IAPWEB

TEMA 6 – ACCESO A BASES DE DATOS

EMPEZAR >



Powered by genially EDUCATION

< 00 >

Introducción

< 01 >

Configuración

< 02 >

PHPMyAdmin

< 03 >

Conexión a MySQL

< 04 >

SQLInjection

< 05 >

Transacciones

< 06 >

Ejercicios



Powered by genially EDUCATION

POR QUÉ USAR UNA BASE DE DATOS

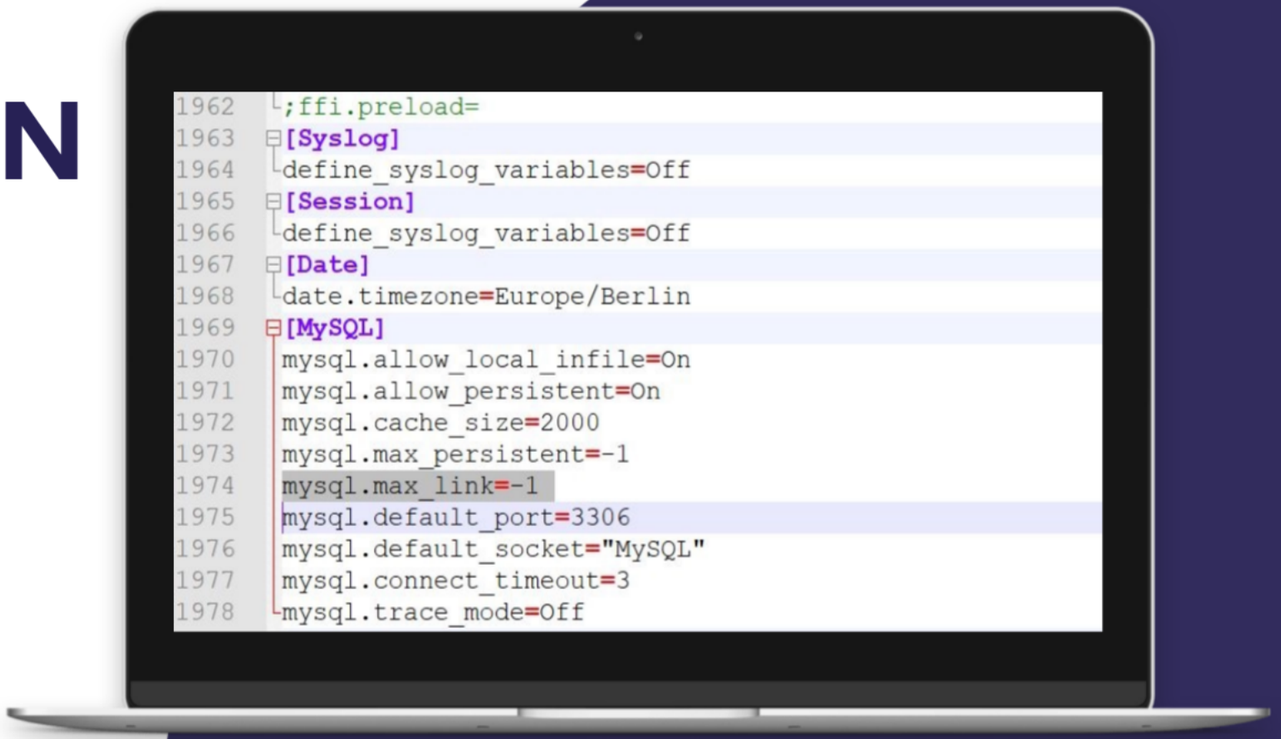
En los temas anteriores hemos aprendido a intercambiar datos entre varias páginas y a guardar información sobre el usuario que está utilizando la aplicación para, de esta forma, personalizar los contenidos que se le muestran. Pero esta información es **temporal**; se eliminará tan pronto como el usuario cierre el navegador o la cookie caduque.

Powered by genially EDUCATION

<01> CONFIGURACIÓN Y ELECCIONES PREVIAS

Lo primero es lo primero

EMPEZAR >



Powered by genially EDUCATION

ELECCIÓN DE LA API

El primer paso, antes de realizar cualquier operación en una base de datos, consiste en elegir la API que vamos a utilizar.
Una **API (Application Programming Interface)** es un conjunto de funciones que permiten acceder y manipular la base de datos.

Powered by genially EDUCATION

MYSQLI PROCEDIMENTAL VS ORIENTADO A OBJETOS

Como ya dijimos, tenemos dos formas de trabajar con la API mysqli:

Powered by genially EDUCATION

PROGRAMACIÓN ORIENTADA A OBJETOS

La **Programación Orientada a Objetos (POO)** es un estilo de programación donde organizamos el código en objetos que representan cosas del mundo real o conceptos. Cada objeto tiene:

- **Propiedades** (datos que describen al objeto).
- **Métodos** o funciones (acciones que el objeto puede realizar).

En POO, usamos clases para definir un tipo de objeto. Luego creamos instancias o "copias" de estas clases, llamadas objetos.

Powered by genially EDUCATION

VAMOS A PONER EN PRÁCTICA LO APRENDIDO

EJERCICIOS >

Powered by genially EDUCATION

ELEGIR QUÉ ERRORES SE VAN A MOSTRAR

Otra de las elecciones que debemos hacer, es el tipo de errores que se le van a mostrar al usuario. Eso se hace mediante la directiva **error_reporting** de **php.ini**.

```
; error_reporting
;   Default Value: E_ALL
;   Development Value: E_ALL
;   Production Value: E_ALL & ~E_DEPRECATED & ~E_STRICT
```

Powered by genially EDUCATION

<02>
PHPMYADMIN

Trabajando con BBDD desde la interfaz

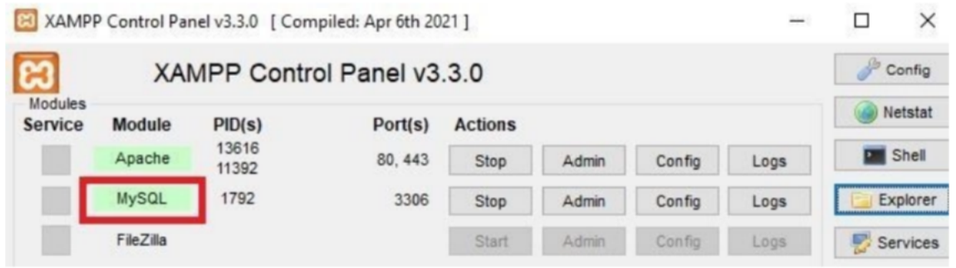
EMPEZAR >

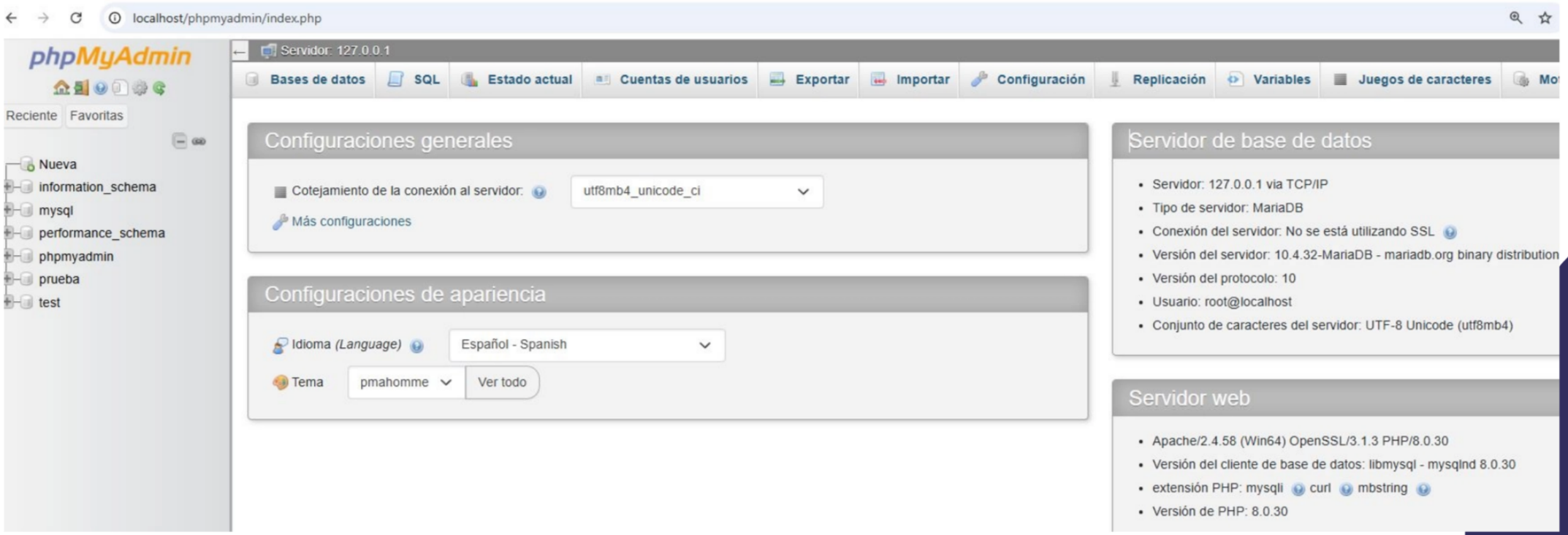


Powered by genially EDUCATION

GESTIONAR LA BD DESDE LA INTERFAZ

Antes de aprender a trabajar con la base de datos por código, necesitamos saber hacerlo mediante la interfaz visual. Para ello, con el **módulo MySQL del XAMPP** arrancado, vamos a la dirección **localhost/phpmyadmin**.

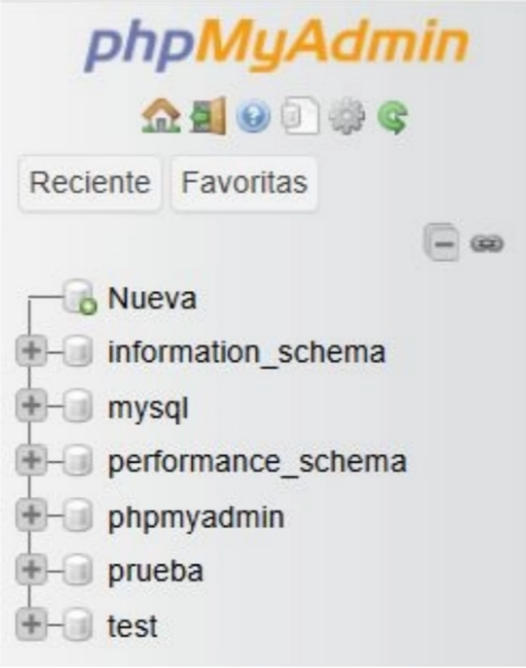




Powered by genially EDUCATION

GESTIONAR LA BD DESDE LA INTERFAZ

En el menú de la izquierda tenemos nuestras BBDD y podemos crear nuevas.



Powered by genially EDUCATION

GESTIONAR LA BD DESDE LA INTERFAZ

El menú superior nos permite realizar una serie de gestiones, tales como ver y modificar la estructura de las tablas creadas, escribir sentencias SQL, importar tablas y datos, exportarlos, crear usuarios y otorgarles permisos, etc.



Como puedes ver, es similar a cualquier otro SGBD.

Powered by genially EDUCATION

VAMOS A PONER EN PRÁCTICA LO APRENDIDO

EJERCICIOS >

Powered by genially EDUCATION

<03> CONEXIÓN A MYSQL POR CÓDIGO

EMPEZAR >



Powered by genially EDUCATION

ESTABLECER CONEXIÓN

Para conectarnos a la base de datos, tenemos que crear un objeto que represente la conexión con ella.
Para hacerlo con mysqli, la sintaxis es la siguiente:

$conexion = new mysqli (servidor, usuario, contraseña, bd, puerto, socket);

Todos los parámetros son opcionales y, de no indicarlos, toman su valor por defecto.

Ejemplo: **$conexion = new mysqli("127.0.0.1", "pepe", "asdf", "prueba");**

Powered by genially EDUCATION

FUNCIÓN QUERY()

Ahora que ya tenemos establecida la conexión con la base de datos, podemos empezar a hacer operaciones sobre ella. Para cualquier operación que queramos realizar, utilizaremos la función **query()** del objeto **conexión**.

Por ejemplo, si queremos crear una tabla 'alumnos':

```
$sentencia = "CREATE TABLE alumnos(id INTEGER PRIMARY KEY, nombre VARCHAR(50), nota DECIMAL(3,1))";
$resultado = $conexion->query($sentencia);
```

Powered by genially EDUCATION

UPDATE, INSERT Y DELETE

Ahora imagina que hacemos un **UPDATE**, **INSERT** o **DELETE**. Normalmente, querremos saber cuántas filas han sido afectadas por esa operación. Esto lo conseguimos mediante la propiedad **affected_rows** del objeto **$conexión**.

Powered by genially EDUCATION

SELECT

Si la instrucción sobre la BD es un **select**, la función **query()** devolverá un conjunto de resultados (result set) con todas las filas seleccionadas.
Este result set debemos guardarlo en una variable para poder recorrerlo fila a fila, y de esta forma, mostrar u operar con los resultados obtenidos.

```php
$sentencia = "SELECT nombre, nota FROM alumnos";
$res = $conexion->query(query: $sentencia);
```

Powered by genially EDUCATION

SELECT

Nota: si la instrucción select fuese a traer demasiados resultados y, con ello, provocar un desbordamiento de memoria, podríamos utilizar un segundo parámetro en la función **query()**: **MYSQLI_USE_RESULT**. De esta manera, no obtendríamos todos los resultados, sino solo los primeros hasta llenar un buffer, y cuando necesitásemos el resto, tendríamos que volver a solicitarlos.

Powered by genially EDUCATION

VAMOS A PONER EN PRÁCTICA LO APRENDIDO

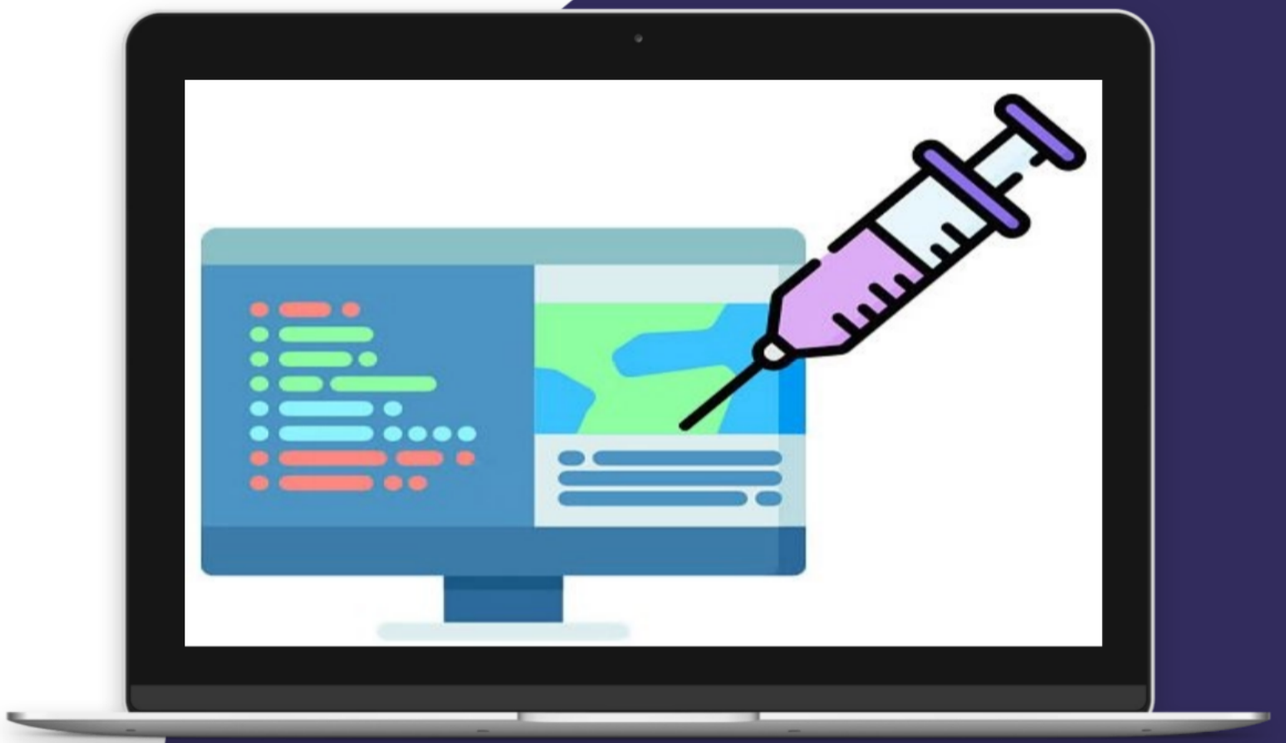
EJERCICIOS >

Powered by genially EDUCATION

<04> INYECCIONES SQL

Qué son y cómo protegernos

EMPEZAR >



Powered by genially EDUCATION

INYECCIONES SQL

Imagina que en la BD tenemos una tabla **'usuarios'** con estos datos:

usuario	password
juan	pass_juan
pepe	pass_pepe

Powered by genially EDUCATION

INYECCIONES SQL

Si la sentencia SQL es **"select * from usuarios where usuario = '$usuario' and password = '$contrasenia'"** y como contraseña metemos **' or '1'='1**, la sentencia resultante quedaría como **"select * from usuarios where usuario = '$usuario' and password = ' ' or '1'='1' "**

Powered by genially EDUCATION

INYECCIONES SQL

EL EJEMPLO SOLUCIONADO QUEDARÍA ASÍ

Otras actuaciones importantes para dotar de más seguridad a nuestra aplicación son cifrar siempre las contraseñas, ocultarlas en los formularios utilizando inputs de tipo **password** y, si solo vamos a consultar datos en la base de datos pero no a modificarlos, conectarse con un usuario que solo tenga **permisos de lectura**.

Powered by genially EDUCATION

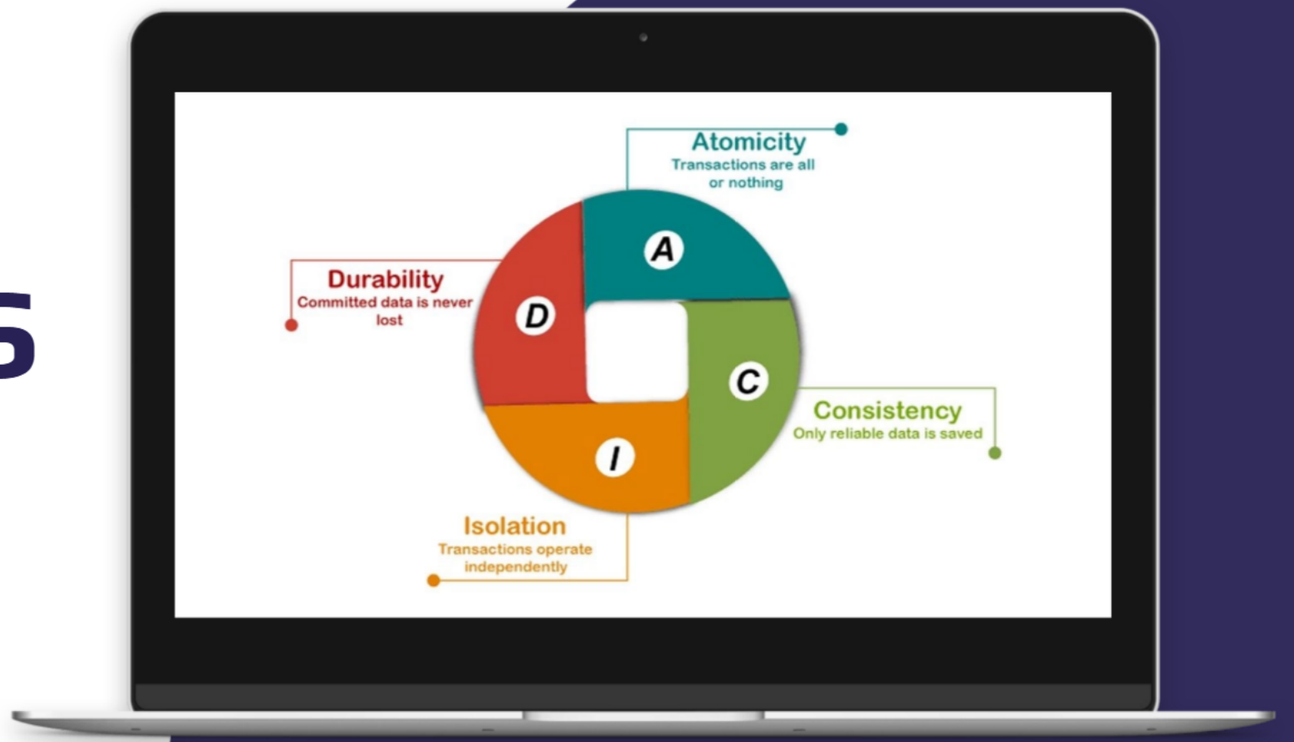
VAMOS A PONER EN PRÁCTICA LO APRENDIDO

EJERCICIOS >

Powered by genially EDUCATION

<05>
USO DE TRANSACCIONES EN PHP

EMPEZAR >



Powered by genially EDUCATION



TRANSACCIONES

En PHP, las **transacciones** son una característica utilizada en combinación con bases de datos que permiten agrupar un conjunto de operaciones de consulta o manipulación de datos de manera que todas se ejecuten exitosamente o ninguna lo haga. Esto asegura la consistencia de los datos incluso en caso de errores.

Powered by genially EDUCATION

TRANSACCIONES

En PHP, las transacciones son **automáticas** por defecto. Esto significa que cada vez que ejecutamos una instrucción **DML** (**UPDATE**, **DELETE** o **INSERT**), la transacción se confirma.

Powered by genially EDUCATION

TRANSACCIONES

Para ello, el objeto conexión dispone del método **autocommit(boolean).** Por tanto, si queremos desactivar las transacciones automáticas para controlarlas nosotros, debemos usar **$conexion->autocommit(false);** al principio.

Powered by genially EDUCATION

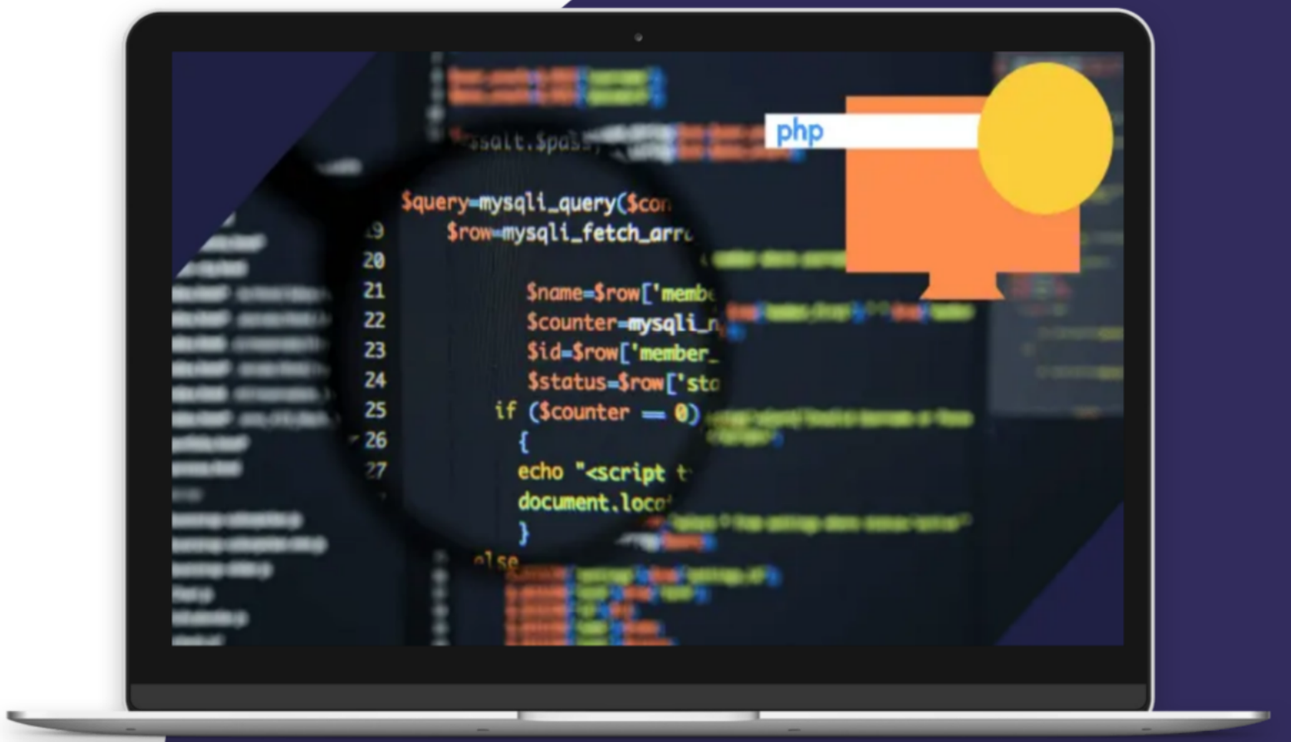
VAMOS A PONER EN PRÁCTICA LO APRENDIDO

EJERCICIOS >

Powered by genially EDUCATION

<06>
EJERCICIOS

EMPEZAR >



Powered by genially EDUCATION

EJERCICIOS

ENUNCIADOS

EJERCICIOS RESUELTOS

Powered by genially EDUCATION

Tema 6 finalizado



Powered by genially EDUCATION