

Bases de Datos + PHP

PDO

Vamos a consultar una base de datos

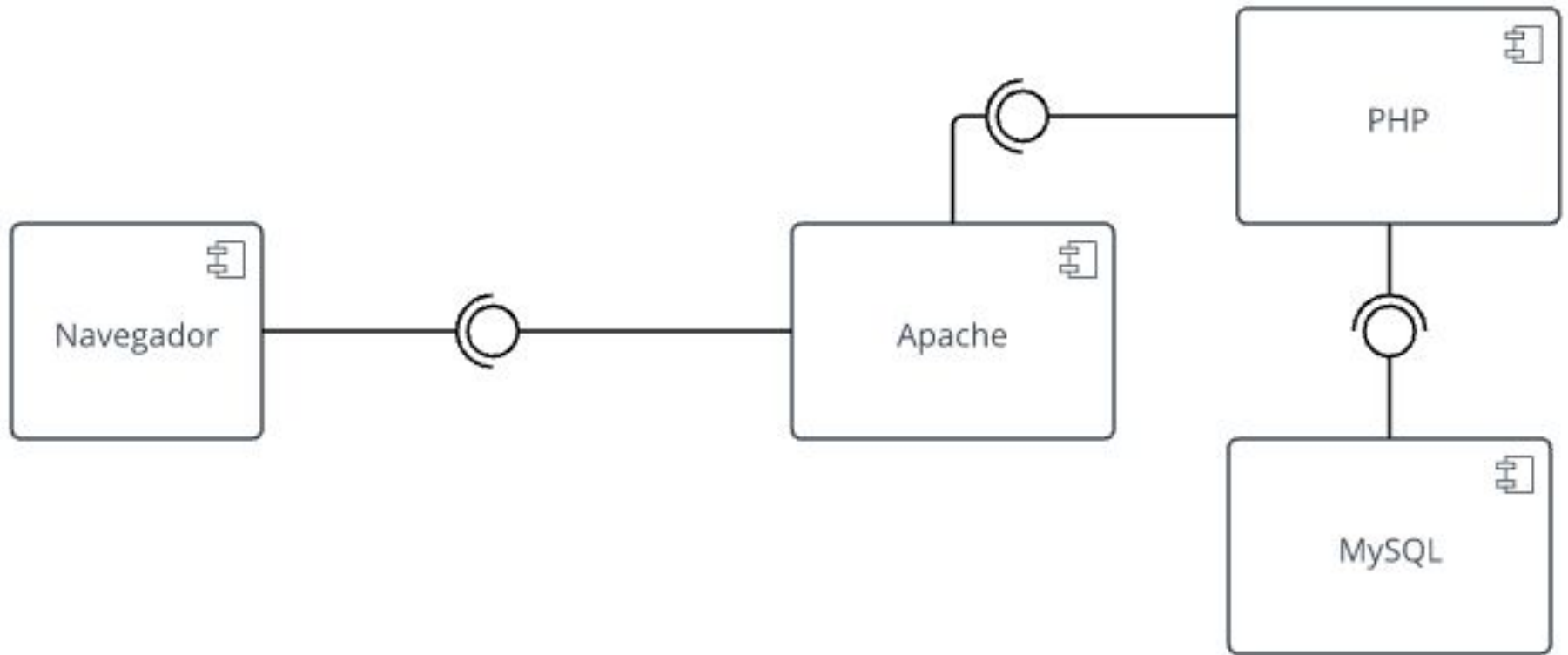
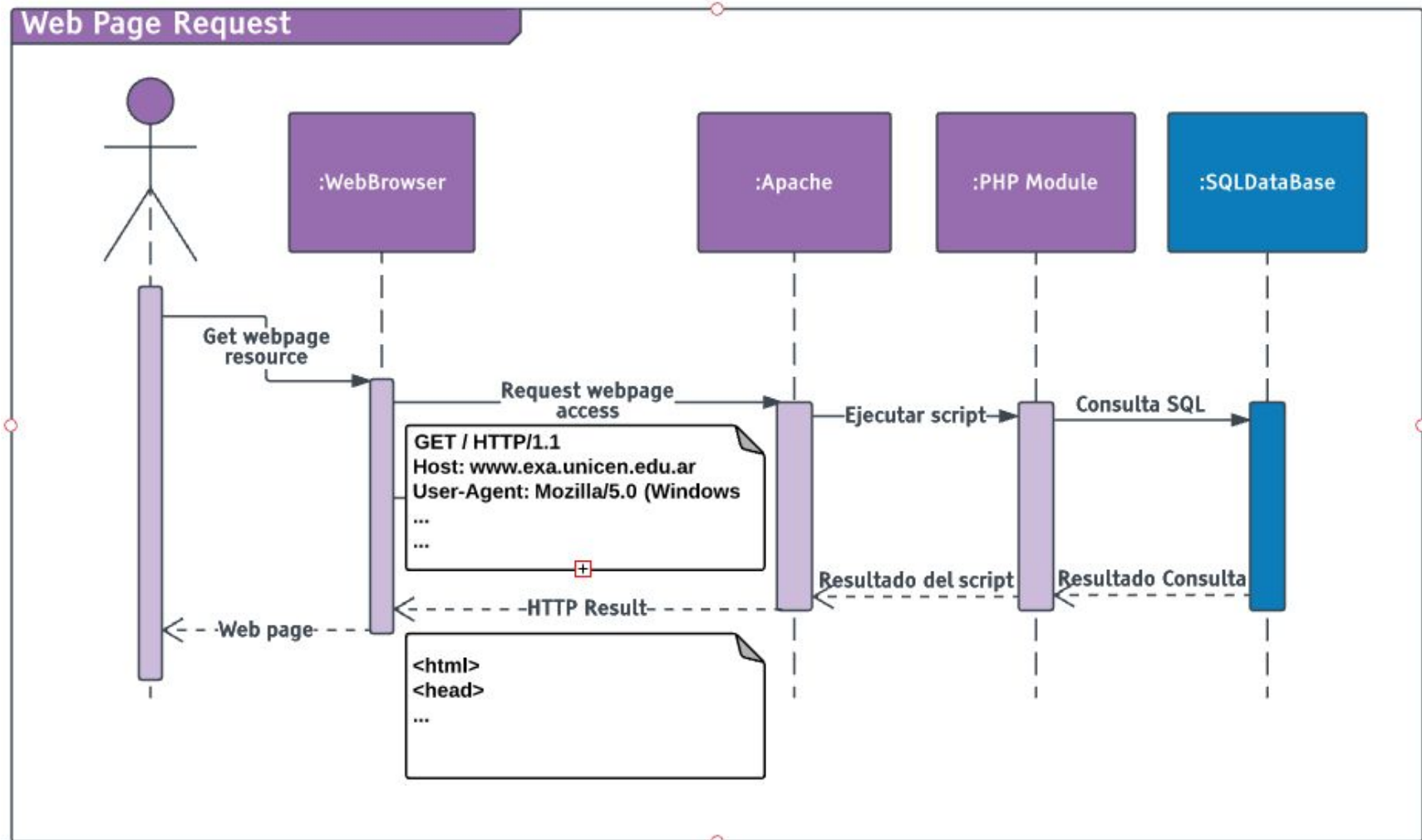


Diagrama de Secuencia con Base de Datos

Agregamos una base de datos



¿Cómo nos conectamos a la base de datos?

Usamos funciones específicas para conectarnos:

1. **Abrimos** una conexión
2. **Enviamos** la consulta y nos devuelve el resultado
3. **Procesamos** los datos para generar el HTML
4. **Cerramos** la conexión

¿Cómo se haría en MySQL?

Funciones de MySQL en PHP

Existen funciones específicas en php para acceder a bases de datos MySQL:

1. **Abrimos** una conexión

```
$link = mysqli_connect($host, $user, $passwd, $db);
```

2. **Enviamos** la consulta y nos devuelve el resultado

```
$result = mysqli_query($link, 'SELECT * WHERE 1=1');  
$arreglo = mysqli_fetch_all($result, MYSQLI_ASSOC);
```

3. **Procesamos** los datos para generar el HTML

```
foreach($arreglo as $value) {  
    echo $value['campo'];  
}
```

4. **Cerramos** la conexión

```
mysqli_close($link);
```

¿Qué pasa si el cliente quiere cambiar la DB?

El cliente viene y nos dice:

“Necesitamos migrar la base a Oracle”

- Habíamos creado la aplicación para que funcione con MySQL.
- Las bases de datos tienen pequeñas diferencias en el código SQL que entienden (algunas no tan pequeñas).
- Vamos a tener que revisar nuestro código y reescribir cada consulta.

¿Cuál es nuestra cara?



¿Se les ocurre que se puede hacer?

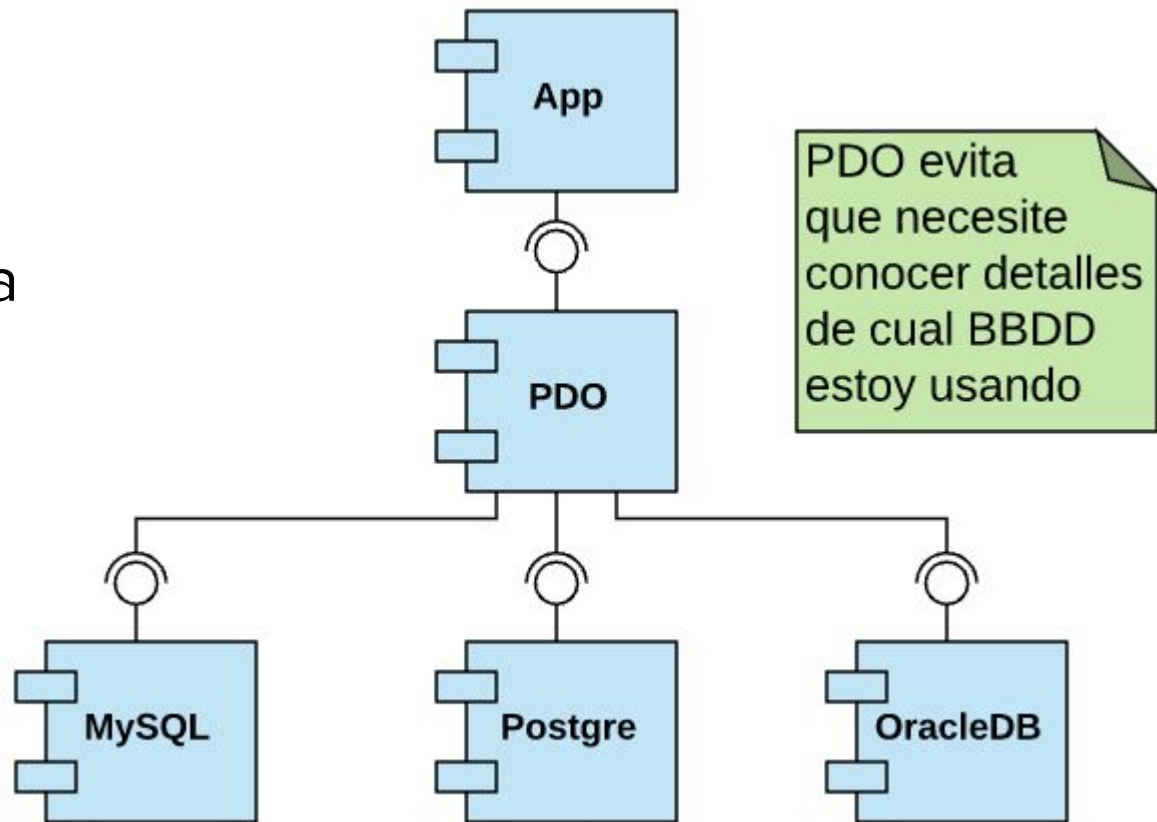


PHP Data Objects (PDO)

¿Qué es una DA abstraction layer?

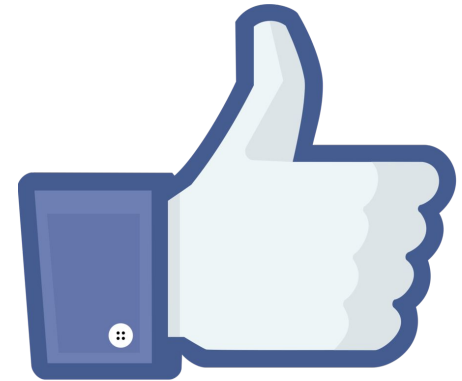
Data Access Abstraction Layer hace que escribamos una vez el código y funcione para cualquier base de datos.

Funciona como un INTERMEDIARIO entre la **lógica** y los **datos** de nuestra aplicación.



PDO: PHP Data Objects

- Herramienta de acceso a bases de datos (Interface).
- Funciona en PHP 5.1 y superiores.
- PDO es compilado: Más rápido.
- PHP soporta la mayoría de las bases de datos conocidas:
 - MySQL
 - SQLite (se guarda en archivos)
 - MSSql
 - etc...



Cada controlador de bases de datos que implemente la interfaz PDO puede exponer características específicas de la base de datos.

¿Cómo se si tengo instalado PDO?

1. Crear un archivo php con: `<?php phpinfo(); ?>`
2. Abrir con el explorador y buscar las siguientes líneas:

PDO

| | |
|-------------|---------|
| PDO support | enabled |
| PDO drivers | mysql |

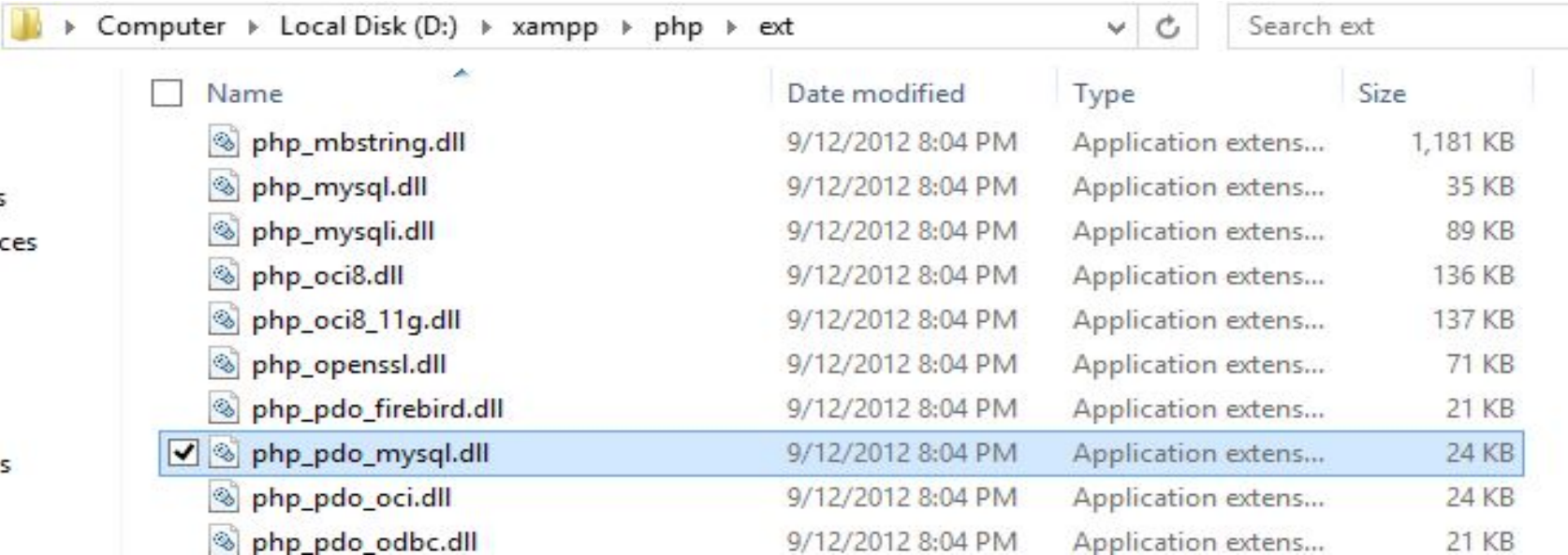
pdo_mysql

| | |
|----------------------|---|
| PDO Driver for MySQL | enabled |
| Client API version | mysqlnd 5.0.11-dev - 20120503 - \$Id: 3c688b6bbc30d36af3ac34fdd4b7b5b787fe5555 \$ |

3. Si están, ya puedo usar PDO con MySQL

Si no aparece ahí...

- Verificar tener PHP 5+ instalado (xampp lo trae)
- Ir a la carpeta xampp/php/ext y ver que tenga:



Computer > Local Disk (D:) > xampp > php > ext

| <input type="checkbox"/> | Name | Date modified | Type | Size |
|-------------------------------------|----------------------|-------------------|-----------------------|----------|
| <input type="checkbox"/> | php_mbstring.dll | 9/12/2012 8:04 PM | Application extens... | 1,181 KB |
| <input type="checkbox"/> | php_mysql.dll | 9/12/2012 8:04 PM | Application extens... | 35 KB |
| <input type="checkbox"/> | php_mysqli.dll | 9/12/2012 8:04 PM | Application extens... | 89 KB |
| <input type="checkbox"/> | php_oci8.dll | 9/12/2012 8:04 PM | Application extens... | 136 KB |
| <input type="checkbox"/> | php_oci8_11g.dll | 9/12/2012 8:04 PM | Application extens... | 137 KB |
| <input type="checkbox"/> | php_openssl.dll | 9/12/2012 8:04 PM | Application extens... | 71 KB |
| <input type="checkbox"/> | php_pdo_firebird.dll | 9/12/2012 8:04 PM | Application extens... | 21 KB |
| <input checked="" type="checkbox"/> | php_pdo_mysql.dll | 9/12/2012 8:04 PM | Application extens... | 24 KB |
| <input type="checkbox"/> | php_pdo_oci.dll | 9/12/2012 8:04 PM | Application extens... | 24 KB |
| <input type="checkbox"/> | php_pdo_odbc.dll | 9/12/2012 8:04 PM | Application extens... | 21 KB |

Configuración

- Verificar que en el php.ini tenga SIN comentar la línea del driver de MySQL.
- Si cambiaron el archivo, **guarden** y **reinicien** Apache.

```
1006 extension=php_mysql.dll
1007 ;extension=php_oci8.dll      ; Use with O
1008 ;extension=php_oci8_11g.dll  ; Use with O
1009 ;extension=php_openssl.dll
1010 ;extension=php_pdo_firebird.dll
1011 extension=php_pdo_mysql.dll
1012 ;extension=php_pdo_oci.dll
1013 ;extension=php_pdo_odbc.dll
1014 ;extension=php_pdo_pgsql.dll
1015 extension=php_pdo_sqlite.dll
```

Nuestra Aplicación Web

Ejemplo

TODO LIST APP

Vamos a construir una aplicación que nos permita administrar una lista de tareas.

(con persistencia en un BD)



Requerimientos funcionales

- Un usuario puede **ver** su lista de tareas
- Un usuario puede **agregar** tareas a su lista
- Un usuario puede **eliminar** tareas de su lista
- Un usuario puede **marcar** como completada una tarea

ToDo List App

¿Cómo va a estar formada cada **tarea**?

- título
- descripción
- prioridad (del 1 al 5)
- finalizada (booleano)

TODO LIST APP

Vamos a construirlo paso a paso en 3 iteraciones:

1. Ver tareas
2. Crear tareas
3. Eliminar tareas

En cada iteración el mínimo código necesario.



Versionado

Vamos a versionar nuestra app

GIT

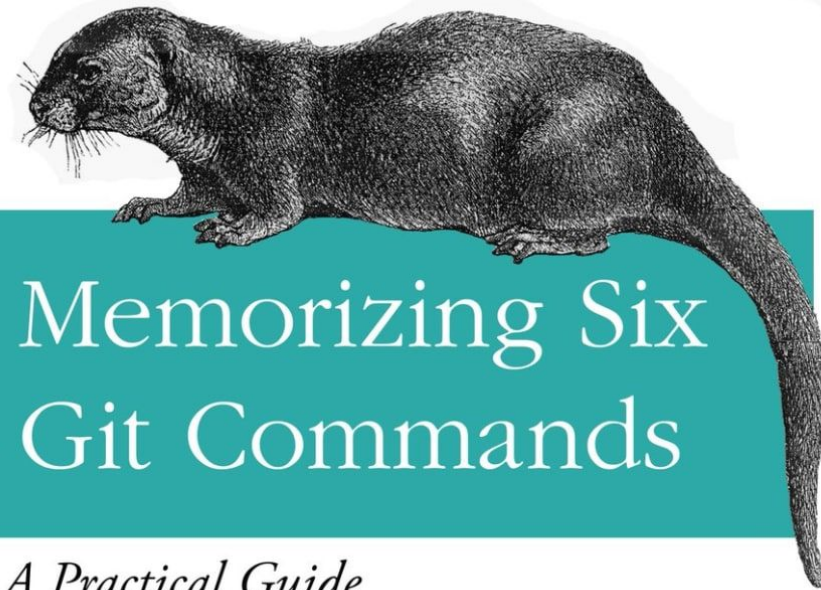
Antes de empezar a programar:

- `pull <remote> <branch>`

Después de programar:

- `add <files>`
- `commit -m <message>`
- `pull <remote> <branch>`
- `push <remote> <branch>`
- `status`

The popular approach to version control



A Practical Guide

O RLY?

@ThePracticalDev

1ra Iteración



Pensemos...

¿Cómo va a ser el **home** de nuestra app?

1er Iteración

- **Título**
- **Formulario**
- **Lista de Tareas**

```
<h1>Lista de Tareas</h1>
```

```
<form>
```

```
...
```

```
</form>
```

```
<ul>
```

```
  <li>...</li>
```

```
  <li>...</li>
```

```
</ul>
```

CHALLENGE ACCEPTED



LET'S CODE IT

**A MEDIDA QUE DAMOS LAS CLASES DEJAMOS ACA
LINKS AL COMMIT DE GIT HASTA ESTE PUNTO**

TRES ARROYOS:

<https://gitlab.com/unicen/Web2/livecoding2020/tres-arroyos/todo-list/-/commit/1232497feac304a2285d9cde6343a148306e683b>

BOLIVAR: <https://gitlab.com/unicen/Web2/livecoding2019/bolivar/todo-list/tree/9fa0e6151f25b4d5ccee68d2fa61fae49f6ab483>

TANDIL:

<https://gitlab.com/unicen/Web2/LiveCoding2018/Tandil/livecoding/commit/9196dbd51d614cd5663f37eeaa214495783919c2>

RAUCH: <https://gitlab.com/unicen/Web2/livecoding2020/rauch/todolist/-/commit/aa1e16bbf7486457825bb3f5d9c387b2ae6fb7e9>

GIT: Salvemos el trabajo

- `git status`
- `git add`
- `git commit -m "mensaje"`
- `git push`



¿De dónde sacamos los datos?

Ya tenemos el HTML, pero ¿de dónde sacamos los datos?



BASE DE DATOS

Vamos a crear una tabla, y leer los datos desde ahí.

¿Qué campos tendrá esa tabla?

Modelo de Entidad-Relaciones (MER)

Pensemos el modelo de la aplicación:



Identificador único de tarea

- clave primaria
- autoincrement

Genera id **automático e incremental**.
Nos olvidamos de manejar el id de las tareas!

Creando la tabla

DEMO

- Entramos a <http://localhost/phpmyadmin/>
- Creamos una BBDD “db_tareas”
- Creamos la tabla tarea con:
 - id_tarea int AI PRIMARY
 - titulo varchar(30)
 - descripcion varchar(250) NULL
 - prioridad int
 - finalizada boolean DEFAULT 0

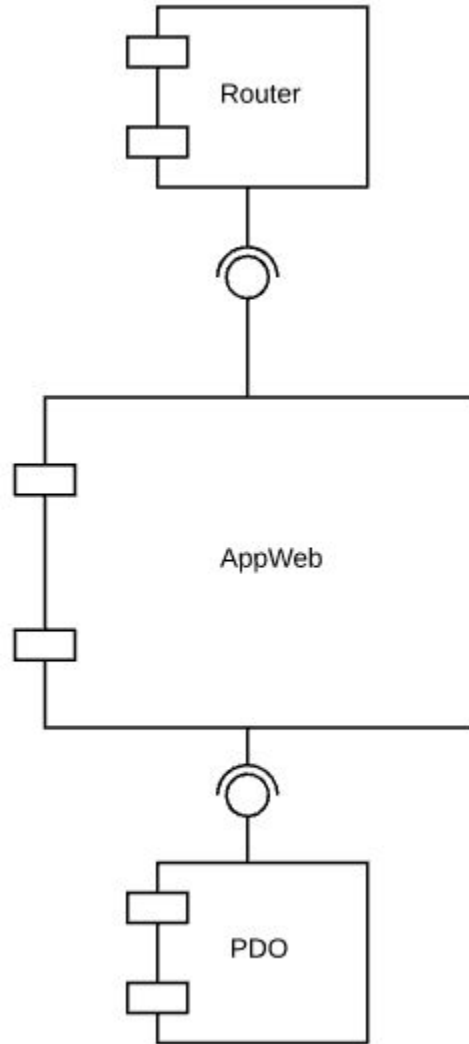
| | # | Nombre | Tipo | Cotejamiento | Atributos | Nulo | Predeterminado | Extra |
|--------------------------|---|---|--------------|-------------------|-----------|------|----------------|----------------|
| <input type="checkbox"/> | 1 | id_tarea  | int(11) | | | No | Ninguna | AUTO_INCREMENT |
| <input type="checkbox"/> | 2 | titulo | varchar(30) | latin1_swedish_ci | | No | Ninguna | |
| <input type="checkbox"/> | 3 | descripcion | varchar(200) | latin1_swedish_ci | | Sí | NULL | |
| <input type="checkbox"/> | 4 | finalizada | tinyint(1) | | | No | 0 | |

Obtengamos las tareas

¿Cómo hacemos la consulta SQL?



Arquitectura (hasta ahora)



¿Cómo me conecto a la DB?

Creamos una nueva conexión:

- servidor: **localhost**
- base de datos: **db_tareas**
- usuario: **root**
- contraseña: **root** *(en xampp generalmente es vacía)*

```
$db = new PDO( 'mysql:host=localhost;'
    . 'dbname=db_tareas;charset=utf8'
    , 'root' , '' );
```

¿Dónde agregamos esta línea?

Al comienzo de nuestro programa PHP

getTareas() - SELECT

prepare(\$sqlQuery)

permite la creación de una sentencia para su posterior uso:

```
$sentencia = $db->prepare( "select * from tarea");
```

execute(\$array)

ejecuta la sentencia que tenemos preparada:

```
$sentencia->execute();
```

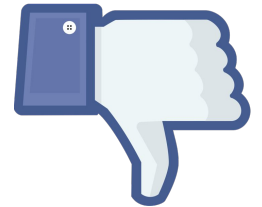
getTareas() - SELECT

fetch()

Itera sobre las tuplas seleccionadas.

```
while($fila = $sentencia->fetch(PDO::FETCH_OBJ)) {  
    echo '<li>'.$fila->título.  
    ':'.$fila->descripcion.'</li>';  
}
```

Aca mezclamos HTML, PHP y SQL!



Conviene separar a el código que nos devuelva un arreglo.
Así la obtención de datos es independiente y no importa si
viene de una BBDD, archivo u otra fuente.



getTareas() - SELECT

fetchALL()

Devuelve un array con las tuplas seleccionadas.

tareas.php

```
1  <?php
2  require_once 'db.php';
3
4  function listarTareas() {
5      // obtiene las tareas de la BD
6      $tareas = getTareas();
7
8      echo "<ul>";
9      foreach($tareas as $tarea) {
10         $echo = '<li class="tarea">' . $tarea->titulo . '</li>';
11     }
12     echo "</ul>";
13 }
14
```

db.php

```
1  <?php
2
3  function getTareas() {
4      $db = new PDO('mysql:host=localhost;
5                  . 'dbname=db_tareas;charset=utf8','root
6
7      $sentencia = $db->prepare("SELECT * FROM tareas
8      $sentencia->execute();
9      $tareas = $sentencia->fetchAll(PDO::FETCH_OBJ);
10
11     return $tareas;
12 }
```


Resumen: Obtener Tareas PDO

1. **Abrimos** una conexión

```
$db = new PDO('mysql:host=localhost;  
               . 'dbname=db_tareas;charset=utf8'  
               , 'root', 'root');
```

2. **Enviamos** la consulta y nos devuelve el resultado

```
$sentencia = $db->prepare( "select * from tarea");  
$sentencia->execute();  
$tareas = $sentencia->fetchAll(PDO::FETCH_OBJ);
```

3. **Procesamos** los datos para generar el HTML

```
foreach($tareas as $tarea) {  
    echo $tarea->nombre;  
}
```

4. **Cerramos** la conexión

En PDO no es necesario cerrar la conexión

Parámetros - PDO::FETCH_*

- *PDO::FETCH_ASSOC*: devuelve un array indexado por los nombres de las columnas del conjunto de resultados.
- *PDO::FETCH_OBJ*: devuelve un objeto anónimo con nombres de propiedades que se corresponden a los nombres de las columnas devueltas en el conjunto de resultados.
- *PDO::FETCH_BOTH*: devuelve un array indexado tanto por nombre de columna, como numéricamente.
- *PDO::FETCH_NUM*: devuelve un array indexado por el número de columna tal como fue devuelto en el conjunto de resultados, comenzando por la columna 0.

getTareas() - Traer todas las filas

- *fetchAll()* trae todas las filas a una tabla:

```
$tareas = $sentencia->fetchAll(  
    PDO::FETCH_GROUP  
    | PDO::FETCH_UNIQUE | PDO::FETCH_ASSOC);
```

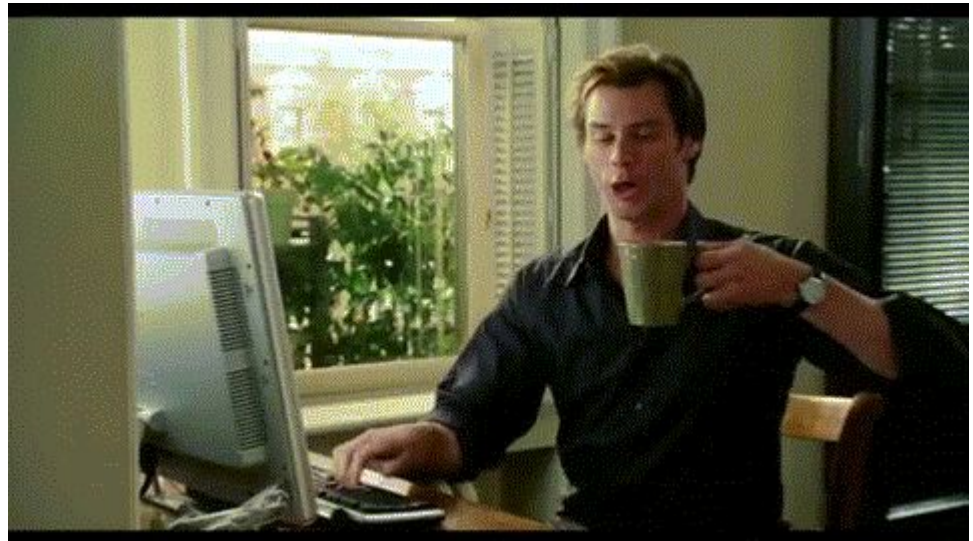
- **PDO::FETCH_GROUP** agrupa las filas usando la primer columna como índice.
- **PDO::FETCH_UNIQUE** No hace un arreglo de arreglos al ser única cada fila por grupo.

Resultado



BOLIVAR: <https://gitlab.com/unicen/Web2/livecoding2019/bolivar/todo-list/commit/9fa0e6151f25b4d5ccee68d2fa61fae49f6ab483>
RAUCH: <https://gitlab.com/unicen/Web2/livecoding2020/rauch/todolist/-/commit/c16f688961fb93b820635a780e9396865bd86d46>
TA: <https://gitlab.com/unicen/Web2/livecoding2020/tres-arroyos/todo-list/-/commit/c6aa627c8a64b0134f39074d911783c2c20061e0>

2da Iteración



¿Cómo piensan qué podemos hacer el INSERT de una tarea?



agregarTarea() - INSERT (Exec)

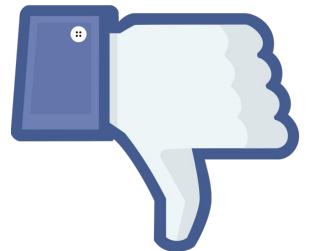
`exec($sqlQuery)`

ejecuta un comando SQL:

`$db->exec("INSERT INTO tarea(titulo)". "VALUES('".$tarea."')");`

`$db->lastInsertId()` nos devuelve el **id** del último elemento insertado:

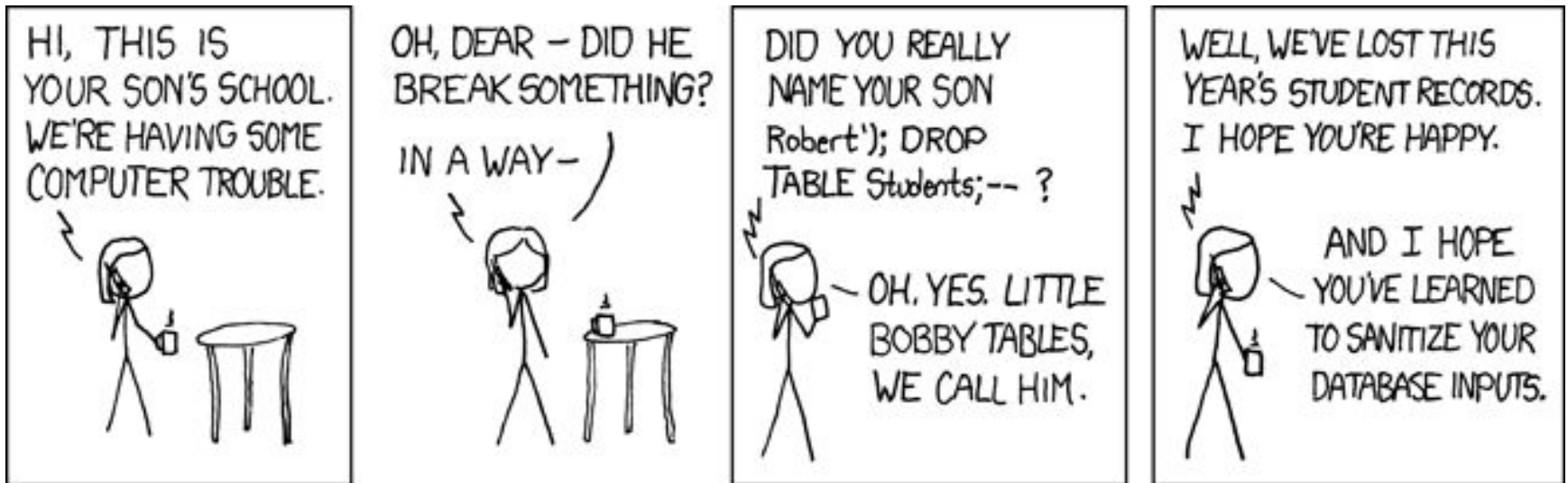
Si bien se puede usar **exec** no es recomendado porque no previene ataques de SQL Injection



Inyección SQL

¿Qué pasa si el usuario nos pasa el siguiente nombre para una tarea?

Tarea'; DROP TABLE Usuario; SELECT '

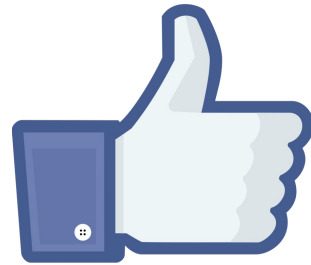


agregarTarea() - INSERT (Prepare)

prepare(\$sqlQuery)

permite la creación de una sentencia “parametrizada” para su posterior uso:

```
$sentencia = $db->prepare(  
    "INSERT INTO tarea(titulo) VALUES(?)");
```



execute(\$array)

ejecuta la sentencia con los parámetros:

```
$sentencia->execute(array('Hacer La página de Web'));  
$sentencia->execute(array("Estudiar otras materias"));
```

Paso de parámetros

- El paso de parámetros le permite a PDO controlar y escapar las variables con contenido inseguro:

```
$sentencia->execute(  
    array("tarea"; DROP TABLE tarea; SELECT "');
```



- PDO escapará los caracteres problemáticos y el texto será insertado normalmente.

agregarTarea() - Último Id

- Podríamos hacer que agregarTarea() devuelva el último ID, para eso

\$db->lastInsertId()

- Nos devuelve el ***id*** del último elemento insertado.

Parámetros por nombre

- Los parámetros también pueden ser pasados de forma asociativa:

```
$sentencia = $this->db->prepare(  
    "INSERT INTO tarea(titulo, descripcion)"  
    ."VALUES(:titulo, :descripcion)");
```

```
$sentencia->execute(  
    array(":titulo"=>$tarea['titulo']  
        ,":descripcion"=>$tarea['descripcion']));
```

¿Cómo lo ejecuto?

¿Dónde va esto?

¿Cómo se ejecuta?

Una vez que una tarea es ingresada, podemos redireccionar devuelta al home de nuestra aplicación:

```
// Redirect browser
```

```
header("Location: /home");
```

Resultado



TANDIL: <https://gitlab.com/unicen/Web2/livecoding2020/tandil/tasks/-/commit/e5d6769ad4d058e669b40af829d30d43e1ebe420>

BOLIVAR: <https://gitlab.com/unicen/Web2/livecoding2019/bolivar/todo-list/commit/98d767aef307425906d6f993572851f46b7e1cc3>

RAUCH: <https://gitlab.com/unicen/Web2/livecoding2020/rauch/todolist/-/commit/c549afd049ae7f20759db1940117ce954694e5a8>

TA: <https://gitlab.com/unicen/Web2/livecoding2020/tres-arroyos/todo-list/-/commit/239260649cc2e9c2c7f4e95ca4d87266864fdda8>

3ra Iteración



¿Cómo hacemos el borrado de una tarea?



Borrar una tarea

- Ejecuto un query que borra una tarea por ID.

```
$sentencia = $db->prepare(  
    "delete from tarea where id=?");
```

```
$sentencia->execute(array($id_tarea));
```

Nuevo requerimiento

- Tenemos que agregar la posibilidad de marcar una tarea como realizada.
- En caso de estar completa se tiene que mostrar en la lista pero tachada.

¿Qué sentencia de SQL voy a usar?

finalizarTarea() - UPDATE

- La actualización es similar a la inserción:

```
$sentencia = $db->prepare(  
    "UPDATE tarea SET Finalizada=1  
    ." WHERE idTarea=?");  
$sentencia->execute(array($tarea['id']));
```

- *\$sentencia*->rowCount() nos dice cuántas filas fueron afectadas en la última ejecución. (También aplicable en *INSERT* o *DELETE*)

Resultado

Lista de Tareas

Tarea

✓ Dormir la siesta

Mente descansada vale doble.

✓ Estudiar

Leer los pocos apuntes que tomó mi compañero.

✓ Prácticos de Web

Finalizar todos los ejercicios de los practicos 1 al 3 para mañana.

Tarea

Título

Descripción

Agregar



TANDIL:

<https://gitlab.com/unicen/Web2/livecoding2020/tandil/tasks/-/commit/b6cd0b42539b9de72d85de5d67090e84bf15ad03>

Bolivar: <https://gitlab.com/unicen/Web2/livecoding2019/bolivar/todo-list/commit/eb0e4385a4bc6a0f0ab2e588248dd17e619c314a>

Transacciones

Escenario

- Supongamos un sistema de compras online.
- Antes de pagar el sistema consulta el saldo para chequear si es suficiente.

¿Qué pasa si un usuario intenta hacer 2 pagos a la vez?



Transacción

Una transacción es un conjunto de instrucciones SQL que se ejecutan como unidad:

- Se ejecutan todas o no se ejecuta ninguna.
- Si una transacción tiene éxito, todas las modificaciones de datos realizadas se guardan en la base de datos.
- Si una transacción falla los cambios no se guardarán.

ACID Atomicity, Consistency, Isolation, Durability

Las transacciones garantizan:

- **Atomicidad:** Asegura que la operación se ha realizado o no, y por lo tanto ante un fallo del sistema no puede quedar a medias.
- **Consistencia:** Sólo se pueden escribir datos válidos respetando los tipos de datos declarados y la integridad referencial.
- **Aislamiento:** Asegura que una operación no puede afectar a otras. Con esto se asegura que varias transacciones sobre la misma información sean independientes y no generen ningún tipo de error.
- **Durabilidad:** Cuando se completa una transacción con éxito los cambios se vuelven permanentes.

PDO

- En PDO una transacción comienza con:
 - `$db->beginTransaction();`
- Todas las operaciones siguientes serán ejecutadas en modo de transacción.
- La transacción se completa con:
 - `$db->commit();`
- O se deshacen los cambios con:
 - `$db->rollBack();`

Solución

```
$db->beginTransaction();  
$consulta = $db->prepare('SELECT saldo'  
    .'FROM usuario WHERE idUsuario = ?');  
$consulta->execute(array($idUsr));  
$saldo = $consulta->fetchColumn();  
if($saldo < $costo) return false;  
$saldo -= $costo;  
$consulta = $db->prepare('UPDATE saldo'  
    .'SET saldo=? WHERE idUsuario = ?');  
$consulta->execute(array($saldo, $idUsr));  
$db->commit();
```

Ejemplo - Continuación

- Ahora queremos asegurar que un monto se debite si y sólo si la compra se realiza.

Excepciones

- Una excepción es la indicación de un problema que ocurre durante la ejecución de un programa.
- Pueden ser capturadas encerrando las funciones que las producen en bloques try:

```
try{/*instrucciones que generan excepciones*/ }
```

```
catch(Exception $e){/* se ejecuta en caso de excepción*/}
```

- En PDO se habilitan con:

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

Try - Catch

```
try{
    $db->beginTransaction();

    /* instrucciones que generan excepciones */

    /* Si todo es correcto, los cambios son guardados. */
    $db->commit();
}catch(PDOException $ex){
    /* Si una transacción falla, deshace los cambios. */
    $db->rollBack();
    log($ex->getMessage());
}
```

Ejercicio

Quiero que al hacer click en una tarea me muestre (en otra página) los detalles de la misma (otros campos con más texto, etc).

Más Información

Documentación Oficial

<http://php.net/manual/en/book.pdo.php>

Php Every Day

<http://www.phpeveryday.com/articles/PDO-Activation-PHP-Data-Objects-Extension-P544.html>

Manual de referencia MySQL

<http://dev.mysql.com/doc/>