

Ejercicio (individual)

Hacer un CRUD con MVC en Laravel que permita gestionar la lista de tareas de un sistema.

Crear proyecto

[Video de instalación](#)

Modelo, migración y controlador

Crear el modelo Tarea, junto con la migración y el controlador de recursos.

Opción rápida:

```
php artisan make:model Tarea -mcr
```

Iniciar proyecto Laravel

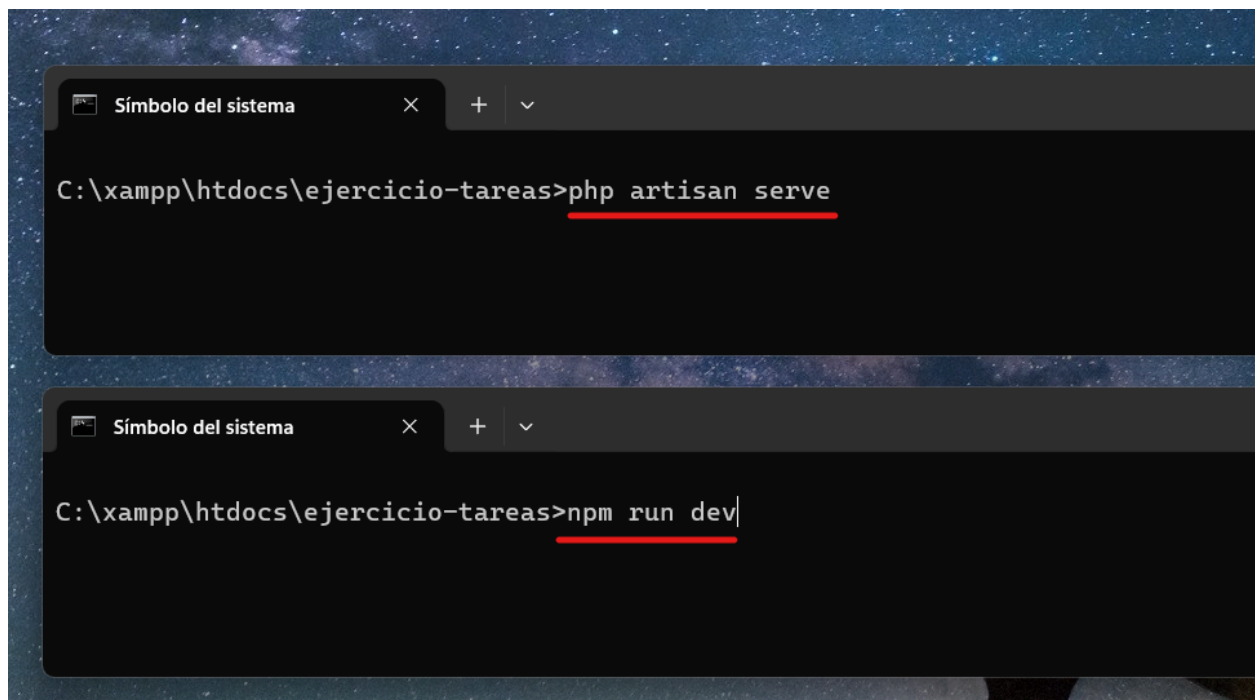
Iniciar el servidor del backend:

```
php artisan serve
```

Y el servidor del frontend:

```
npm run dev
```

En este punto se deben abrir dos consolas para levantar ambos servidores:



Levantar proyecto

Abrir el proyecto en algún editor como Visual studio code



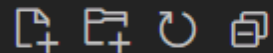
File Edit Selection View Go Run ...



EXPLORER



▼ UNTITLED (WORKSPACE)



▼ ejercicio-tareas



- > app
- > bootstrap
- > config
- > database
- > node_modules
- > public
- > resources
- > routes
- > storage
- > tests
- > vendor



⚙ .editorconfig

⚙ .env

💰 .env.example

🔒 .gitattributes

🔒 .gitignore

≡ artisan

{ } composer.json

{ } composer.lock

{ } package-lock.json

{ } package.json

📡 phpunit.xml

ℹ README.md



> OUTLINE

Conexión a la base de datos

Editar el archivo .env indicando la información para conectarse a la base de datos:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=nombre_base_de_datos
DB_USERNAME=root
DB_PASSWORD=
DB_COLLATION=utf8mb4_unicode_ci
```

Migración de tareas

Editar la migración agregando tres columnas (se pueden agregar más si así la/el alumna/o lo desea):

```
$table->string('nombre', 100);
$table->text('descripcion');
$table->boolean('resuelta')->default(false);
```

```
return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('tareas', function (Blueprint $table) {
            $table->id();
            $table->string('nombre', 100);
            $table->text('descripcion');
            $table->boolean('resuelta')->default(false);
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
```

```
{  
    Schema::dropIfExists('tareas');  
}  
};
```

Por último ejecutar la migración:

```
php artisan migrate
```

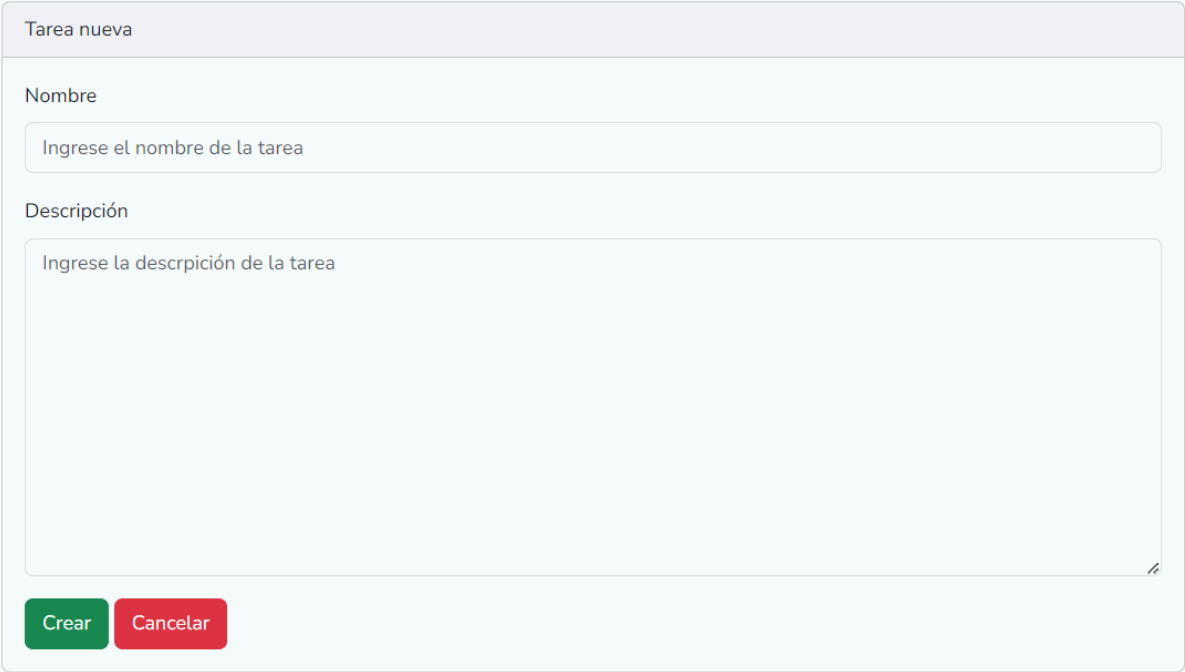
Consigna

Crear un gestor de tareas que permita:

Crear una tarea

En el método `create()`:

Mostrar un formulario que permita ingresar la información de la tarea: nombre y descripción.



Formulario para crear una nueva tarea. El formulario tiene un encabezado "Tarea nueva" en un recuadro gris. Debajo, hay dos campos de entrada: "Nombre" con un placeholder "Ingrese el nombre de la tarea" y "Descripción" con un placeholder "Ingrese la descripción de la tarea". En la parte inferior, hay dos botones: "Crear" (verde) y "Cancelar" (rojo).

En el método `store()`:

Se debe validar previamente que tanto el nombre, como la descripción de la tarea estén validados. Y además que el nombre no tenga un máximo de 50 caracteres.

- The nombre field is required.
- The descripcion field is required.

Por último crear el registro con el nombre, descripción y la columna: 'resuelta' con el valor false.

Finalmente redirigir al método index()

Mostrar lista de tareas

En el método index():

Mostrar una tabla que se vea de una forma similar:

Tarea nueva			
Agregar nueva			
Nombre	Descripción	Resuelta	Acciones
Lavar la ropa	El canasto está lleno.	No	Editar Eliminar
Pasear al perro	Rufus quiere ejercitarse	Sí	Editar Eliminar

Editar tarea

En el método edit():

Mostrar un formulario similar al del método create(), pero que muestre un checkbox, un select, o cualquier otro elemento de formulario que permita cambiar el estado de la tarea de si está resuelta o no:

Editar tarea

Nombre

Lavar la ropa

Descripción

El canasto está lleno

☒ Resuelta

Modificar

Cancelar

En el método update():

Se debe validar previamente que tanto el nombre, como la descripción de la tarea estén validados. Y además que el nombre no tenga un máximo de 50 caracteres.

Por último crear el registro con el nombre, descripción y la columna: 'resuelta' dependiendo de lo que haya elegido el usuario.

En caso de usar un checkbox, recordar que se debe enviar un valor boolean:

Vista:

```
<input type="checkbox" name="resuelta" value="1"
@checked($tarea->resuelta)>
```

Controlador:

```
'resuelta' => ($request->resuelta == '1')
```

Finalmente redirigir al método index()

Eliminar tarea

Al hacer click en el botón eliminar se debe redirigir al método `destroy()`.

Recordar que para estos casos la acción debe enviar por el verbo DELETE, por tanto el botón debe estar contenido en un formulario.