Base de datos en equipo

SQL,XAMPP y Github

GF Academy

Versión 1.0 - Fernando Flores Junio 2026

1. Introducción

Estas trabajando con un compañero, clonaste un repositorio de Github o alguien te pasó un proyecto PHP, pero al abrirlo te das cuenta de algo: el código está ahí, pero la base de datos no.

Esto pasa mucho en proyectos escolares y en equipos pequeños. GitHub guarda los archivos del proyecto, como .php, .css, .js o imágenes, pero no guarda automáticamente la base de datos local que tienes en MySQL. Por ejemplo, si tú creas una base de datos llamada tienda_db y dentro agregas una tabla llamada usuarios, esa base de datos existe en tu computadora, no en la computadora de tu compañero.

Por eso, cuando otra persona clona el proyecto, puede tener exactamente el mismo código, pero no tendrá las mismas tablas, columnas o datos iniciales. El resultado típico es que el sistema marca errores como "tabla no encontrada", "base de datos desconocida" o simplemente no muestra información.

Para evitar ese problema, una forma práctica de trabajar es guardar los comandos SQL del proyecto dentro de una carpeta llamada database/. Así cualquier integrante del equipo puede ejecutar esos comandos desde phpMyAdmin y crear la misma base de datos en su computadora.

En esta guía no usaremos terminal. Todo se hará desde XAMPP, MySQL y phpMyAdmin.

2. La idea principal

Cuando trabajamos con base de datos en equipo, no basta con pasar el código del proyecto. También necesitamos pasar la estructura de la base de datos.

La estructura de una base de datos incluye cosas como:

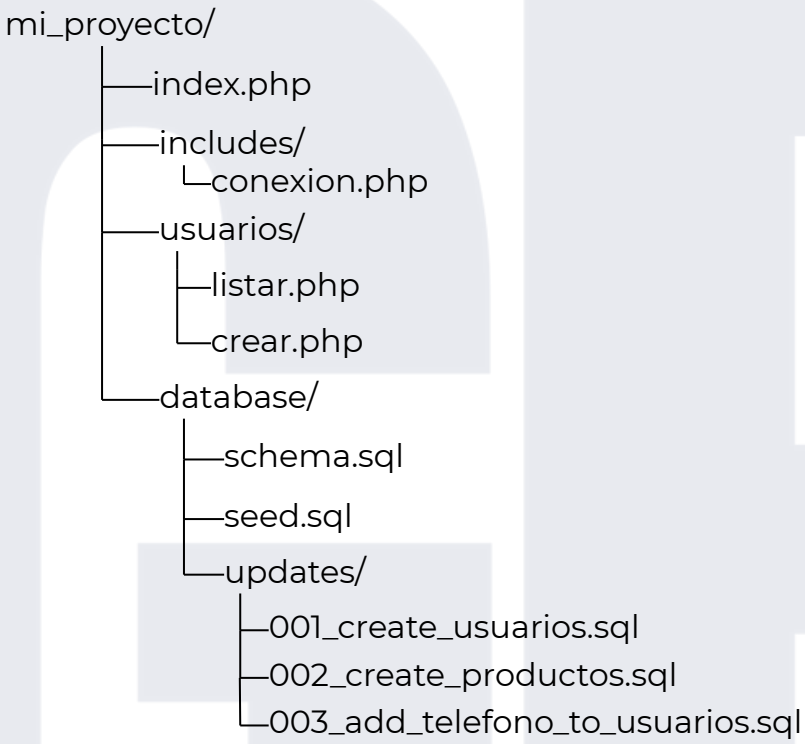
- El nombre de la db
- Tablas
- Columnas
- Tipo de datos
- Llaves primarias (Primary Key)
- Relaciones entre tablas

Para compartir eso de forma ordenada, podemos usar archivos **.sql**

Un archivo **.sql** es simplemente un archivo de texto que contiene comandos SQL. Esos comandos pueden crear bases de datos, crear tablas, insertar datos o modificar estructuras existentes.

3. Estructura recomendada

Una estructura sencilla para proyectos con XAMPP puede ser esta:



La carpeta importante para este tema es **database/**

4. ¿Qué es schema.sql?

El archivo schema.sql contiene la estructura principal de la base de datos.

Normalmente aquí se colocan los comandos para:

- Crear la base de datos
- Seleccionar la base de datos
- Crear las tablas principales
- Definir columnas, tipos de datos y llaves primarias

Ejemplo:

```sql
CREATE DATABASE IF NOT EXISTS gf_tienda
CHARACTER SET utf8mb4
COLLATE utf8mb4_unicode_ci;

USE gf_tienda;

CREATE TABLE IF NOT EXISTS usuarios (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    correo VARCHAR(120) NOT NULL,
    fecha_registro TIMESTAMP DEFAULT
    CURRENT_TIMESTAMP
);
```

Este archivo es como el esqueleto de la base de datos. Sin él, el proyecto puede tener código PHP, pero no tendrá dónde guardar ni consultar información.

Se puede notar que la sintaxis de SQL es similar al lenguaje humano, especialmente porque muchos comandos se leen casi como una instrucción directa. En este documento no se explicará SQL a profundidad, pero sí veremos de forma superficial algunos términos que aparecen en los comandos utilizados.

- CHARACTER SET: define qué tipo de caracteres puede guardar la base de datos.
- COLLATE: define cómo se comparan y ordenan los textos dentro de la base de datos.

utf8mb4: permite guardar correctamente letras con acentos, la ñ, símbolos especiales e incluso emojis.
utf8mb4_unicode_ci: indica que se usarán reglas compatibles con muchos idiomas. La parte ci significa case insensitive, es decir, que no distingue entre mayúsculas y minúsculas al comparar textos.
Actualmente no es necesario declarar el CHARACTER SET ni COLLATE, ya que vienen por defecto. Pero es una buena practica si trabajas con servidores legacy

5. ¿Qué es seed.sql?

El archivo **seed.sql** contiene datos inciales o datos de prueba.

Sirve para que, después de crear las tablas, el proyecto ya tenga información básica para probar.

Ejemplo:

```sql
INSERT INTO usuarios (nombre, correo)
VALUES ('Fernando Flores',
'fernando@example.com'), ('Ana López',
'ana@example.com'), ('Carlos Pérez',
'carlos@example.com');
```

Este archivo no siempre es obligatorio, pero ayuda mucho cuando estás enseñando, probando o trabajando en equipo.

La idea es que tu compañero no tenga que inventar datos desde cero. Ejecuta **seed.sql** y listo, ya tiene usuarios de prueba

6. ¿Qué es la carpeta updates?

La carpeta **updates/** sirve para guardar cambios pequeños que se hacen despues de la primera versión de la base de datos.

Por ejemplo, al inicio tal vez creaste una tabla **usuarios** con **id**, **nombre** y **correo**, pero después el proyecto necesita agregar un teléfono.

En vez de decirle a tu compañero que busque la tabla usuarios y agregue una columna nueva, puedes crear un archivo como este:

003_add_telefono_to_usuarios.sql

Y dentro colocar:

```sql
USE gf_tienda;

ALTER TABLE usuarios ADD telefono
VARCHAR(20);
```

Así el cambio queda guardado, ordenado y caulquier persona puede aplicarlo. La **numeración** ayuda a saber **el orden** en que deben **ejecutarse** los cambios.

7. Cómo ejecutar comandos SQL

Primero asegúrate de tener encendidos Apache y MySQL desde XAMPP.

Luego entra a: http://localhost/phpmyadmin/

Dentro de phpMyAdmin puedes ejecutar comandos SQL de dos formas:

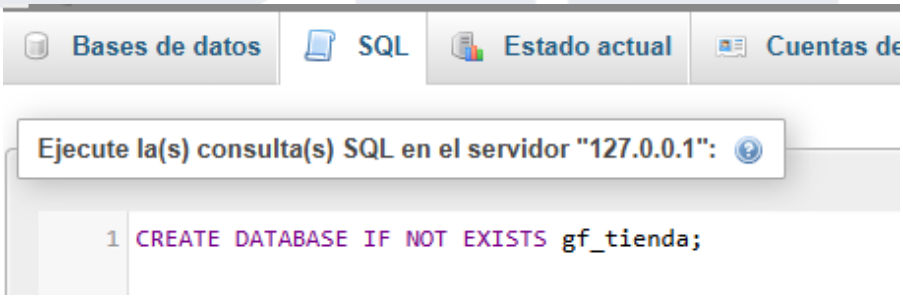
- Desde la pestaña SQL
- Desde la opción importar, subiendo un archivo .sql

Para este documento, usaremos la pestaña SQL que es mas rápida y sencilla

8. Crear una base de datos usando SQL

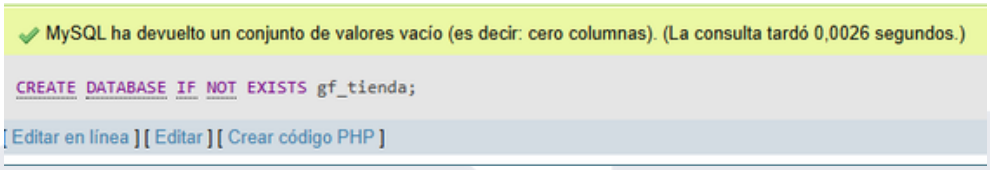
en phpMyAdmin, entra a la pestaña SQL y escribe:

```sql
CREATE DATABASE IF NOT EXISTS gf_tienda;
```



Luego presiona el botón **Continuar** (en la parte inferior derecha)

Si todo sale bien, phpMyAdmin creará la base de datos.



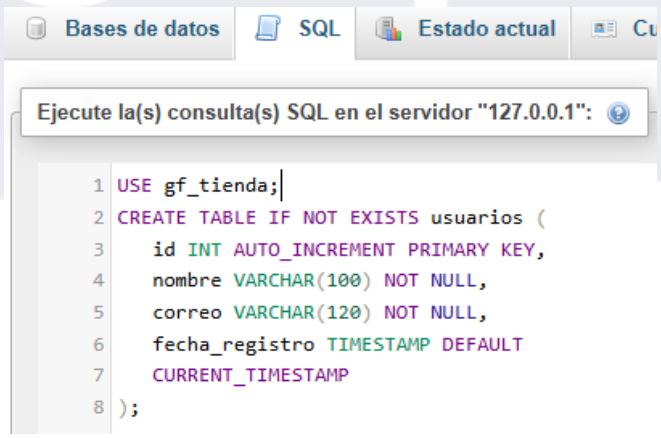
Después puedes seleccionarla desde el panel izquierdo.

9. Crear una tabla usando SQL

Una vez seleccionada la base de datos, entra otra vez a la pestaña SQL y ejecuta:

```sql
CREATE TABLE IF NOT EXISTS usuarios (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    correo VARCHAR(120) NOT NULL,
    fecha_registro TIMESTAMP DEFAULT
    CURRENT_TIMESTAMP
);
```

Si no estas seguro de haber seleccionado la base de datos que creamos, antes de ejecutar el comando CREATE TABLE, agrega USE gf_tienda;



Este comando crea una tabla llamada **usuarios**.

La columna **id** será el identificador único de cada usuario.

La columna **nombre** guardará el nombre.

La columna **correo** guardará el correo.

La columna **fecha_registro** guardará automáticamente la fecha en la que se insertó el registro

10. Insertar datos de prueba

Después de crear la tabla, puedes insertar datos usando:

```sql
INSERT INTO usuarios (nombre, correo)
VALUES ('Fernando Flores',
'fernando@example.com'), ('Ana López',
'ana@example.com');
```

Si entras a la tabla **usuarios** y das clic en Examinar, deberías ver los registros insertados

11. Flujo recomendado para trabajar en equipo

Cuando trabajes con un compañero, el flujo puede dividirse en dos situaciones: la primera instalación del proyecto y las actualizaciones posteriores.

Primera vez que se configura el proyecto

Este flujo se usa cuando una persona clona el repositorio por primera vez y todavía no tiene la base de datos creada en su computadora

1. Clonar el proyecto desde github
2. Abrir XAMPP
3. Iniciar Apache y MySQL
4. Entrar a phpMyAdmin
5. Ejecutar el archivo **schema.sql** para crear la base de datos y sus tablas principales
6. Ejecutar **seed.sql** si necesitan datos de prueba
7. Revisar que el archivo de **conexion.php** tenga el mismo nombre de base de datos que se creó

El archivo **schema.sql** se ejecuta solo una vez, cuando todavía no existe la base de datos en la computadora local

Cuando ya tienes el proyecto y haces pull

Este flujo se usa cuando ya tenías el proyecto funcionando, pero otro compañero hizo cambios y los subió a GitHub

1. Hacer pull al repositorio
2. Revisar si hay archivos nuevos dentro de la carpeta **database/updates/**
3. Ejecutar solo los archivos nuevos de **updates/** que todavía no se hayan aplicado

4. Revisar si el cambio afecta el archivo **conexion.php** o alguna parte del codigo

12. Importante

Esta forma de trabajar es ideal para proyectos PHP normales, XAMPP, phpMyAdmin y prácticas escolares.

En frameworks como Laravel existe otra forma llamada migraciones, usando comando como:

```
php artisan migrate
```

Pero eso solo aplica cuando el proyecto está hecho con Laravel.

Para proyectos PHP normales, guardar y ejecutar archivos **.sql** sigue siendo una forma válida, clara y muy útil para trabajar en equipo

13. Cierre

Trabajar con base de datos no significa que cada integrante tenga que crear tablas manualmente desde cero. Lo más ordenado es guardar los comandos SQL del proyecto y ejecutarlos cuando sea necesario.

Con este flujo, cualquier compañero puede levantar el proyecto en su computadora y tener la misma base de datos local para probar, modificar y seguir desarrollando.