

Índice de contenidos

Tecnologías de almacenamiento de datos	01	05	Uso de conjuntos de datos para almacenar información
Conexiones con bases de datos	02	06	Actualización y eliminación de información almacenada
Recuperación de la información almacenada	03	07	Otros almacenes de datos
Publicación de información recuperada en páginas web	04		

INTRODUCCIÓN

Las aplicaciones web prácticamente no se pueden entender sin medios para el almacenamiento de la información persistente, más allá de técnicas de persistencia en el cliente.

Esto implica que el código puede estar en una constante interacción con distintas herramientas como las bases de datos, de forma que mantenemos la información para su posterior uso.



TECNOLOGÍAS DE ALMACENAMIENTO DE DATOS

TECNOLOGÍAS DE ALMACENAMIENTO DE DATOS

Cuando hablamos de tecnologías de almacenamiento de datos, hay varias opciones muy utilizadas, algunas locales, otras en la nube. Lo que diferenciará si es local o no radica en el lugar donde se almacenen los datos, ya sea en las instalaciones de la empresa (local/privado) o en las instalaciones de otra empresa (nube/pública).

Existen soluciones que hacen una mezcla, se las suele llamar nube híbrida.

TECNOLOGÍAS DE ALMACENAMIENTO DE DATOS

Ventajas y desventajas del almacenamiento privado:

- +Mayor control de la información
- +Acceso muy restringido
- +Sin dependencia de internet (suele ser más rápido)
- Costes de mantenimiento del hardware
- Dificultad en la gestión y escalabilidad
- Vulnerabilidad a desastres si no hay copias de seguridad

TECNOLOGÍAS DE ALMACENAMIENTO DE DATOS

Ventajas y desventajas del almacenamiento público (nube):

- +Sencillez de uso (zero configuration)
- +Costes de hardware inexistentes, solo por uso
- +Seguridad contra desastres o al menos recuperación configurada
- +Alta escalabilidad y flexibilidad
- Costes de uso pueden ser muy elevados (y recurrentes)
- Menor control de la información (y privacidad)
- Dependencia de internet

TECNOLOGÍAS DE ALMACENAMIENTO DE DATOS

Existen varios tipos de tecnologías de almacenamiento de datos, entre ellas destacamos:

- **Almacenamiento basado en ficheros** (JSON, XML, YAML, o TXT, por ejemplo): No se suele utilizar porque no existe una sincronización fácil con otros accesos a la aplicación, o se usa para cosas muy concretas.
- **Bases de datos relacionales (SQL):**
 - Ideales para datos relacionados
 - Con soporte para consultas en desde muchos lenguajes (PHP, por ejemplo)
 - Integración sencilla en diferentes frameworks

TECNOLOGÍAS DE ALMACENAMIENTO DE DATOS

Existen varios tipos de tecnologías de almacenamiento de datos, entre ellas destacamos:

- **Bases de datos no relacionales(NoSQL):**
 - Se usan para almacenes de datos no relacionados o semiestructurados.
 - Alta escalabilidad y flexibilidad
 - Sin soporte para consultas SQL o consultas complejas.
- **Almacenamiento en memoria:**
 - Sistemas de cacheo y almacenamiento temporal (Redis)
 - Muy rápido
 - Ideal para datos de sesión de usuario

TECNOLOGÍAS DE ALMACENAMIENTO DE DATOS

Existen varios tipos de tecnologías de almacenamiento de datos, entre ellas destacamos:

- **APIs de almacenamiento remoto:**
 - Conectan la aplicación con servicios externos de almacenamiento de datos
 - Firebase, Amazon S3, etc.
 - Gastos según uso.

TECNOLOGÍAS DE ALMACENAMIENTO DE DATOS

¿Cuál vamos a usar?

Dependerá de nuestra aplicación:

- Para datos estructurados: Bases de datos SQL (MySQL, PostgreSQL, etc.).
- Para datos no estructurados o con alta escalabilidad: NoSQL (MongoDB, DynamoDB, etc.).
- Para almacenamiento de archivos: Amazon S3, Google Cloud Storage o local.
- Para caching: Redis o Memcached.
- Para tiempo real: Firebase (o similares) o bases de datos en memoria.



CONEXIONES CON BASES DE DATOS

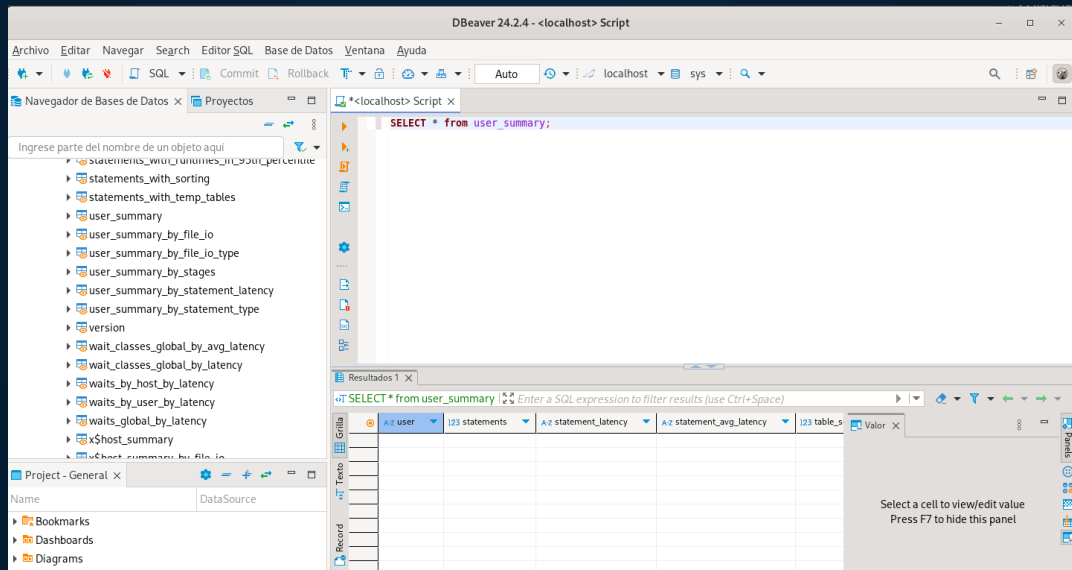
CONEXIONES CON BASES DE DATOS

Desde PHP hay varias formas de conectar con una base de datos relacional, una procedimental y otra orientada a objetos. En nuestro caso, solo vamos a utilizar la orientada a objetos, ya que los conocemos y sabemos usar.

Esta aproximación se llama PDO, y PHP normalmente utilizará bases de datos MySQL. Si no lo tenemos todavía, habrá que activar el módulo `mysqli` de `php.ini` o instalar los paquetes necesarios para que funcione en Linux.

CONEXIONES CON BASES DE DATOS

Puede ser útil tener algún software como DBeaver o PHPMyAdmin para revisar de forma más cómoda la información almacenada en las tablas.



CONEXIONES CON BASES DE DATOS

Podemos crear en una base de datos una tabla de prueba usuarios:

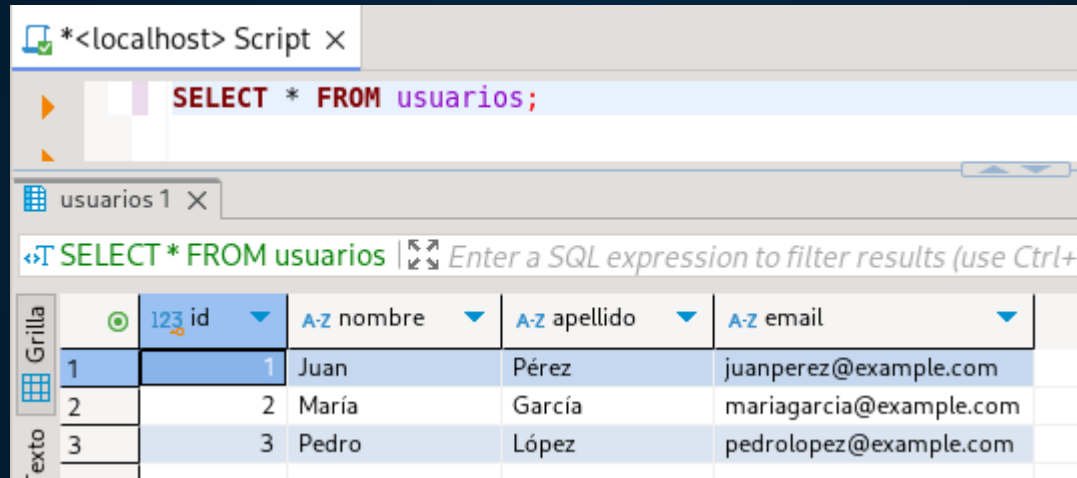
```
CREATE TABLE usuarios (  
  id INT AUTO_INCREMENT PRIMARY KEY,  
  nombre VARCHAR(50),  
  apellido VARCHAR(50),  
  email VARCHAR(100)  
);
```

E insertamos datos:

```
INSERT INTO usuarios (nombre, apellido, email)  
VALUES  
  ('Juan', 'Pérez', 'juanperez@example.com'),  
  ('María', 'García', 'mariagarcia@example.com'),  
  ('Pedro', 'López', 'pedrolopez@example.com');
```

CONEXIONES CON BASES DE DATOS

Ahora, si usamos DBeaver, podemos comprobar la información con un select simple:



The screenshot shows the DBeaver interface. At the top, a script editor window titled '*<localhost> Script x' contains the SQL query: `SELECT * FROM usuarios;`. Below the script editor, a results window titled 'usuarios 1 x' displays the query results in a table grid. The table has five columns: 'id', 'nombre', 'apellido', and 'email'. The first row is highlighted in blue and contains the values 1, Juan, Pérez, and juanperez@example.com. The second row contains 2, María, García, and mariagarcia@example.com. The third row contains 3, Pedro, López, and pedrolopez@example.com.

	id	nombre	apellido	email
1	1	Juan	Pérez	juanperez@example.com
2	2	María	García	mariagarcia@example.com
3	3	Pedro	López	pedrolopez@example.com

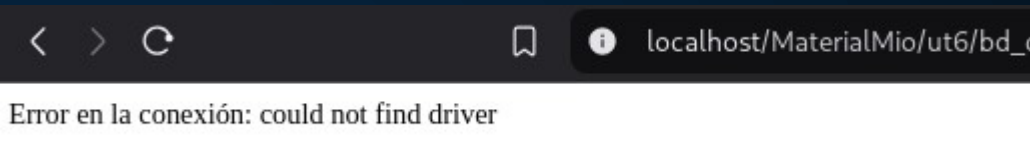
Vamos a intentar conectar a la base de datos desde PHP para más adelante usar la información almacenada en nuestra página.
Ver ejemplo conexión.

CONEXIONES CON BASES DE DATOS

Si al abrir el ejemplo nos encontramos con el mensaje de excepción que nos avisa de que falta el driver, nos tocará comprobar si tenemos el driver activado dentro del archivo `php.ini`.

Podemos ver qué archivo está activo con un `phpinfo()` o el comando `php -ini`.

Hay que buscar las extensiones y descomentar las dos de mysql en nuestro caso, si usasemos otra BD lo haríamos con la correspondiente.



```
extension=mysqli  
;extension=odbc  
;zend_extension=opcache  
;extension=pdo_dblib  
extension=pdo_mysql  
;extension=pdo_odbc  
;extension=pdo_pgsql  
;extension=pdo_sqlite  
;extension=pgsql
```

Por último, tendremos que reiniciar el servidor web. En linux lo podemos hacer con un `sudo systemctl restart apache2`

RECUPERACIÓN DE LA INFORMACIÓN ALMACENADA



RECUPERACIÓN DE LA INFORMACIÓN ALMACENADA

La recuperación de la información se refiere al proceso de obtener datos que han sido guardados previamente en una base de datos u otro sistema de almacenamiento. Este proceso es esencial para aplicaciones web, ya que permite mostrar información dinámica como registros de usuarios, productos, reportes, etc.

Además, no solo permite mostrarlo, si no que podremos trabajar con los datos que están en la BD.

RECUPERACIÓN DE LA INFORMACIÓN ALMACENADA

Hay que tener claro que necesitamos para recuperar la información:

- Conectar a una BD usando un driver como `pdo_mysql`
- Realizar una consulta
- Saber en qué formato se entrega la información solicitada, para poder manipularla.

En el caso de las conexiones, ya lo hemos visto, aunque también podemos usar `mysqli`, utilizando procedimientos.

Las consultas serán, en MySQL, consultas SQL como las que venimos haciendo desde el curso pasado, solo que hay que envolverlas en sintaxis PHP.

Por último, hay que saber cómo queremos los datos finales, si como una array o como un archivo JSON, por ejemplo.

RECUPERACIÓN DE LA INFORMACIÓN ALMACENADA

Ver ejemplo de consulta.

Si no queremos extraer una array con todos los registros, sino ir trabajando fila a fila, podemos utilizar el método `fetch()`.

Por ejemplo, podemos ir creando objetos fila a fila y meterlos en una array según leemos los resultados.

Ver ejemplo de consulta objeto.

RECUPERACIÓN DE LA INFORMACIÓN ALMACENADA

Además de las consultas normales, podemos realizar lo que se suele llamar consultas preparadas, que, utilizando parámetros, se utilizan valores dados por variables como partes de la consulta.

Estas consultas se usan para ejecuciones de SQL más seguras, eficientes y protegidas contra las inyecciones SQL. Son más rápidas cuando se usan múltiples veces, puesto que reutilizan el plan de ejecución.

Ver ejemplo consultas preparadas.

RECUPERACIÓN DE LA INFORMACIÓN ALMACENADA

Las consultas preparadas se ejecutan como hemos visto, usando execute, pero hay dos formas de asociar los datos a la consulta:

Característica	bindParam	bindValue
Vincula	Variable	Valor directo o variable
Momento de evaluación	En el execute	Al llamar a bindValue
Reutilización	Útil para variables	Útil para valores constantes
Valores directos	No	Sí
Acepta variables	Sí	Sí

PUBLICACIÓN DE INFORMACIÓN RECUPERADA EN PÁGINAS WEB



PUBLICACIÓN DE INFORMACIÓN RECUPERADA EN PÁGINAS WEB

Normalmente, la información recuperada de sistemas de almacenamiento se utiliza para una de dos cosas:

- Se muestra directamente en alguna página web
- Se procesa antes de mostrarse en alguna página web

Habitualmente se hace con el uso de tablas u otras estructuras HTML personalizadas (cards, por ejemplo).

Ejercicio:

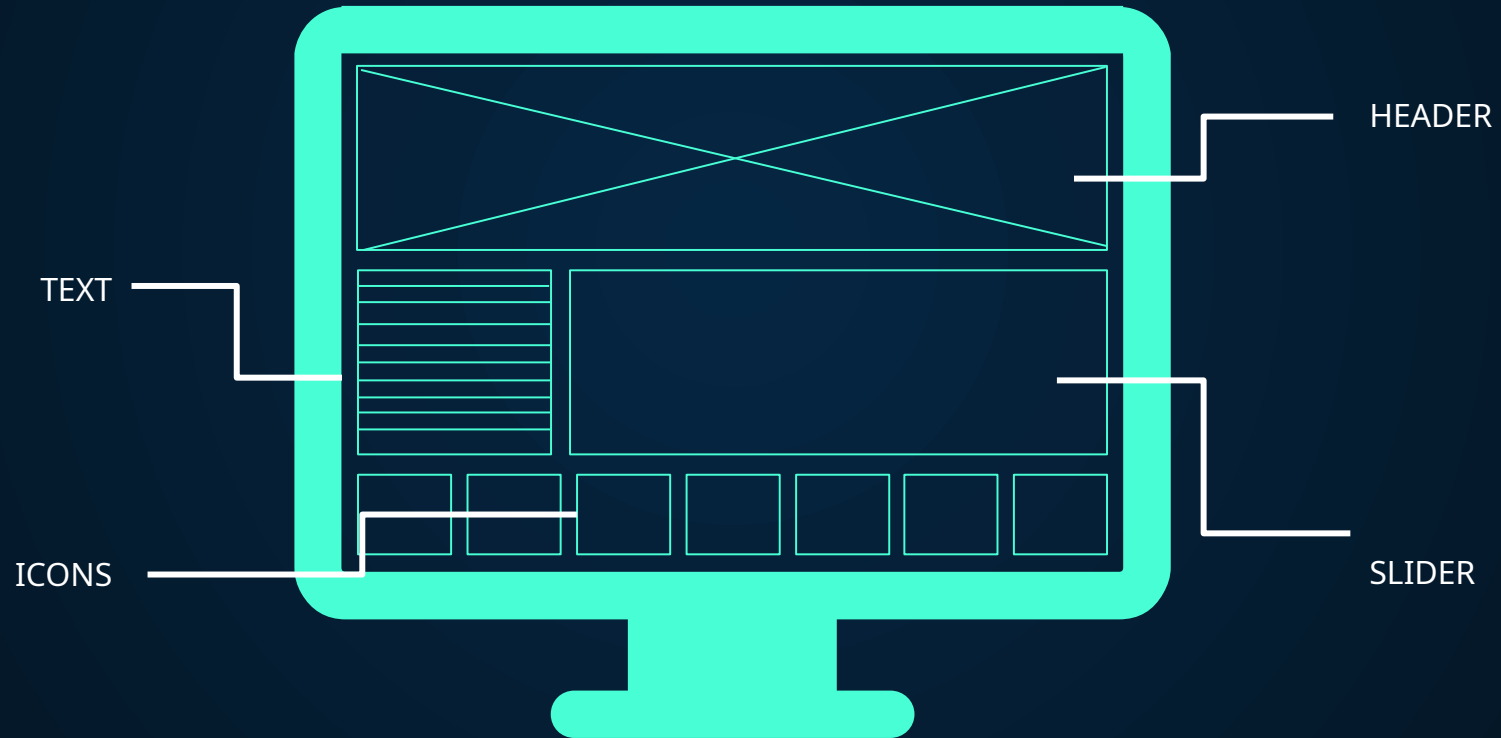
Basándote en los ejemplos de consulta, muestra la información de la tabla usuarios en una tabla.

PUBLICACIÓN DE INFORMACIÓN RECUPERADA EN PÁGINAS WEB

Ejercicio:

Basándote en los ejemplos y ejercicios, realiza un programa que sea capaz de paginar utilizando consultas preparadas, sesiones (para almacenar la información una vez solo) y enlaces a las páginas, de forma que en cada página se vean 10 registros en una tabla.

USO DE CONJUNTOS DE DATOS PARA ALMACENAR INFORMACIÓN



USO DE CONJUNTOS DE DATOS PARA ALMACENAR INFORMACIÓN

Podemos guardar la información de matrices u objetos en la base de datos utilizando las consultas preparadas, por ejemplo, de manera que se vayan intercambiando los valores de los parámetros a la hora de hacer las inserciones.

Ver ejemplo de inserción.

Cuando hacemos una inserción, actualización, o borrado, no obtenemos resultados de registros, pero sí podemos saber cuántas filas han sido afectadas usando el método `rowCount()`, de `PDOStatement`, al que podemos llamar al ejecutar la consulta. Sería el equivalente para el `SELECT` de realizar un `count()` del objeto que nos da `fetchAll()`.



ACTUALIZACIÓN Y ELIMINACIÓN DE INFORMACIÓN ALMACENADA

ACTUALIZACIÓN Y ELIMINACIÓN DE INFORMACIÓN ALMACENADA

Al igual que podemos recuperar información y almacenarla, también podemos actualizar y borrar la información.

La idea es utilizar, en general, consultas preparadas para realizar un borrado controlado, o una actualización controlada. Siempre usando filtros como WHERE.

Ver ejemplo de borrado.

SERIALIZACIÓN



SERIALIZACIÓN

La serialización es un proceso por el cual se convierte una estructura de datos o un objeto en una cadena (formato plano) que se puede almacenar o transmitir fácilmente. Esta cadena puede luego deserializarse para reconstruir la estructura de datos original.

La serialización es útil para guardar datos complejos en archivos, bases de datos o para enviarlos a través de la red.

Para serializar, se utiliza la función `serialize()`, que admite arrays, objetos y tipos básicos, mientras que para deserializar se utiliza `unserialize()`, devolviendo nuevamente la cadena en una estructura de datos original.

SERIALIZACIÓN

La idea es que guardemos los datos serializados en archivos, utilizando funciones como `file_put_contents("datos.txt", serialize($data))`, y los recuperamos usando algo como:

```
$contenido = file_get_contents("datos.txt");  
$data = unserialize($contenido);  
print_r($data);
```

Ver ejemplo de serialización.

SERIALIZACIÓN

Advertencias:

- No deberíamos deserializar datos que no sepamos realmente de quien vienen (podría ser malicioso).
- La versión de PHP puede hacer que las cadenas serializadas no sean compatibles.
- Podemos almacenar más de un objeto serializado en el mismo archivo usando delimitadores, o almacenando una estructura que contenga otros objetos/estructuras, o, incluso, si vamos “engordando” la estructura serializada leyendo, deserializando, añadiendo y volviendo a serializar y escribir.

Si queremos más interoperabilidad, sería conveniente utilizar algo como JSON para transmitir la información, cosa que podemos hacer con las funciones `json_encode()` y `json_decode()`.

Ver ejemplo codificado json.



OTROS ALMACENES DE DATOS

OTROS ALMACENES DE DATOS

Además de MySQL u otras bases de datos relacionales, podemos utilizar en nuestras aplicaciones web bases de datos NoSQL, como Firebase Firestore.

Esta base de datos permite almacenar y sincronizar datos en tiempo real y en la nube.

El modelo de datos NoSQL almacena datos en documentos organizados en colecciones, en el caso de Firestore, las colecciones son objetos JSON que pueden tener subcolecciones a su vez.

Para usar Firebase necesitaremos una clave API de google. Normalmente se utiliza con lenguajes que permiten programación asíncrona.