📌 Tipos de Relaciones en Laravel

Laravel proporciona varias relaciones, entre las más comunes están:
✅ **Uno a Uno (`hasOne`)**
✅ **Uno a Muchos (`hasMany`)**
✅ **Muchos a Uno (`belongsTo`)**
✅ **Muchos a Muchos (`belongsToMany`)**
✅ **Relación Polimórfica**

🔹 1. Relación Uno a Uno (`hasOne`)

Ejemplo: **Un usuario tiene un perfil** (una tabla `users` y otra `profiles`).

📌 Paso 1: Crear las Migraciones

bash
CopiarEditar

```
php artisan make:migration create_profiles_table
```

En la migración `create_profiles_table.php`, define la relación:

php
CopiarEditar

```
Schema::create('profiles', function (Blueprint $table) {
    $table->id();

$table->foreignId('user_id')->constrained()->onDelete('cascade');
    $table->string('bio')->nullable();
    $table->timestamps();
});
```

Ejecuta la migración:

bash
CopiarEditar

```
php artisan migrate
```

📌 Paso 2: Definir la Relación en el Modelo

En `app/Models/User.php`:

php
CopiarEditar
```php
class User extends Model
{
    public function profile()
    {
        return $this->hasOne(Profile::class);
    }
}
```

En `app/Models/Profile.php`:

php
CopiarEditar
```php
class Profile extends Model
{
    public function user()
    {
        return $this->belongsTo(User::class);
    }
}
```

📌 Paso 3: Usar la Relación

php
CopiarEditar
```php
$user = User::find(1);
$profile = $user->profile; // Obtener el perfil del usuario
```

🔹 2. Relación Uno a Muchos (`hasMany`)

Ejemplo: **Un usuario puede tener muchos posts**.

📌 Paso 1: Crear la Migración

bash
CopiarEditar
```bash
php artisan make:migration create_posts_table
```

En la migración `create_posts_table.php`:

php
CopiarEditar
```
Schema::create('posts', function (Blueprint $table) {
    $table->id();

$table->foreignId('user_id')->constrained()->onDelete('cascade');
    $table->string('title');
    $table->text('content');
    $table->timestamps();
});
```

Ejecuta la migración:

bash
CopiarEditar
```
php artisan migrate
```

📌 Paso 2: Definir la Relación en los Modelos

En `app/Models/User.php`:

php
CopiarEditar
```
class User extends Model
{
    public function posts()
    {
        return $this->hasMany(Post::class);
    }
}
```

En `app/Models/Post.php`:

php
CopiarEditar
```
class Post extends Model
{
    public function user()
    {
        return $this->belongsTo(User::class);
```

```
    }
}
```

📌 Paso 3: Usar la Relación

php
CopiarEditar

```
$user = User::find(1);
$posts = $user->posts; // Obtener todos los posts del usuario

$post = Post::find(1);
$user = $post->user; // Obtener el usuario de un post
```

🔹 3. Relación Muchos a Muchos (`belongsToMany`)

Ejemplo: **Un usuario puede tener muchos roles y un rol puede pertenecer a muchos usuarios**.

📌 Paso 1: Crear la Tabla Intermedia

bash
CopiarEditar

```
php artisan make:migration create_role_user_table
```

En la migración `create_role_user_table.php`:

php
CopiarEditar

```
Schema::create('role_user', function (Blueprint $table) {
    $table->id();

$table->foreignId('user_id')->constrained()->onDelete('cascade');

$table->foreignId('role_id')->constrained()->onDelete('cascade');
    $table->timestamps();
});
```

Ejecuta la migración:

bash
CopiarEditar

```
php artisan migrate
```

📌 Paso 2: Definir la Relación en los Modelos

En `app/Models/User.php`:

php
CopiarEditar

```
class User extends Model
{
    public function roles()
    {
        return $this->belongsToMany(Role::class);
    }
}
```

En `app/Models/Role.php`:

php
CopiarEditar

```
class Role extends Model
{
    public function users()
    {
        return $this->belongsToMany(User::class);
    }
}
```

📌 Paso 3: Usar la Relación

php
CopiarEditar

```
$user = User::find(1);
$roles = $user->roles; // Obtener roles del usuario

$role = Role::find(1);
$users = $role->users; // Obtener usuarios con ese rol

// Asignar un rol a un usuario
$user->roles()->attach(2); // Asigna el rol con ID 2
$user->roles()->detach(2); // Elimina el rol con ID 2
```

```
$user->roles()->sync([1, 3]); // Reemplaza los roles actuales con 1
y 3
```

Ejemplo: Migraciones sin relaciones

php
CopiarEditar

```
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateUsersTable extends Migration
{
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->timestamps();
        });
    }

    public function down()
    {
        Schema::dropIfExists('users');
    }
}
```

php
CopiarEditar

```
class CreateProfilesTable extends Migration
{
    public function up()
    {
        Schema::create('profiles', function (Blueprint $table) {
            $table->id();
            $table->unsignedBigInteger('user_id'); // No definimos
la clave foránea aquí
            $table->string('bio');
            $table->timestamps();
```

```
        });
    }

    public function down()
    {
        Schema::dropIfExists('profiles');
    }
}
```

🔹 Aquí `profiles.user_id` **no tiene clave foránea**, solo es una columna de tipo `unsignedBigInteger`.

📌 ¿Cómo se definen las relaciones en los modelos?

En lugar de definir claves foráneas en las migraciones, **Laravel usa los modelos para establecer la relación**.

php
CopiarEditar
```
class User extends Model
{
    public function profile()
    {
        return $this->hasOne(Profile::class, 'user_id');
    }
}
```

php
CopiarEditar
```
class Profile extends Model
{
    public function user()
    {
        return $this->belongsTo(User::class, 'user_id');
    }
}
```

🔹 Laravel **manejará la relación automáticamente** cuando hagas consultas como:

php

CopiarEditar

```
$user = User::find(1);
$profile = $user->profile; // Obtiene el perfil del usuario
```