}
```

- Opción 2

```
$consulta=$conexion->prepare"(select * from libros where id =:param1 and titulo= :param2");
$parametros= array(valor1, valor2);
$resultado = $consulta->execute($parametros);
if($resultado);
        if( $fila=$consulta ->fetch()){
            echo $fila['atributo1'],..., $fila['atributoN'];
        }
}
```

13 CONSULTAS INSERT/UPDATE/DELETE

- Se definen y ejecutan como una consulta con parámetros.
- Para obtener el nº de registros afectados: $consulta->rowCount()
- Obtener último id generado: **$conexion->lastInsertId()**

14 EJECUCIÓN DE RUTINAS ALMACENADAS

FUNCIONES

- Se llaman dentro de un select

```
try {
      $consulta = $conexion->prepare("select funcion(?)");
      if($consulta->execute(array($valor))){
          if($fila=$consulta->fetch()){
             echo "<br/>El resultado es ".$fila[0];
           }
       }
 } catch (PDOException $e) {
       echo "<br/>".$e->getMessage();
 }
```

PROCEDIMIENTOS

```
try {
      $consulta = $conexion->prepare("CALL
      procedimiento(?)");
       if$consulta->execute(array($valor))){
          if($fila=$consulta->fetch()){
             echo "<br/>El resultado es ".$fila[0];
           }
       }
 } catch (PDOException $e) {
       echo "<br/>".$e->getMessage();
 }
```

- Si hay parámetros de salida, hay que registrarlos con bindPram (**Bug de API PDO->No funciona**).

```
$sentencia = $mbd->prepare("CALL    procedimiento(?)");
//Registrar parámetro de salida 1 de tipo string
$sentencia->bindParam(1, $variableOUT, PDO::PARAM_STR);
// llamar al procedimiento almacenado
$sentencia->execute();
echo $variableOUT
```

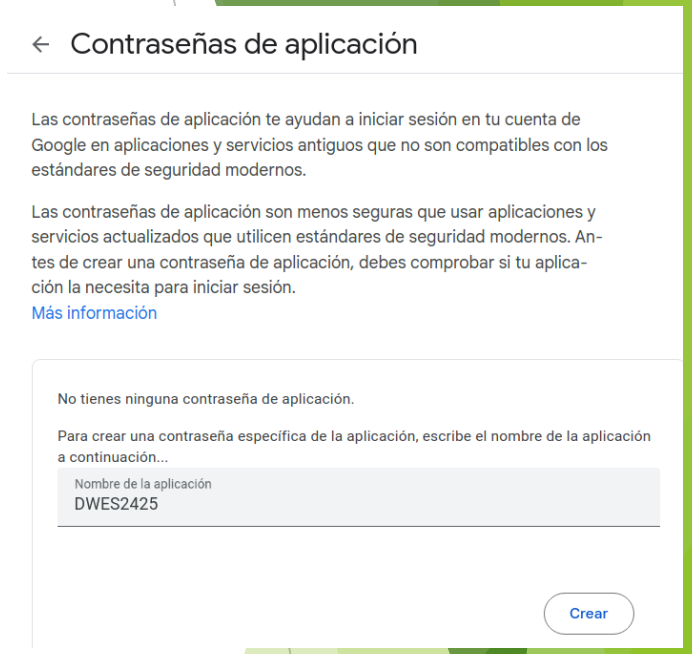
- Si el procedimiento muestra más de un resultado, se puede recuperar con la sentencia $sentencia->nextRowSet()

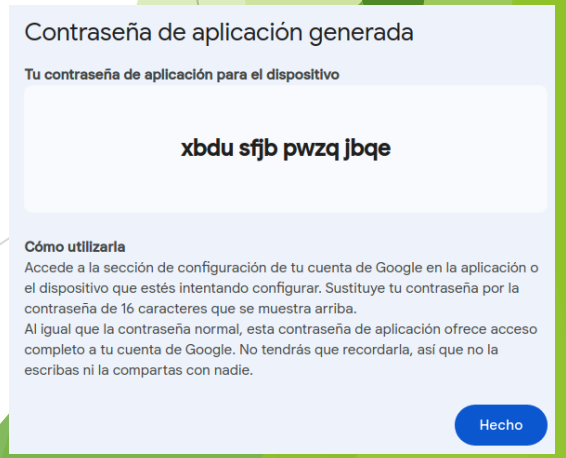
16 TRANSACCIONES

- Ofrecen cuatro características principales: Atomicidad, Consistencia, Aislamiento y Durabilidad (ACID por sus siglas en inglés). En términos sencillos, se garantiza que cualquier trabajo llevado a cabo en una transacción, incluso si se hace por etapas, sea aplicado a la base de datos de forma segura, y sin interferencia de otras conexiones, cuando sea consignado.
- Si al finalizar un script hay una transacción pendiente de confirmar, se deshace automáticamente.
- Toda consulta que se ejecute tiene su propia transacción implícita.
- Depende del motor de la BD.
- Para trabajar con transacciones, PDO incorpora tres métodos:
  - $conexion->beginTransaction. Deshabilita el modo "autocommit" y comienza una nueva transacción, que finalizará cuando ejecutes uno de los dos métodos siguientes.
  - $conexion-> commit. Confirma la transacción actual.
  - $conexion-> rollback. Revierte los cambios llevados a cabo en la transacción actual.

17 ENVÍO DE CORREO - CONFIGURACIÓN

- Instalar gestor de dependencias de php **Composer**
  - Windows: Ejecutar instalador https://getcomposer.org/Composer-Setup.exe.
  - Ubuntu: sudo apt install composer
  - Se instala en C:\ProgramData\Composer Setup\bin y se actualiza variable de entorno PATH
- Comprobar en **php.ini** que no está comentada la línea: extension=zip
- Descargar **Librería PHPMailer** desde Composer
  - Desde consola, situarse en el sitio web y ejecutar: composer require phpmailer/phpmailer
- Configurar cuenta de Google desde las que se envían los correos
  - Antes, fácil de usar con cuentas Google, simplemente había que activar aplicaciones menos seguras.
  - A partir de mayo de 2022, Google no deja activar aplicaciones menos seguras. Para poder usar la cuenta de Gmail desde php necesitamos generar una contraseña de apliación y para ello debemos activar la verificación en dos pasos.
    - Gestionar cuenta de Google/seguridad
    - Iniciar sesión en Google/Verificación en dos pasos/Activar verificación en dos pasos
    - Buscar Contraseñas de aplicación y crear una contraseña de aplicación





18 ENVÍO DE CORREO - EJEMPLO

```php
//Incluir librería PHPMailer
use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\Exception;
require_once './vendor/autoload.php';
function enviarCorreo(){
$resultado = false;
try{
$correo = new PHPMailer(true);
//Confirgurar datos del servidor saliente
$correo->isSMTP(); $correo->Host = 'smtp.gmail.com';
$correo->SMTPAuth = true; $correo->Username= 'tucorreo@gmal.com';
$correo->Password = 'tu Contraseña De Aplicacion';
$correo->SMTPSecure=PHPMailer::ENCRYPTION_SMTPS;
$correo->Port=465;
//Configuración del correo que vamos a escribir
$correo->setFrom('tucorreo@gmal.com','Tu Nombre');
$correo->addAddress('correoPara@xxx.xx','Nombre Destinatario 1');
$correo->addAddress('correoPara@xxx.xx','Nombre Destinatario 2');
//Configuración del contenido del mensaje
$correo->isHTML(true); $correo->CharSet='UTF-8';
$correo->Subject='Asunto '; $texto = 'Texto del correo con html';
$correo->Body=$texto;
$correo->AltBody="<h1>Textos del correo</h1>";
//Enviar correo
if($correo->send()){ $resultado=true;}
}
catch(Exception $e){ echo $e->getMessage();}
return $resultado;
}
```

```php
<?php
//Import PHPMailer classes into the global namespace
//These must be at the top of your script, not inside a function
use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\SMTP;
use PHPMailer\PHPMailer\Exception;

//Load Composer's autoloader
require_once '../../vendor/autoload.php';
function enviarCorreo($mensajeHTML, $mensajeNoHTML){
    $resultado = false;
    //Create objeto para Email. true indica que se capturan las excepciones
    $mail = new PHPMailer(true);
    try {
        //Datos del servidor de correro
        //$mail->SMTPDebug = SMTP::DEBUG_SERVER;            //Enable verbose debug output
        $mail->isSMTP();                                    //Send using SMTP
        $mail->Host       = 'smtp.gmail.com';               //Set the SMTP server to send through
        $mail->SMTPAuth   = true;                           //Enable SMTP authentication
        $mail->Username   = 'rmmateosv@gmail.com';          //SMTP username
        $mail->Password   = [redacted];                     //SMTP password
        $mail->SMTPSecure = PHPMailer::ENCRYPTION_SMTPS;    //Enable implicit TLS encryption
        //TCP port to connect to. use 587 if you have set `SMTPSecure = PHPMailer::ENCRYPTION_STARTTLS`
        $mail->Port       = 465;
        //Recipients
        $mail->setFrom('rmmateosv@gmail.com', 'Taller');
        $mail->addAddress('rmmateos@gmail.com', 'Destinatario1');     //Add a recipient
        $mail->addAddress('rmmateos@educarex.es','Destinatario2');    //Name is optional
        //$mail->addReplyTo('info@example.com', 'Information');
        //$mail->addCC('cc@example.com');
        //$mail->addBCC('bcc@example.com');
        //Attachments
        //$mail->addAttachment('/var/tmp/file.tar.gz');         //Add attachments
        //$mail->addAttachment('/tmp/image.jpg', 'new.jpg');    //Optional name
        //Content
        $mail->isHTML(true);                                  //Set email format to HTML
        $mail->Subject = 'Factura Reparación';
        $mail->Body    = $mensajeHTML;
        $mail->AltBody = $mensajeNoHTML;
        if($mail->send())
          $resultado = true;
        else
          return true;
    } catch (Exception $e) {
        return "Error al enviar el mensaje: {$mail->ErrorInfo}";
    }
    return $resultado;
}
```