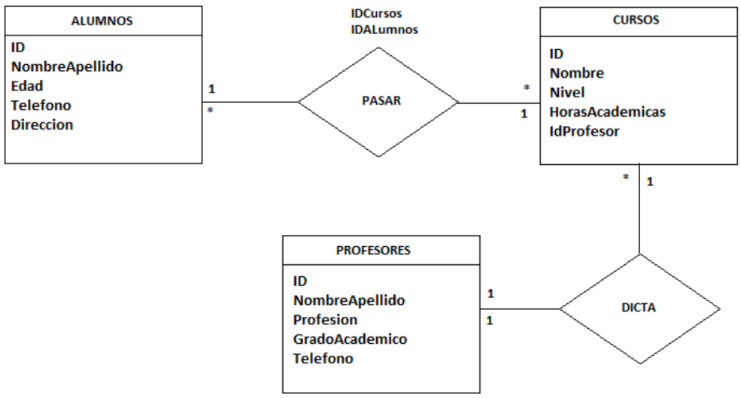
PROIEKTU BAT SORTU - AULA ADIBIDEA

Behin ikusita zelan sortzen den proiektu bat eta zelan egin dezakegun lan horregaz, jarraituko dugu aurrera....

Gure taula "**Alumnos**", "**Profesor**" eta "**Curso**" taulekin erlazionatuko dugu. Hurrengo entidad-relacion jarraituz...

Adibide hau jarraitzeko taulak gazteleraz erabiliko ditugu eta baita euren propietateak ere.



1-N erlazioa

Profesors Migrazioa

Hasiko dugu beraz Profesores migrazioa sortzen, modeloa ere aldi berean egingo dugu, ikusten dugun moduan, laravel-en konbentzioa jarraitzerakoan, taula "profesors" izango da.

```
php artisan make:Model Profesor -m
```

Eta migrazioa, zehaztuko diogu zelakoa gure dugu gure taula izatea:

```
            $table->string('nombreApellido', 75);
            $table->string('profesion', 35);
            $table->string('gradoAcademico', 35)->nullable();
            $table->string('telefono', 35)->nullable();
```

Profesor Moduloa

Behin migrazioa eginda, gure moduloa sortuko dugu:

```
    protected $table = "profesors";
    protected $primaryKey = "id";
    protected $fillable = ['nombreApellido', 'profesion', 'gradoAcademico', 'telefono'];
    protected $hidden = [ 'id'];
```

Oraingoan migrazioak fresh batekin exekutatuko ditugu.

```
php artisan migrate:fresh
```

Curso Migrazioa

Lehenengo esta behin Modeloa sortuko dugu, kasu honetan Curso izenarekin:

```
php artisan make:Model Curso -m
```

Eta migrazioa, zehaztuko diogu zelakoa gure dugu gure taula izatea, kasu honetan eremu berezi bat dugu, gure profesor_id eremua, kasu honetan foreign key bat izango dena, profesors taularen id; hori zehazteko hurrengo adibidean dugun moduan jarriko dugu:

```
$table->string('nombre', 75);
$table->string('nivel', 35);
```

```
$table->string('horasAcademicas', 35)->nullable();
$table->unsignedBigInteger('profesor_id');
$table->foreign('profesor_id')->references('id')->on('profesors');
```

Curso Modeloa

Modeloa aldatzera goaz orain:

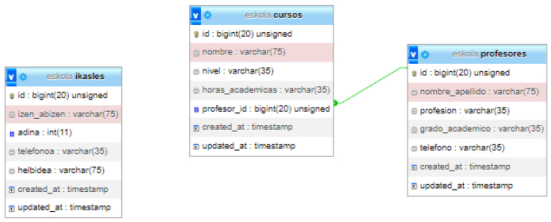
```
    protected $table = "cursos";
    protected $primaryKey = "id";
    protected $fillable = ['nombre', 'nivel', 'horasAcademicas', 'profesor_id'];
    protected $hidden = [ 'id'];
```

Exekutatuko ditugu gure migrazioak berriro:

```
php artisan migrate:fresh
```

Ikusten dugunez, gure taula biak sortu ditu eta baita euren arteko erlazioa ere.

Orain faltako litzaiguke alumnos eta cursos-en artekoa sortzen; kasu honetan ezberdin egin beharko dugu, N-M erlazioa dalako.



Eloquent erlazioak

Erlazioak modeloetan adierazi behar dira ere. Horretarako hainbat gauza izan behar ditugu kontutan

- **Irakasle batek hainbat kurtso eman ditzake eta kurtso bat irakasle batek bakarrik eman dezake, beraz badugu 1-N erlazioa bat.**

Hori definitzeko, Profesor modeloan, **hasMany** metodoa erabiliko dugu, metodo honek definitzen bait du 1-N erlazioak.

```
public function cursos(){
  return $this->hasMany(Curso::class);
}
```

```
class Profesor extends Model
{
    use HasFactory;

    protected $table = "profesors";
    protected $primaryKey = "id";
    protected $fillable = ['nombreApellido', 'profesion', 'gradoAcademico', 'telefono'];
    protected $hidden = [ 'id'];

    public function cursos(){
        return $this->hasMany(Curso::class);
    }
}
```

```
class Curso extends Model
{
    use HasFactory;

    protected $table = "cursos";
    protected $primaryKey = "id";
    protected $fillable = ['nombre', 'nivel', 'horasAcademicas', 'profesor_id'];
    protected $hidden = [ 'id'];

    public function profesor(){
        return $this->belongsTo(Profesor::class);
    }
}
```

Eta kontrakoa definitzeko, hau da, hainbatetik baterako erlazioa, **belongsTo** metodoa erabiliko dugu, hori Curso modeloan definituko dugu:

```
public function profesor(){
  return $this->belongsTo(Profesor::class);
}
```

Factory Profesor

Sortuko dugu gure Factory-a datu gehiago izateko:

```
php artisan make:factory ProfesorFactory
```

Factory horren definition metodoan, gure taularen eremu guztiak definituko ditugu:

```
    public function definition(){
        return [
            'nombreApellido' => $this->faker->firstName() . ' ' . $this->faker->lastName(),
            'profesion' => $this->faker->randomElement(['Ing. Sistema', 'Ing Informatico', 'Adm Empresas',
'Ing Comercial']),
            'gradoAcademico' => $this->faker->word(),
            'telefono' => $this->faker->phoneNumber()
        ];
    }
```

Orain, gure ProfesorFactory erabili ahal izateko, gure DataBaseSeeder aldatuko dugu:

```
Profesor::factory(10)->create();
```

Factory Alumno

Sortuko dugu gure Factory-a datu gehiago izateko:

```
php artisan make:factory AlumnoFactory
```

Kodea hurrengo hau izanik:

```
       return [
            // Ikasle taulan datuak sartzeko erregistroa
```

```
            'nombre'=> $this->faker->firstName() . ' ' . $this->faker->lastName(),
            'edad'=>$this->faker->randomElement([18,19,20,21]),
            'telefono'=>$this->faker->phoneNumber(),
            'direccion'=>$this->faker->address()
        ];
```

DataBaseSeeder aldatuko dugu:

```
Alumno::factory(50)->create();
```

Factory Curso

```
php artisan make:factory CursoFactory
```

Aurrekoetan bezala, gure taularen datuak sartuko ditugu:

```
return [
'nombre' => $this->faker->word(),
'nivel' => $this->faker->randomElement(['Basico', 'Intermedio', 'Avanzado']),
'horasAcademicas' => $this->faker->randomElement(['10 Horas', '40 Horas', '80 Horas']),
'profesor_id' => Profesor::all()->random()->id
];
```

Ikusten dugunez azkeneko sententzian, profesor_id bat sartzeko, hartuko dugu gure Profesor modelotik datu guztiak eta euren artean bat hartuko dugu modu aleatorio batean.

Horretarako, modeloa gehitu behar diogu: `use App\Models\Profesor;`

Hori guztia egin ostean, Seederrak exekutatuko ditugu:

```
php artisan db:seed
```