# Laravel (continuação, parte 3)

## Your first Eloquent Relationships

A project has tasks, OR a task belongs to a project.

Make model with –m (migration)

```
php artisan make:model Task -m
```

Não confundir com make:controller CtrlName –m ModelName, cada comando tem letras especificas

**The task migration:**

```
public function up()
{
    Schema::create('tasks', function (Blueprint $table) {
        $table->increments('id');
        $table->unsignedInteger('project_id'); // unsignedInteger for only positive numbers
        $table->string('description');
        $table->boolean('completed')->default(false);
        $table->timestamps();
    });
```

```
    }
php artisan migrate
```

## Make the relationship

At the projects **model**

```
class Project extends Model
{
    protected $guarded = [];

    public function tasks()
    {
        return $this->hasMany(Task::class);
        /* examples:
        return $this->belongsTo;
        return $this->hasManyThrough;
        return $this->morphMany;
        */
    }
}

// objectivo: $project->tasks; to get project tasks at controller
```

**To test: Create tasks in database manually**

At view projects/show.blade.php:

```
    @if ($project->tasks->count())
    <div>
        @foreach ($project->tasks as $task)
            <li>{{ $task->description }}</li>
        @endforeach
    </div>
    @endif
```

Cofinanciado por:

MADEIRA 14-20 REGIÃO AUTÓNOMA DA MADEIRA PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

## Inverse relationship

```
class Task extends Model
{

    public function project()
    {
        return $this->belongsTo(Project::class);
    }
}
// objectivo: $task->project; to get the task correspondent project
```

**Documentação:** https://laravel.com/docs/6.x/eloquent-relationships

# Form Action Considerations / Update task

## View

projects/show.blade.php:

```
        @foreach ($project->tasks as $task)
        <div>
            <form method="POST" action="/tasks/{{ $task->id }}">
                @method('PATCH')
                @csrf

                <label class="checkbox" for="completed">
                    <input type="checkbox" name="completed" onChange="this.form.submit()">
                    {{ $task->description }}
                </label>
            </form>
        </div>
    @endforeach
```

**Routes possíveis:**

Cofinanciado por:

MADEIRA 14-20 REGIÃO AUTÓNOMA DA MADEIRA PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

PATCH /projects/id/tasks/id
PATCH / tasks/id (mais simples)

## Route

```
Route::patch('/tasks/{task}', 'ProjectTasksController@update'); // or just TasksController
```

Não usamos resource devido a ter relação, os urls podem não ser standard e só vamos usar 2 urls.

## Controller

```
php artisan make:controller ProjectTasksController
```

Update sem verificação do valor **on**, porque na migration foi definido um valor default (false) e quando uma checkbox está vazia não é enviada no request.
**NOTA**: Não esquecer de definir o **fillable ou guarded** no model, devido a usar o método update:

```
use App\Task;

class ProjectTasksController extends Controller
{
    public function update(Task $task)
    {
        $task->update([
            'completed' => request()->has('completed')
        ]);

        return back();
    }
}
```

## Get task status in View

Colocar a checkbox checked e adicionar uma classe nas tasks completas:

Cofinanciado por:

```
    <style>
        .is-complete {
            text-decoration: line-through;
        }
    </style>


<label class="checkbox {{ $task->completed ? 'is-complete' : '' }}" for="completed">
            <input type="checkbox" name="completed" onChange="this.form.submit()" {{ $task->completed ? 'checked' : '' }}>
```

## Create New Project Tasks

### View (projects/show.blade.php)

```
        <form method="POST" action="/projects/{{ $project->id }}/tasks" class="box">
        @csrf
        <div class="field">
            <label  class="label" for="description">New Task</label>

            <div class="control">
                <input type="text" class="input" name="description" placeholder="New Task">
            </div>
        </div>

        <div class="field">
            <div class="control">
                <button type="submit" class="button is-link">Add Task</button>
            </div>
        </div>
    </form>
```

### Route

```
Route::post('/projects/{project}/tasks', 'ProjectTasksController@store'); // or just tasks/{task} BUT with a hidden input with project id
```

Cofinanciado por:

MADEIRA 14-20 REGIÃO AUTÓNOMA DA MADEIRA PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

Neste caso, ao contrário do update, é importante saber o project, porque a task ainda não existe, logo não tem project.

## Controller

```
class ProjectTasksController extends Controller
{
    public function store(Project $project)
    {
        $attributes = request()->validate(['description' => 'required|min:3']);

        $project->addTask($attributes);

        /*
        Task::create([
            'project_id' => $project->id,
            'description' => request('description')
        ]);
        */
        return back();
    }

// ...
```

## Model (Project)

```
    public function addTask($task)
    {
        $this->tasks()->create($task);
        // adiciona o project_id automaticamente
    }
```

Cofinanciado por:

MADEIRA 14-20 REGIÃO AUTÓNOMA DA MADEIRA PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

Esta separação é **opcional**, serve para separar, simplifica o controller ficando só a saber o que é para ser feito, se quiser saber mais (como é feito) tenho que ir ao model. A separação pode ser justificada por exemplo para a separação de responsabilidades numa equipa.

## Includes

Para incluir, em diferentes páginas/view, partes de código que repetem por várias páginas usamos o @include

**Novo ficheiro views/errors.blade.php**

```
@if ($errors->any())
<div class="notification is-danger">
    <ul>
        @foreach($errors->all() as $error)
            <li>{{ $error }}</li>
        @endforeach
    </ul>
</div>
@endif
```

Na view show.blade.php:

```
        @include('errors')
    </form>
```

## Fontes e mais recursos

https://laravel.com/
https://laracasts.com/series/laravel-from-scratch-2018
https://laravel.com/docs/6.x
https://laracasts.com/skills/laravel

Cofinanciado por:
MADEIRA 14-20 REGIÃO AUTÓNOMA DA MADEIRA PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu