

Guía de Estudio: Migraciones en Laravel

1. El Problema (¿Por qué no usar simplemente phpMyAdmin?)

Imagina que estás haciendo un proyecto en grupo. Creas tu base de datos y le pasas un archivo .sql a tu compañero para que lo importe. Al día siguiente, te das cuenta de que olvidaste la columna telefono, así que la añades en tu computadora. Ahora tienes que enviarle *otro* archivo .sql a tu compañero o pedirle que la añada manualmente. Cuando el proyecto crece, mantener las bases de datos sincronizadas entre varios programadores es un caos.

La Solución: Las migraciones.

En lugar de crear tablas haciendo clics en un gestor visual, escribes **código PHP** que describe cómo debe ser la tabla. Así, el código de la base de datos viaja junto con el resto de tu proyecto.

2. El Concepto: Viajes en el tiempo para tu base de datos

Una migración es como un "punto de guardado" (save state) de un videojuego para tu base de datos. Cada archivo de migración tiene exactamente dos tareas:

- **El método up() (Avanzar):** Define lo que quieres construir. Por ejemplo, "Crea una tabla llamada artículos con un título y un texto".
- **El método down() (Retroceder):** Define cómo deshacer exactamente lo que hizo el método up. Por ejemplo, "Elimina la tabla artículos".

Si te equivocas, Laravel usa el método down() para regresar la base de datos a como estaba antes de tu error.

3. Tu Caja de Herramientas (Comandos Esenciales)

Estos son los tres comandos que usarás todos los días en tu terminal:

Comando	¿Qué hace?	Analogía
php artisan make:migration create_ejemplo_table	Crea un archivo en blanco en tu proyecto para que escribas la estructura de tu nueva tabla.	Comprar un plano en blanco para dibujar tu casa.
php artisan migrate	Lee todos los planos que no se han construido y los ejecuta en MySQL. Transforma el código PHP en tablas reales.	Entregarle los planos a los obreros para que construyan.

Comando	¿Qué hace?	Analogía
php artisan migrate:rollback	Deshace la última migración que ejecutaste (llama al método down). Útil si corriste el comando y recordaste que faltaba un campo.	Demoler lo último que construiste porque quedó mal.

4. Reglas de Oro

1. **Nunca edites una migración vieja:** Si ya corriste php artisan migrate y el código ya está en la computadora, no modifiques ese archivo. Si necesitas una nueva columna, crea una **nueva** migración que solo añada esa columna.
2. **Nombres descriptivos:** El nombre del archivo debe decir exactamente qué hace (ej. add_edad_to_usuarios_table).
3. **No le temas a romper cosas en local:** Mientras estés aprendiendo en tu computadora, si todo se vuelve un desastre, el comando php artisan migrate:fresh borra todas las tablas y las vuelve a crear desde cero, dejándote una base de datos limpia.