PDO

Base de datos

En un documento php dentro de la carptea de lib que luego debemos linkear en cada uno de los otros doc con un `Include_once("libs/data_base.php");`

Conexión

Para abrir la conexión a través del método PDO es necesario establecer unas variables antes. En ellas se establecerá el nombre del servidor,el usuario y contraseña y el nombre de la base de datos.

```
    $servername='db';
    $username='root';
    $password='test';
    $dbname='donacion';
```

Una vez que esto se ha decidido, abrimos un try/catch para establecer la conexión y recoger el error en el caso de que suceda.

```
    try{
        $connection = new PDO("mysql:host=$servername;dbname=$dbname",
$username,$password);
        $connection ->
setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);
        echo "Database connected successfully";
        return $connection;
    } catch (PDOException $e){
        echo "Connection failed: " . $e->getMessage();
    }
```

- `$connection` Es la variable que usaremos para poder trabajar en el resto de la función.
- `new PDO` Es el indicador de que se abre una nueva conexión PDO.
- `mysql` Indica el sistema de gestión de datos que se utilizará. A la que se alsiganarán las variables de antes para completar la información.

Despues de esto le decimos como manejar errores:

```
$connection -> setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);
```

- `setAttribute` Es el método por el cual podemos modificar los atributos de la concexión.
- `PDO::ATTR_ERRMODE` Especifica el modo en que PDO manejará los errores.
- `PDO::ERRMODE_EXCEPTION` Indica que PDO debe lanzar una excepción si ocurre un error.

```
return $connection
```

Lo que devuelve la conexión en caso de ser positiva para proceder con la petición.

Bloque catch

Se recupera la excepción con PDOException y se guarda en la variable `$e` y con echo imprimimos el mensaje a través de `getMessage()`.

Crear DB

```
function create_DB ($connection){
    $sql="CREATE DATABASE IF NOT EXISTS DONACION";
    executeQuery($connection,$sql);
}
```

Creamos la petición y la guardamos en una variable. Por último con la función creada por nosotros, le pasamos la concexion y la peticion se ejecuta. Esta estructura será ña utilizada en cualquiera de las otras peticiones a la DB.

Ejecucion de petición

```
function executeQuery($connection,$sql){
    try{
        $connection->query($sql);
    }catch(PDOException $e){
        echo $e->getMessage();
    }
}
```

Debemos pasarle la variable con la conexión y petición y con `$connection->query($sql);` se ejecuta la peticion. Esto dentro de una estructura try/catch como la que se hizo antes.

Procesado en las páginas

La estructura que a de seguir en cada página en la que debamos interactuar con un formulario se divide la página en 2 secciones para el php.

Creacion en el server

En la página principal se debe crear la DB y las tablas, al menos la primera vez. Por lo que es omportante en las funciones encargadas de estas creaciones, poner un IF NOT EXISTS

```
$connection=getConnection();
create_DB($connection);
select_DB($connection);
create_table_admin($connection);
create_table_donantes($connection);
create_table_historico($connection);
```

Retroalimentación

En cada formulario que tengamos que realizar se validar los datos antes de que se envien. Para ello, las primeras lineas en el body deben hacer referencia a esto:

```
<?php include_once ("style/header.html");
$mensaje="";
if($_SERVER['REQUEST_METHOD']==="POST" && isset($_POST['submit'])){
    $nombre= validate($_POST['name']);
    $apellidos=validate($_POST['apellidos']);
    $age=validate($_POST['age']);
    $grp_sangre=validate($_POST['grp_sangre']);
    $cod_postal=validate($_POST['cod_postal']);
    $tlf=validate($_POST['tlf']);

    $connection = getConnection();
    select_DB($connection);
}
```

- En el caso de querer incluir cualquier efecto estético como puede ser `include_once ("style/header.html");` debe realizarse antes también.
- Debemos checkear cuando carguemos la página si hay algo cargado con un if, lo normal es que en este momento este vacio. En el if filtramos por el tipo de formulario (get o post) y que se haya enviado.

```
if($_SERVER['REQUEST_METHOD']==="POST" && isset($_POST['submit']))
```

Para el primer paso(el tipo de formulario) usamos la variable `$_SERVER` para acceder al método de envio. Y con `isset($_POST)` sabemos si se ha activado el botón de submit.

En el caso de entrar en el if procedemos a validar todos los campos que queramos. Para ello podemos crear otro documento con todas las validaciones (también en libs), en este caso debemos incluir ese documento (antes de cualquier código). Cada campo validado se guarda en una variable que se reconzca para poder pasar las a la DB.

```
$conexion = get_conexion();
select_DB($conexion);
alta_usuario($conexion, $nombre, $apellidos, $edad, $provincia);
$conexion->close();
```

Primero abrimos conexión y la guardamos en la variable. Acto seguido llamamos a la funcion para conectarnos a la DB con esa conexión. Por último llamamos a la funcion que realice la peticion a la DB que busquemos, pasandole la conexion y los datos validados. Siempre cerrar la conexión a la base de datos, se puede hacer a través de una función o en cada documento.

Una vez tengamos esta parte lista podremos empezar con el formulario. La parte de PHP de los formularios se centra en la primera línea, en la pripia etiqueta del Form.

```
<Form method="post" action="<?php echo
htmlspecialchars($_SERVER['PHP_SELF']);?>">
```

En ella se indica el metodo por el que se va a enviar el formulario. En la segunda parte se referencia el donde se va a enviar. junto con `htmlspeciachars` para evitar problemas con código malicioso. y con la supervariable `$_SERVER`.

POO

Base de datos

En un documento php dentro de la carptea de lib que luego debemos linkear en cada uno de los otros doc con un `Include_once("libs/data_base.php");`

Conexión

```
function get_conexion(){
    $conexion = new mysqli('db','root','test');
    if($conexion->connect_errno != null){
        die("Fallo en la conexion: ".$conexion->connect_error." con número
".$conexion->connect_errno);
    }
    return $conexion;
}
```

En la primera línea le indicamos el sistema que se usara `new mysql` y le pasamos la información de la DB (nombre, usuario y contraseña) `('db','root','test')`.

En el caso de que devuleva desde la variable un error, debemos recogerlo y mostrarlo. Con un if, cuya condición sea que el error sea diferente a nulo, hacemos que imprima el mensage de error y que termine el evento usando `die`.

```
($conexion->connect_errno != null)
```

Crear DB

Para crear la DB debemos crear una función que pueda trabajar con las consultas a mysql. La estructura de esta función será usada para crear tablas, y otras consultas, solo cambiarán las variables que necesitan y la estructura interna.

```
function crear_DB($conexion){
    $sql = "Create database if not exists tienda";
    ejecutar_consulta($conexion, $sql);
}
```

Para crear esta función debemos facilitarle la conexión que hemos creada anteriormente. Una vez dentro crearemos una variable con la consulta deseada y llamaremos a nuestra función para ejecutarla.

Función ejecutar consultas

```
function ejecutar_consulta($conexion, $sql){
    $resultado = $conexion->query($sql);
    if ($resultado == false){
        die($conexion->error);
    }
    return $resultado;
}
```

Es necesario pasar la consulta y la conexión. Con la conexión ejecutamos el método query, pasandole la consulta. Y para manejar los errores con un of nos aseguramos que el evento es igual a false debemos matar el proceso e imprimir el error.

Procesado en las páginas

Creacion en el server

```
$conexion=get_conexion();
crear_DB($conexion);
select_DB($conexion);
create_user_table($conexion);
```

Parq iniciar la DB en el servidor debemos idicarselo en la primera página que se cargue. Es importante que en las funciones en las que la consulta implica la creación de estructuras, debemos asegurarnos de que especificamos el `IF NOT EXISTS` para que no creemos cada vez que iniciamos o nos alerte de error.

Retroalimentación

Igual que PDO.

Editar DB

En el caso de que queramos editar un elemtneo de nuestra base de datos, el usuario debe ver cargados los datos que ya tendríaos guardados para poder editarlos. En ese caso se añade una capa más de complejidad.

Para prepar la página inicial debemos generar una estructura php antes del código html. Debemos indicar y guardar en una variable la conexión a la DB, asi como asignar variables vacias a todos los elements que debe mostrar y editar el usuario.

Con una estructura if/else comprobamos si el usuario a mandado la informacion del registro. En el momento que se inicia esa pagina por primera vez esto va a ser negativo asi que saltará al else. Pero una vez editado si que entrará en esta parte que es exactamente la misma que la de añadir una persona (guardar en variables los valores validados) y se guardan con nuestra función de editar_user.

¿Pero que pasa la primera vez que se carga la página? Al entrar al else se encuentra con este código:

```
if(isset($_GET["id"])){
    $id_user=$_GET["id"];
    $user = buscar_usuario($conexion, $id_user);
    if($user->num_rows >0){
        $row=$user->fetch_assoc();
        $id_user=$row['id'];
        $nombre=$row['nombre'];
        $apellidos=$row['apellidos'];
        $edad=$row['edad'];
        $provincia=$row['provincia'];
    }
}else{
    $id_user=0;
    $nombre="";
    $apellidos="";
    $edad=0;
    $provincia="corunha";
}
```

En esta sección el if interior se asegura de recibir (por GET), en caso positivo busca con nuestra función el usuario y lo asigna a una variable. Con fetch dividimos por filas y reasignamos las variables del principio.

Por último, en el propio formulario en el apartado valor le asignamos el homónimo.