



LARAVEL

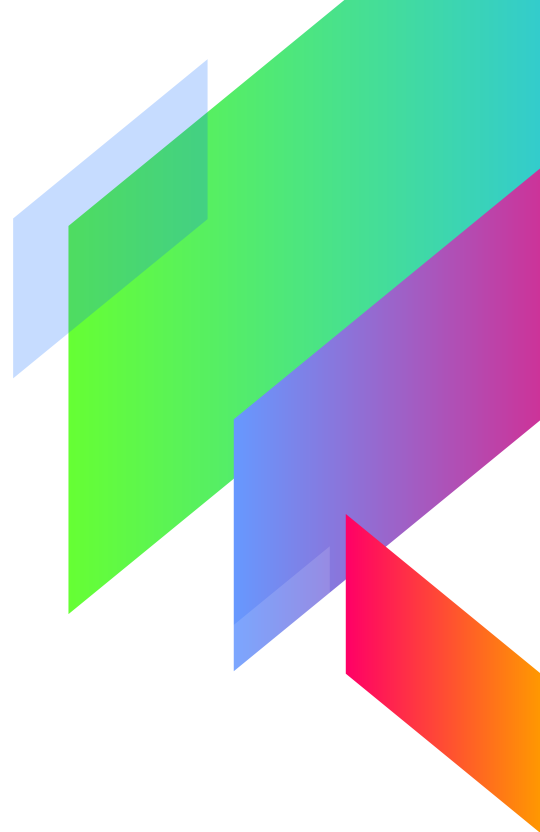
CLASE 05





RELACIONES

Manejo de relaciones en Laravel



RELACIONES

DIRECCIONALIDAD

A -- conoce a --> B

No es lo mismo que...

B -- conoce a --> A

CARDINALIDAD

A -- tiene un --> B

No es lo mismo que...

A -- tiene muchos --> B

ELOQUENT: RELACIONES

N:1 - Belongs To

- ✗ Episode **belongsTo** Season
- ✗ Profile **belongsTo** User

1:0...1 - Has One

- ✗ User **hasOne** Profile
- ✗ Person **hasOne** DNI

1:0...N - Has Many

- ✗ Season **hasMany** Episode
- ✗ Genre **hasMany** Movie

N:M - Belongs to Many

- ✗ Actor **belongsToMany** Movie
- ✗ Client **belongsToMany** Product

```
class User extends Model
{
    // hasOne / belongsTo
    public function profile()
    {
        return $this->hasOne(Profile::class, 'foreign_key', 'other_key');
    }
}

class Profile extends Model
{
    public function user()
    {
        return $this->belongsTo(User::class, 'foreign_key', 'other_key');
    }
}

// A partir de un Objeto User
$user = User::find(1);

// Obtenemos su objeto Profile relacionado
$profile = $user->profile;

// Y viceversa...
$profile = Profile::find(42);
$user = $profile->user;
```

```
class User extends Model
{
    // hasMany / belongsTo
    public function pets()
    {
        return $this->hasMany(Pet::class, 'foreign_key', 'local_key');
    }
}

class Pet extends Model
{
    public function user()
    {
        return $this->belongsTo(User::class, 'foreign_key', 'other_key');
    }
}

// A partir de un Objeto User obtenemos su colección de objetos Pet relacionados
$user = User::find(1);
$pets = $user->pets;

// Y viceversa
$pet = Pet::find(9);
$user = $pet->user;

// Podemos hacer una query para obtener solo algunos
$dogs = $user->pets()->where('type', 'dog')->get();
```

```
class Actor extends Model
{
    // belongsToMany
    public function movies()
    {
        return $this->belongsToMany(Movie::class, 'table', 'foreign_key', 'other_key');
    }
}
```

```
class Movie extends Model
{
    public function actors()
    {
        return $this->belongsToMany(Actor::class, 'table', 'foreign_key', 'other_key');
    }
}
```

// A partir de un Objeto Actor obtenemos su colección de objetos Movies relacionados

```
$actor = Actor::find(1);
```

```
$movies = $actor->movies;
```

// Y viceversa

```
$movie = Movie::find(23);
```

```
$actors = $movie->actors;
```

// También podemos hacer queries

```
$bestActors = $movie->actors()->where('rating', '>', 8)->orderBy('last_name')->get();
```

**ES MOMENTO DE
PRACTICAR !**



The image features abstract geometric shapes in the corners. On the left, there are overlapping shapes in shades of green, blue, orange, and purple. On the right, there are overlapping shapes in shades of green, blue, purple, and orange. The central text is in a bold, black, sans-serif font.

**¡ HASTA LA
PROXIMA !**