

## ÍNDICE

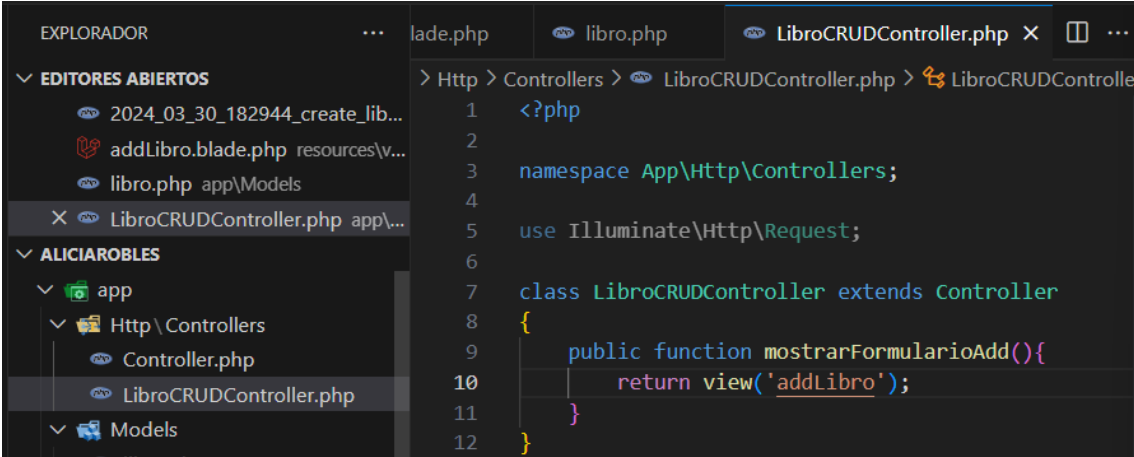
1. Creación del Modelo de Libros: Genera un modelo y, un controlador para esta tabla2	
2. Controlador de Libros: En el controlador de libros, añade funciones para el CRUD (BUSCAR, INSERTAR, BORRAR, ACTUALIZAR) de libros.....	2
3. Vistas de Libros: Crea vistas usando blade para mostrar todos los libros, ver detalles de un libro, añadir un nuevo libro y editar un libro existente.....	5
A) MOSTRAR TODOS LOS LIBROS.....	5
A) VER DETALLES DE UN LIBRO .....	7
B) AÑADIR UN NUEVO LIBRO .....	8
C) EDITAR UN LIBRO EXISTENTE .....	11
4. Creación del Modelo de Préstamos: Genera un modelo y un controlador. ....	14
5. Controlador de Préstamos: En el controlador de préstamos, añade funciones para el CRUD de préstamos. Recuerda que debes validar que el libro esté disponible antes de permitir el préstamo y actualizar el estado del libro cuando se realice un préstamo o una devolución.....	14
6. Vistas de Préstamos: Crea vistas para mostrar todos los préstamos, ver detalles de un préstamo, añadir un nuevo préstamo y finalizar un préstamo existente. (El usuario sabremos cual es en la siguiente practica) .....	17
A) MOSTRAR TODOS LOS PRÉSTAMOS.....	17
A) DETALLES DE UN PRÉSTAMO .....	18
B) AÑADIR UN NUEVO PRÉSTAMO .....	19
C) FINALIZAR UN PRÉSTAMO EXISTENTE .....	20
7. Rutas: Añade rutas en tu archivo web.php para cada acción en los controladores de libros y préstamos.....	22

## 1. Creación del Modelo de Libros: Genera un modelo y, un controlador para esta tabla

Voy a continuar con la app de la práctica anterior y sobre ella crearé modelo-vista controlador.

En la práctica anterior generé el modelo Libro, ahora le vamos a crear el controlador:

```
PS C:\aliciarobles> php artisan make:controller LibroCRUDController
```



```
<?php
namespace App\Http\Controllers;

use Illuminate\Http\Request;

class LibroCRUDController extends Controller
{
    public function mostrarFormularioAdd(){
        return view('addLibro');
    }
}
```

## 2. Controlador de Libros: En el controlador de libros, añade funciones para el CRUD (BUSCAR, INSERTAR, BORRAR, ACTUALIZAR) de libros.

-En el modelo libro, añadimos métodos como:

Obtener todos

Obtener por id

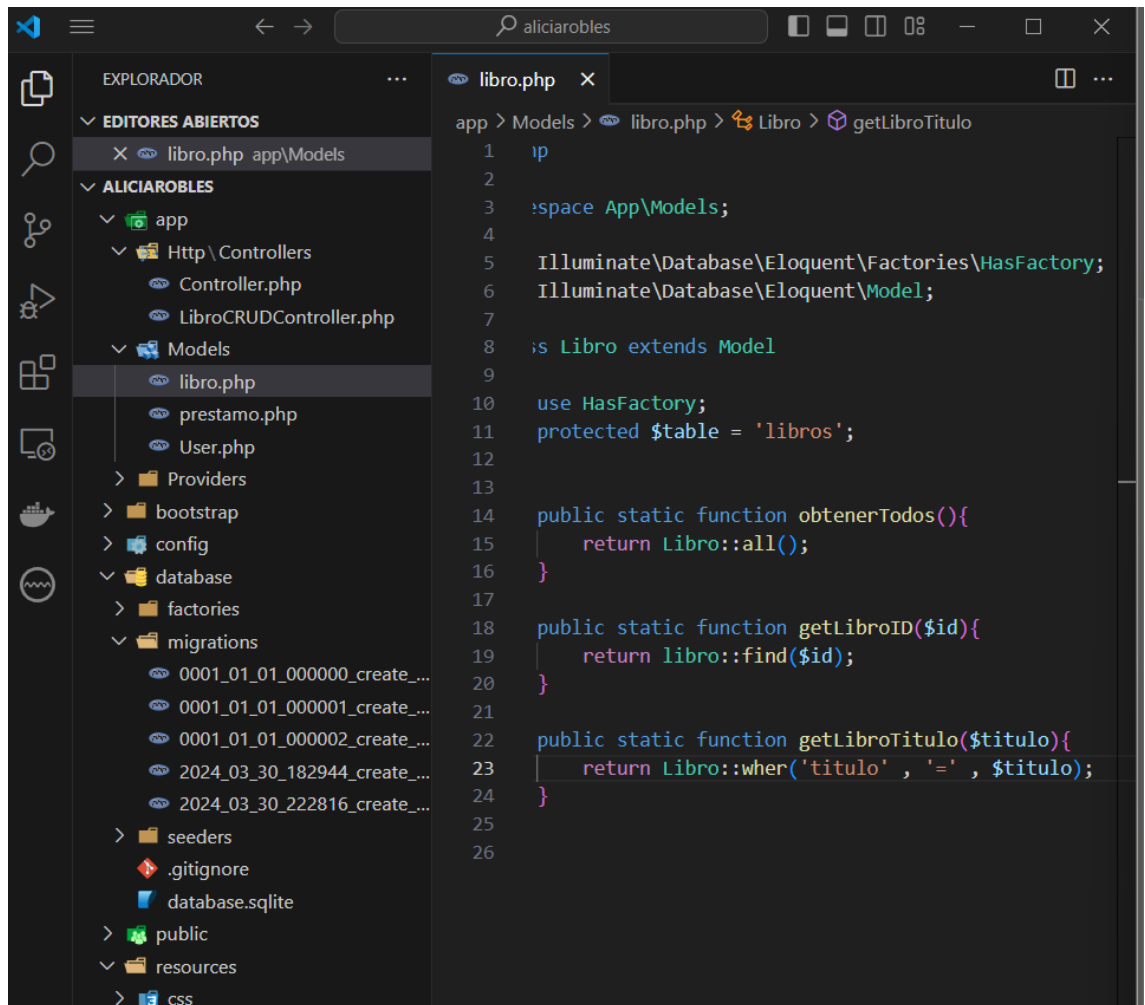
Obtener por titulo

Crear nuevo libro

Actualizar

Borrar

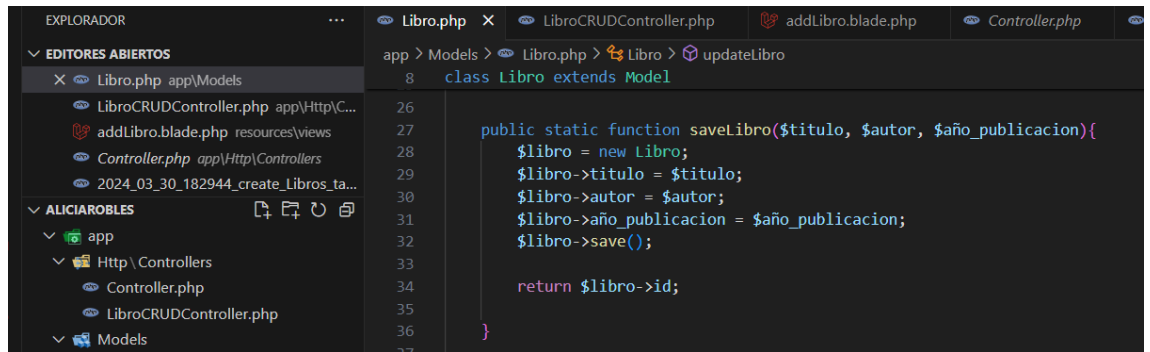
En el archivo *Libro.php* escribo las funciones para: Obtener todos, Obtener por id y Obtener por título:



The screenshot shows the Visual Studio Code interface. On the left, the 'EXPLORADOR' (Explorer) sidebar displays the project structure. Under 'ALICIA ROBLES' > 'app' > 'Models', the file 'libro.php' is selected. The main editor window shows the code for 'libro.php' with the following content:

```
1  namespace App\Models;
2
3  use Illuminate\Database\Eloquent\Factories\HasFactory;
4  use Illuminate\Database\Eloquent\Model;
5
6  class Libro extends Model
7  {
8      use HasFactory;
9      protected $table = 'libros';
10
11      public static function obtenerTodos(){
12          return Libro::all();
13      }
14
15      public static function getLibroID($id){
16          return libro::find($id);
17      }
18
19      public static function getLibroTitulo($titulo){
20          return Libro::wher('titulo' , '=' , $titulo);
21      }
22  }
```

Crear nuevo libro:

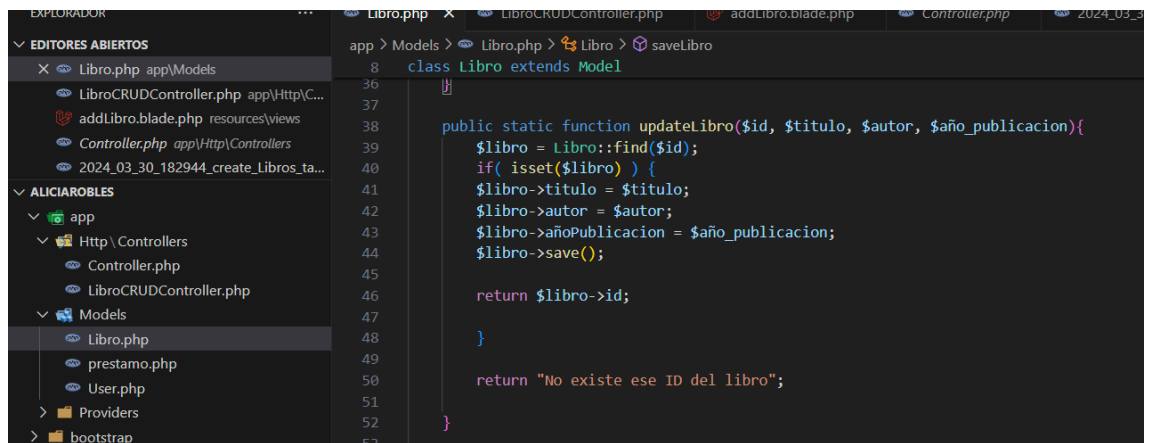


The screenshot shows the Visual Studio Code editor with the 'Libro.php' file open in the 'app/Models' directory. The code defines a 'Libro' class extending 'Model'. A static method 'saveLibro' is implemented, which creates a new 'Libro' instance, sets its 'titulo', 'autor', and 'año\_publicacion' attributes, and then calls 'save()' to persist the record. The method returns the ID of the saved book.

```
class Libro extends Model
{
    public static function saveLibro($titulo, $autor, $año_publicacion){
        $libro = new Libro;
        $libro->titulo = $titulo;
        $libro->autor = $autor;
        $libro->año_publicacion = $año_publicacion;
        $libro->save();

        return $libro->id;
    }
}
```

Actualizar libro:



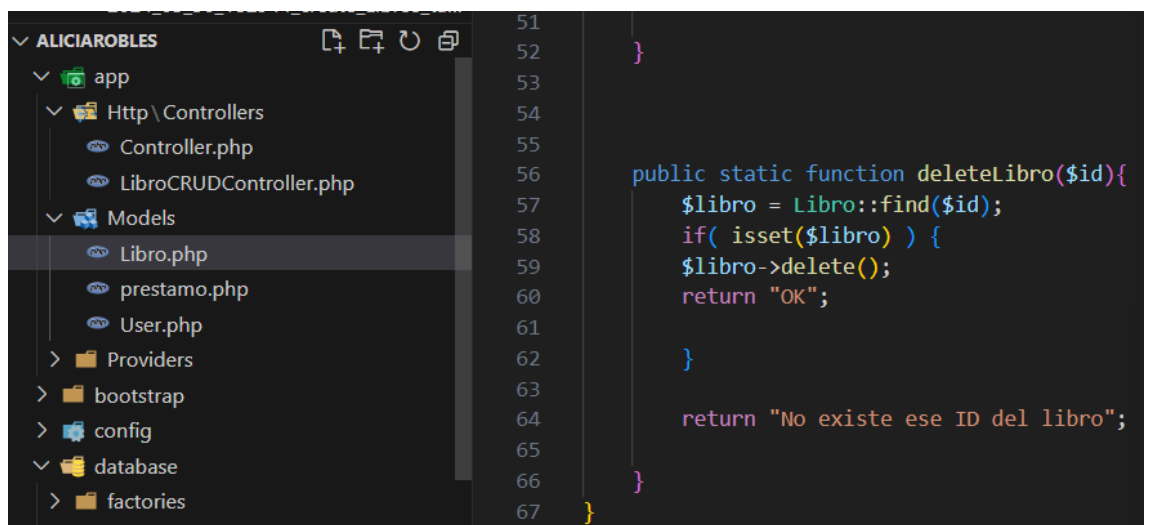
The screenshot shows the Visual Studio Code editor with the 'Libro.php' file open in the 'app/Models' directory. The code defines a 'Libro' class extending 'Model'. A static method 'updateLibro' is implemented, which finds a book by its ID, checks if it exists using 'isset', and then updates its 'titulo', 'autor', and 'año\_publicacion' attributes. It then calls 'save()' to update the record. If the book does not exist, it returns a message: 'No existe ese ID del libro'.

```
class Libro extends Model
{
    public static function updateLibro($id, $titulo, $autor, $año_publicacion){
        $libro = Libro::find($id);
        if( isset($libro) ) {
            $libro->titulo = $titulo;
            $libro->autor = $autor;
            $libro->año_publicacion = $año_publicacion;
            $libro->save();

            return $libro->id;
        }

        return "No existe ese ID del libro";
    }
}
```

Borrar:



The screenshot shows the Visual Studio Code editor with the 'Libro.php' file open in the 'app/Models' directory. The code defines a 'Libro' class extending 'Model'. A static method 'deleteLibro' is implemented, which finds a book by its ID, checks if it exists using 'isset', and then calls 'delete()' to remove the record. It returns 'OK' if successful. If the book does not exist, it returns a message: 'No existe ese ID del libro'.

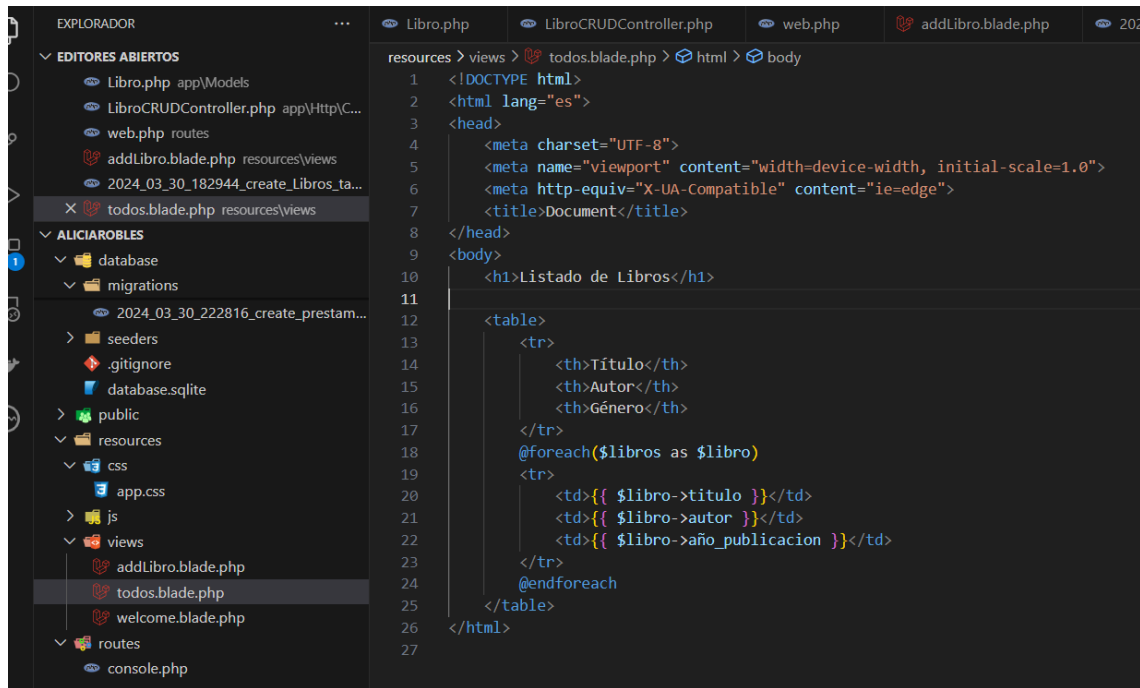
```
class Libro extends Model
{
    public static function deleteLibro($id){
        $libro = Libro::find($id);
        if( isset($libro) ) {
            $libro->delete();
            return "OK";
        }

        return "No existe ese ID del libro";
    }
}
```

### 3. Vistas de Libros: Crea vistas usando blade para mostrar todos los libros, ver detalles de un libro, añadir un nuevo libro y editar un libro existente.

#### A) MOSTRAR TODOS LOS LIBROS

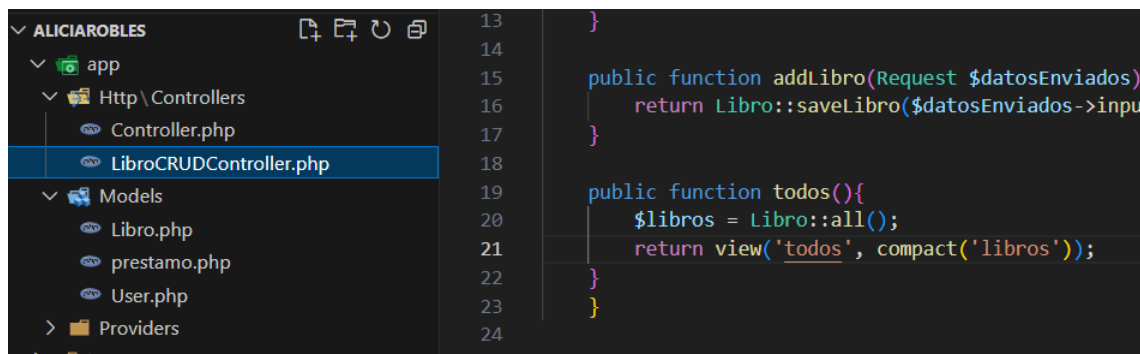
Voy a crear un nuevo archivo llamado *todos.blade.php* para obtener una nueva vista:



The screenshot shows the Visual Studio Code editor with the file explorer on the left and the code editor on the right. The file explorer shows the project structure with the 'resources' folder expanded, and 'views' > 'todos.blade.php' selected. The code editor shows the content of 'todos.blade.php' with the following code:

```
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <meta http-equiv="X-UA-Compatible" content="ie=edge">
7   <title>Document</title>
8 </head>
9 <body>
10  <h1>Listado de Libros</h1>
11
12  <table>
13    <tr>
14      <th>Título</th>
15      <th>Autor</th>
16      <th>Género</th>
17    </tr>
18    @foreach($libros as $libro)
19      <tr>
20        <td>{{ $libro->titulo }}</td>
21        <td>{{ $libro->autor }}</td>
22        <td>{{ $libro->año_publicacion }}</td>
23      </tr>
24    @endforeach
25  </table>
26 </body>
27 </html>
```

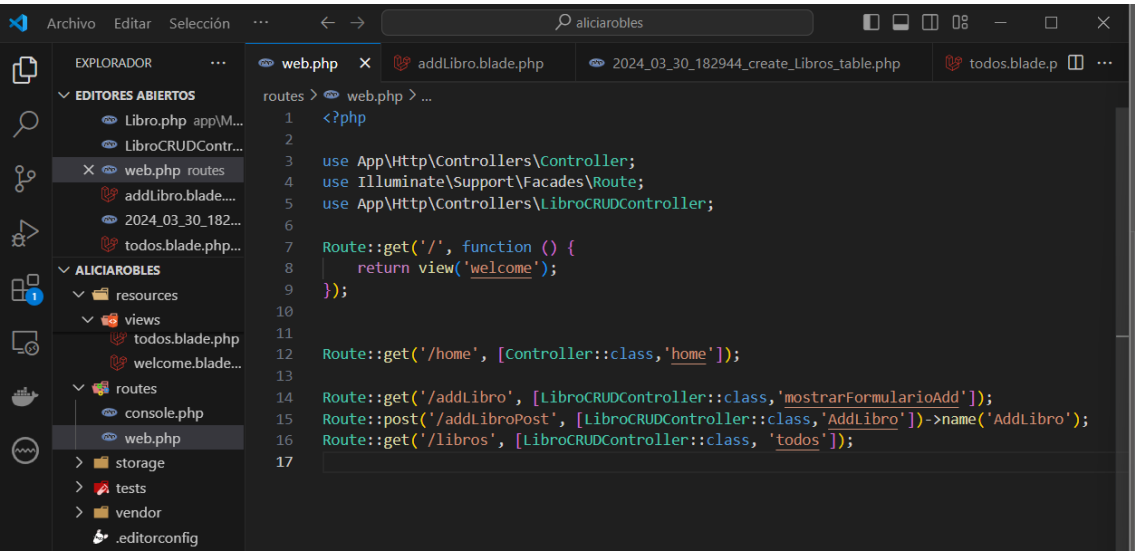
En el controlador añado la función de mostrar todos:



The screenshot shows the Visual Studio Code editor with the file explorer on the left and the code editor on the right. The file explorer shows the project structure with the 'app' folder expanded, and 'Http' > 'Controllers' > 'LibroCRUDController.php' selected. The code editor shows the content of 'LibroCRUDController.php' with the following code:

```
13 }
14
15 public function addLibro(Request $datosEnviados)
16 {
17   return Libro::saveLibro($datosEnviados->input);
18 }
19
20 public function todos(){
21   $libros = Libro::all();
22   return view('todos', compact('libros'));
23 }
24
25 }
```

Y por último añado la ruta:

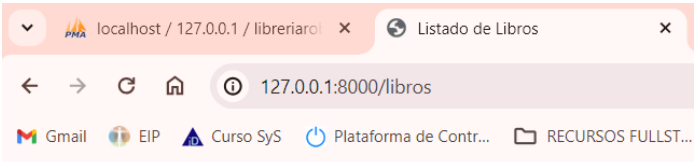


The screenshot shows a code editor with the following content:

```
routes > web.php > ...
1  <?php
2
3  use App\Http\Controllers\Controller;
4  use Illuminate\Support\Facades\Route;
5  use App\Http\Controllers\LibroCRUDController;
6
7  Route::get('/', function () {
8      return view('welcome');
9  });
10
11
12 Route::get('/home', [Controller::class, 'home']);
13
14 Route::get('/addLibro', [LibroCRUDController::class, 'mostrarFormularioAdd']);
15 Route::post('/addLibroPost', [LibroCRUDController::class, 'AddLibro'])->name('AddLibro');
16 Route::get('/libros', [LibroCRUDController::class, 'todos']);
17
```

The Explorer panel on the left shows the project structure:

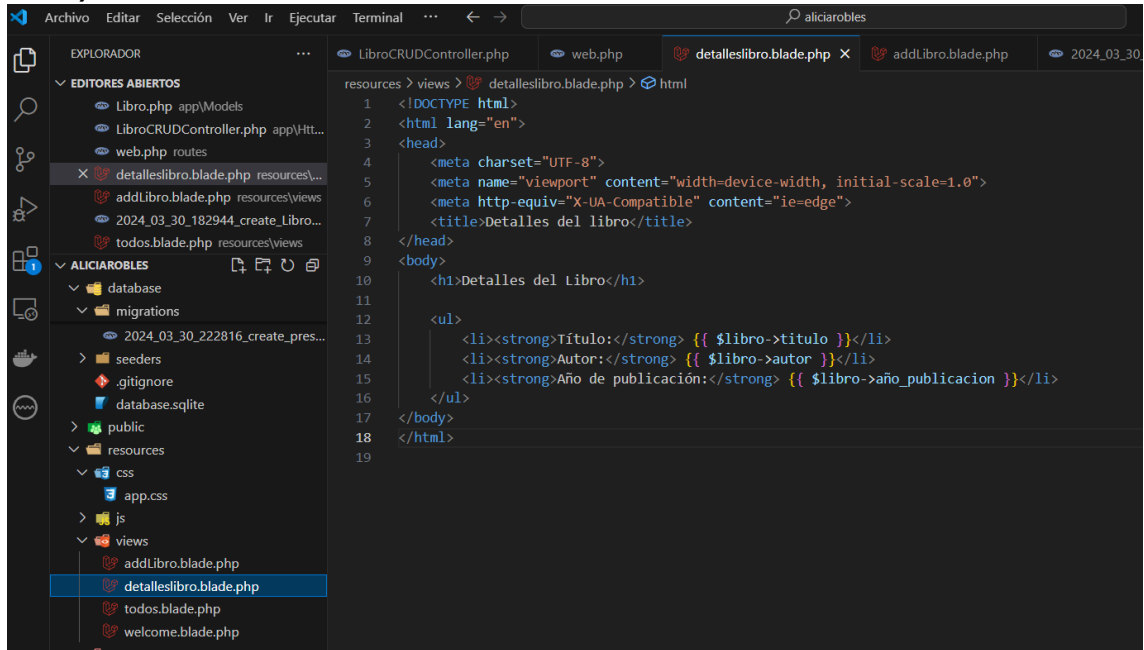
- LIBRO.php app\M...
- LibroCRUDContr...
- web.php routes
- addLibro.blade....
- 2024\_03\_30\_182...
- todos.blade.php...
- ALICIAROBLES
  - resources
  - views
    - todos.blade.php
    - welcome.blade...
  - routes
    - console.php
    - web.php
  - storage
  - tests
  - vendor
  - .editorconfig



## Listado de Libros

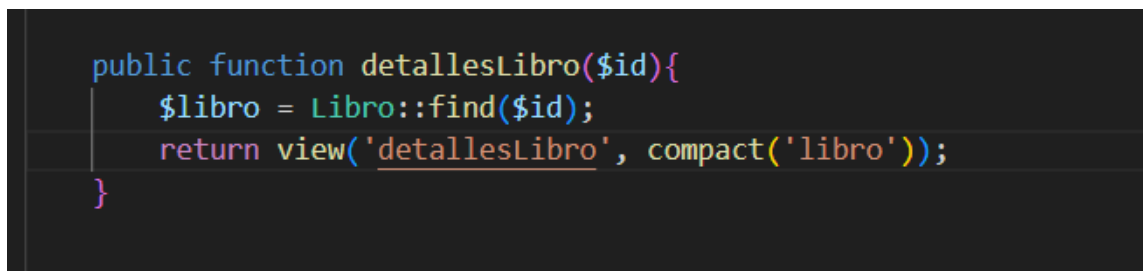
Título	Autor	Género
Los Pilares de la Tierra	Ken Follett	1989
La mano de Fátima	Idelfonso Falcones	2010
Juego de Tronos	George R.R. Martin	1996
Danza de dragones	George R.R. Martin	2011
Un mundo feliz	Aldous Huxley	1932
El niño con el pijama de rayas	John Boyne	2006

## A) VER DETALLES DE UN LIBRO



The screenshot shows the Visual Studio Code editor with the file explorer on the left and the code editor on the right. The file explorer shows the project structure with the 'resources' folder expanded, and 'detalleslibro.blade.php' selected. The code editor displays the following HTML code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <meta http-equiv="X-UA-Compatible" content="ie=edge">
7     <title>Detalles del libro</title>
8 </head>
9 <body>
10     <h1>Detalles del Libro</h1>
11
12     <ul>
13         <li><strong>Titulo:</strong> {{ $libro->titulo }}</li>
14         <li><strong>Autor:</strong> {{ $libro->autor }}</li>
15         <li><strong>Año de publicación:</strong> {{ $libro->año_publicacion }}</li>
16     </ul>
17 </body>
18 </html>
19
```



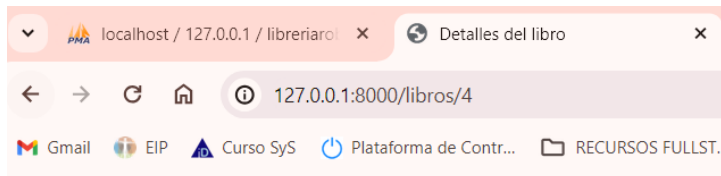
The screenshot shows a PHP function definition in a code editor. The function is named 'detallesLibro' and takes a parameter '\$id'. It uses the 'Libro::find' method to retrieve a book by its ID and then returns a Blade view with the book's details.

```
public function detallesLibro($id){
    $libro = Libro::find($id);
    return view('detallesLibro', compact('libro'));
}
```



The screenshot shows the Visual Studio Code editor with the file explorer on the left and the code editor on the right. The file explorer shows the 'routes' folder expanded, and 'web.php' selected. The code editor displays the following route definitions:

```
10
11
12 Route::get('/home', [Controller::class, 'home']);
13
14 Route::get('/addLibro', [LibroCRUDController::class, 'mostrarFormularioAdd']);
15 Route::post('/addLibroPost', [LibroCRUDController::class, 'AddLibro'])->name('AddLibro');
16 Route::get('/libros', [LibroCRUDController::class, 'todos']);
17 Route::get('/libros/{id}', [LibroCRUDController::class, 'detalleslibro']);
18
```

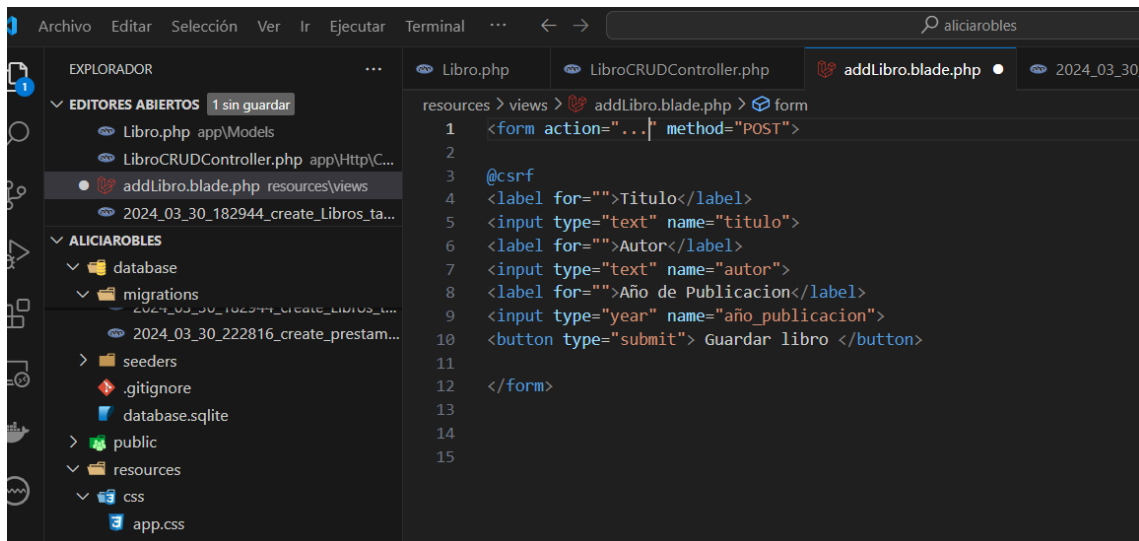


## Detalles del Libro

- **Título:** Danza de dragones
- **Autor:** George R.R. Martin
- **Año de publicación:** 2011

## B) AÑADIR UN NUEVO LIBRO

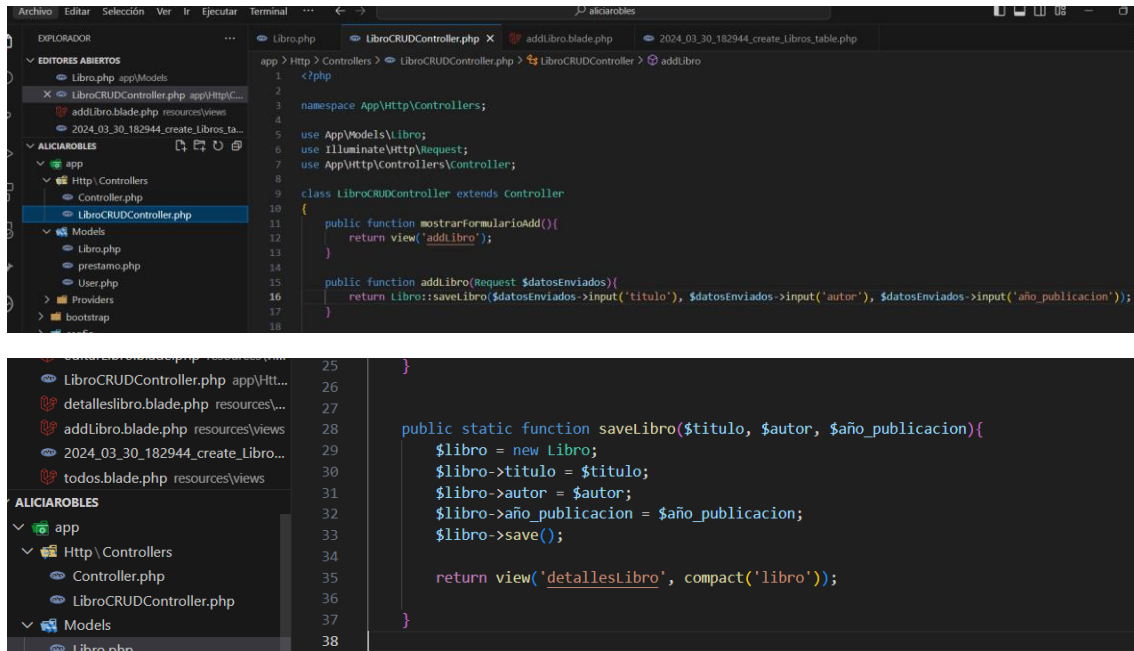
-Voy a crear un nuevo archivo llamado *addLibro.blade.php* para obtener una nueva vista, que será un formulario:



En el controlador añado las funciones *mostrarFormularioAdd* y *addLibro* que llama a la función *saveLibro* en el modelo *Libro*:



## Alicia Robles Remacho Actividad 2 Laravel

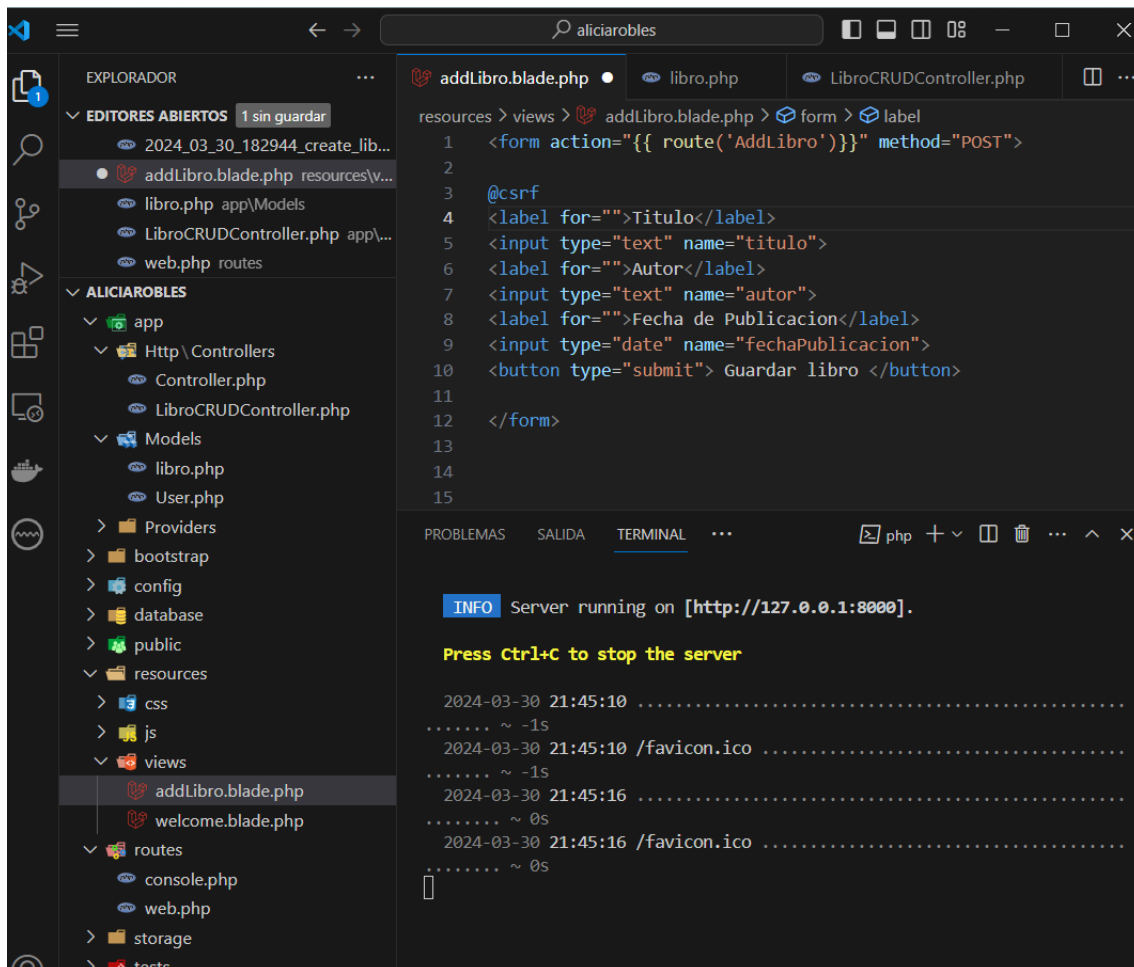


```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use App\Models\Libro;
6 use Illuminate\Http\Request;
7 use App\Http\Controllers\Controller;
8
9 class LibroCRUDController extends Controller
10 {
11     public function mostrarFormularioAdd()
12     {
13         return view('addLibro');
14     }
15
16     public function addLibro(Request $datosEnviados)
17     {
18         return Libro::saveLibro($datosEnviados->input('titulo'), $datosEnviados->input('autor'), $datosEnviados->input('año_publicacion'));
19     }
20
21     public static function saveLibro($titulo, $autor, $año_publicacion){
22         $libro = new Libro;
23         $libro->titulo = $titulo;
24         $libro->autor = $autor;
25         $libro->año_publicacion = $año_publicacion;
26         $libro->save();
27
28         return view('detallesLibro', compact('libro'));
29     }
30 }
```

Le añado la ruta:

```
Route::get('/addLibro', [LibroCRUDController::class, 'mostrarFormularioAdd']);
Route::post('/addLibroPost', [LibroCRUDController::class, 'AddLibro'])->name('AddLibro');
```

Y lo enlazo poniendo en el action del form mi función:



```
1 <form action="{{ route('AddLibro') }}" method="POST">
2
3 @csrf
4 <label for="">Titulo</label>
5 <input type="text" name="titulo">
6 <label for="">Autor</label>
7 <input type="text" name="autor">
8 <label for="">Fecha de Publicacion</label>
9 <input type="date" name="fechaPublicacion">
10 <button type="submit"> Guardar libro </button>
11
12 </form>
13
14
15
```

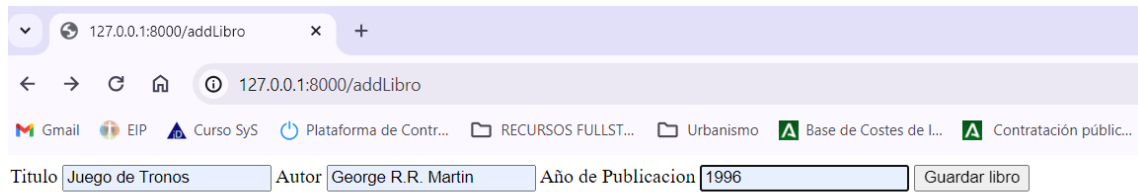
PROBLEMAS SALIDA TERMINAL

INFO Server running on [http://127.0.0.1:8000].

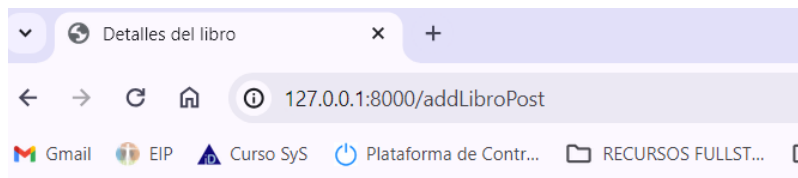
Press Ctrl+C to stop the server

```
2024-03-30 21:45:10 ..... ~ -1s
2024-03-30 21:45:10 /favicon.ico ..... ~ -1s
2024-03-30 21:45:16 ..... ~ 0s
2024-03-30 21:45:16 /favicon.ico ..... ~ 0s
```

Pongo en la terminal *php artisan serve* y lo tengo.



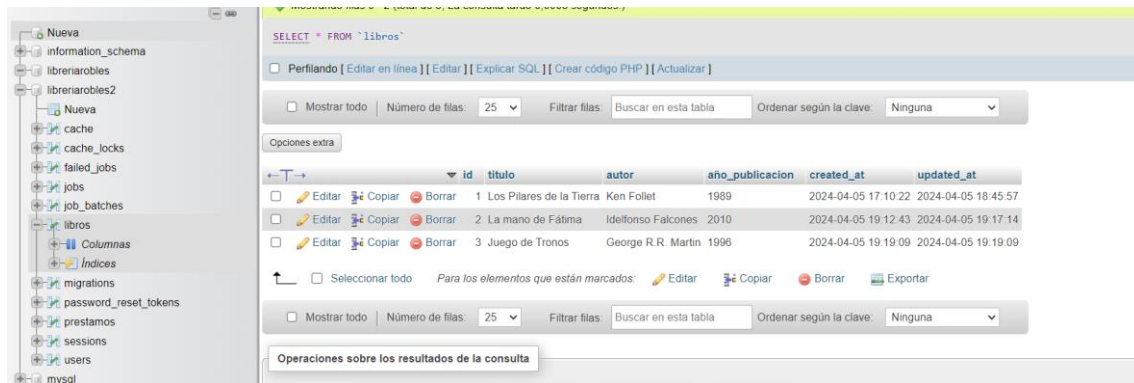
Una vez que le doy a guardar:



## Detalles del Libro

- **Título:** Juego de Tronos
- **Autor:** George R.R. Martin
- **Año de publicación:** 1996

Me añade en la base de datos el registro que he creado:



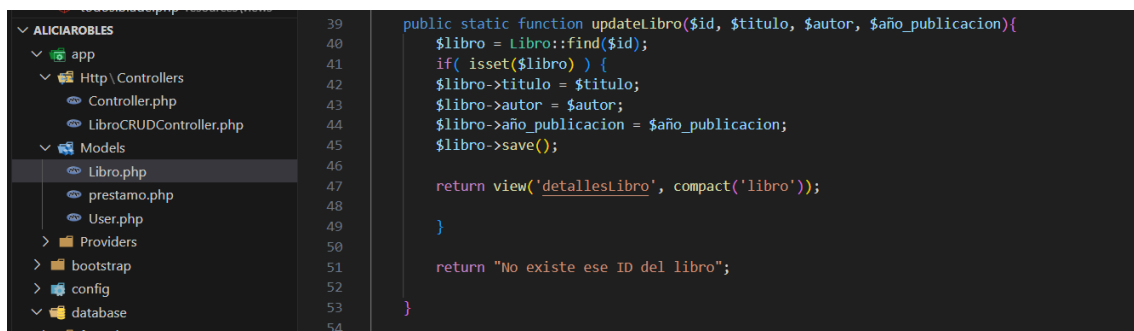
### C) EDITAR UN LIBRO EXISTENTE

Sigo los mismos pasos que el apartado D. Creo 2 funciones en el controlador:

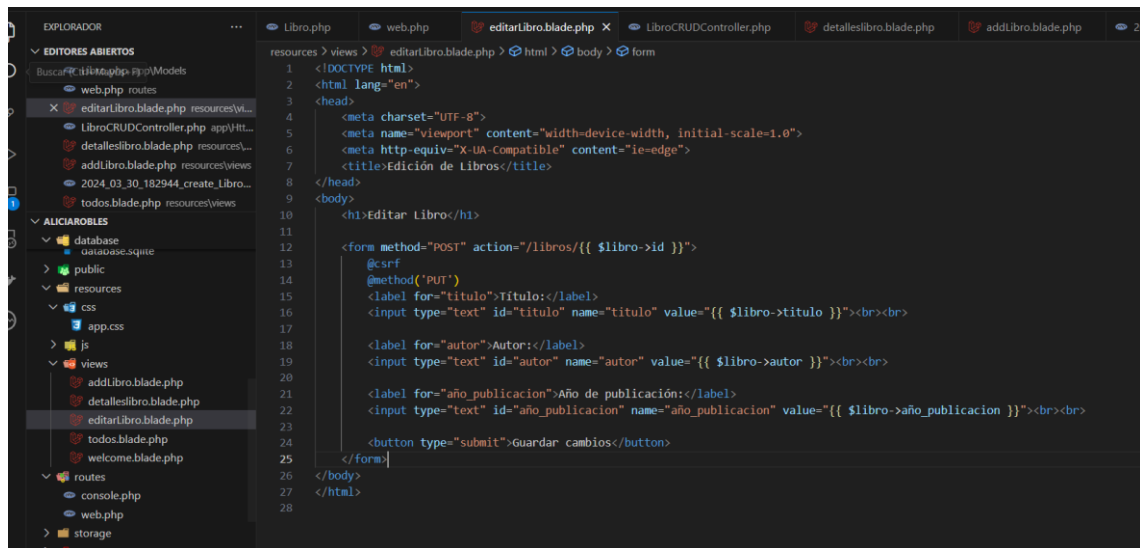
```
public function editarLibro($id)
{
    $libro = Libro::findOrFail($id);
    return view('editarLibro', compact('libro'));
}

public static function update(Request $datosEnviados,$id){
    return Libro::updateLibro($id, $datosEnviados->input('titulo'), $datosEnviados->input('autor'), $datosEnviados->input('año_publicacion'));
}
```

Donde la función updateLibro está en en el model Libro.php:



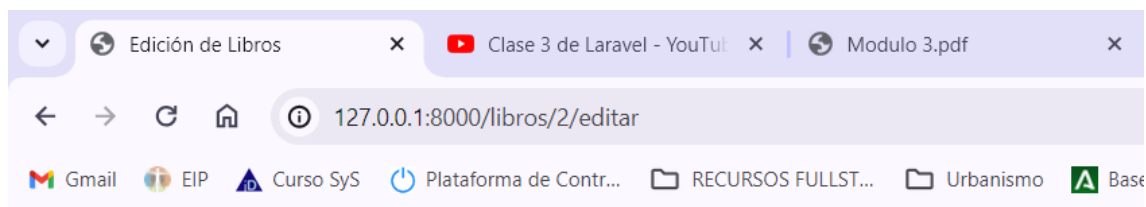
Creo la vista del formulario para editar:



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 <meta http-equiv="X-UA-Compatible" content="ie=edge">
7 <title>Edición de Libros</title>
8 </head>
9 <body>
10 <h1>Editar Libro</h1>
11
12 <form method="POST" action="/libros/{{ $libro->id }}">
13     @csrf
14     @method("PUT")
15     <label for="titulo">Titulo:</label>
16     <input type="text" id="titulo" name="titulo" value="{{ $libro->titulo }}"><br><br>
17     <label for="autor">Autor:</label>
18     <input type="text" id="autor" name="autor" value="{{ $libro->autor }}"><br><br>
19     <label for="año_publicacion">Año de publicación:</label>
20     <input type="text" id="año_publicacion" name="año_publicacion" value="{{ $libro->año_publicacion }}"><br><br>
21     <button type="submit">Guardar cambios</button>
22 </form>
23 </body>
24 </html>
```

Y creo las rutas:

```
18 Route::get('/libros/{id}/editar', [LibroCRUDController::class, 'editarLibro']);
19 Route::put('/libros/{id}', [LibroCRUDController::class, 'update'])->name('update');
```

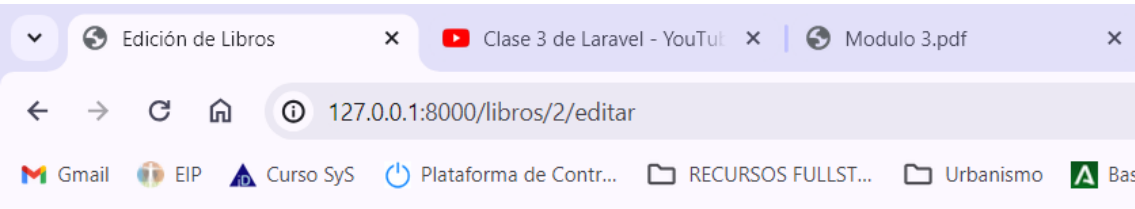


## Editar Libro

Título:

Autor:

Año de publicación:

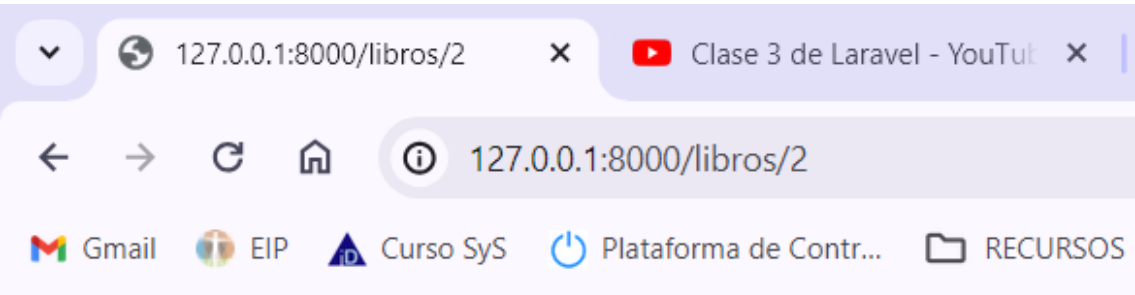


# Editar Libro

Título:

Autor:

Año de publicación:

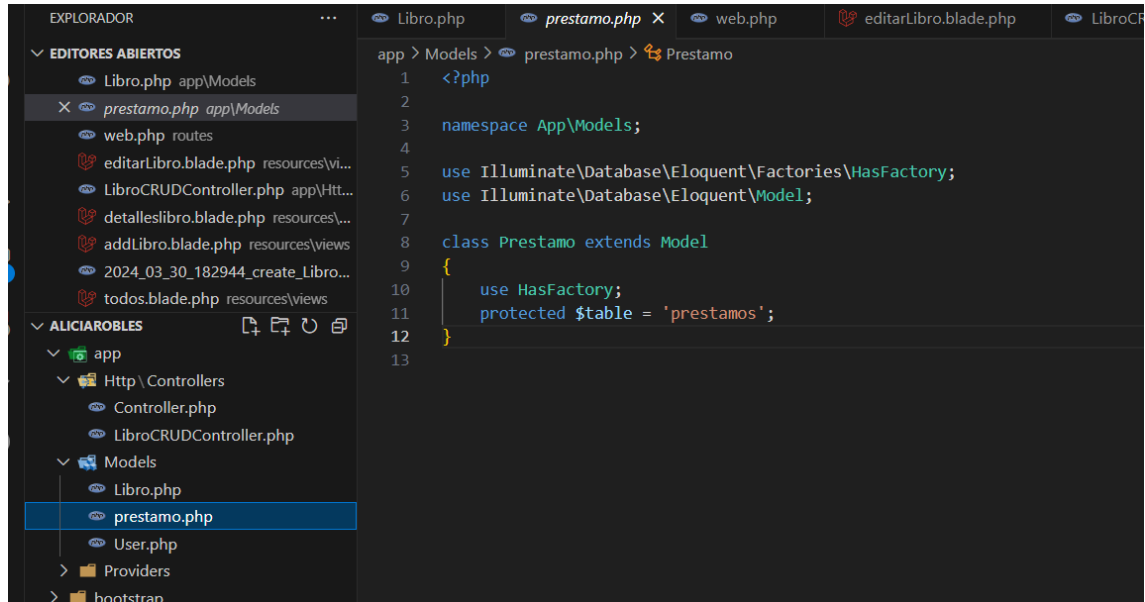


2



## 4. Creación del Modelo de Préstamos: Genera un modelo y un controlador.

Tengo el modelo de préstamos creado:

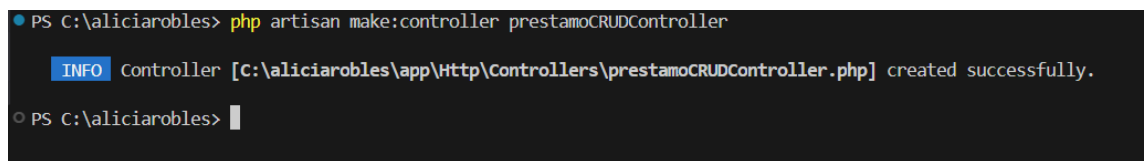


The screenshot shows an IDE with the following content:

- EXPLORADOR (Left Panel):** Shows the project structure. Under 'MODELS', the file 'prestamo.php' is selected.
- EDITOR (Main Panel):** Displays the code for 'app > Models > prestamo.php'. The code is as follows:

```
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class Prestamo extends Model
9 {
10     use HasFactory;
11     protected $table = 'prestamos';
12 }
13
```

Y ahora creo un controlador para este modelo:



The screenshot shows a terminal window with the following commands and output:

```
PS C:\aliciarobles> php artisan make:controller prestamoCRUDController

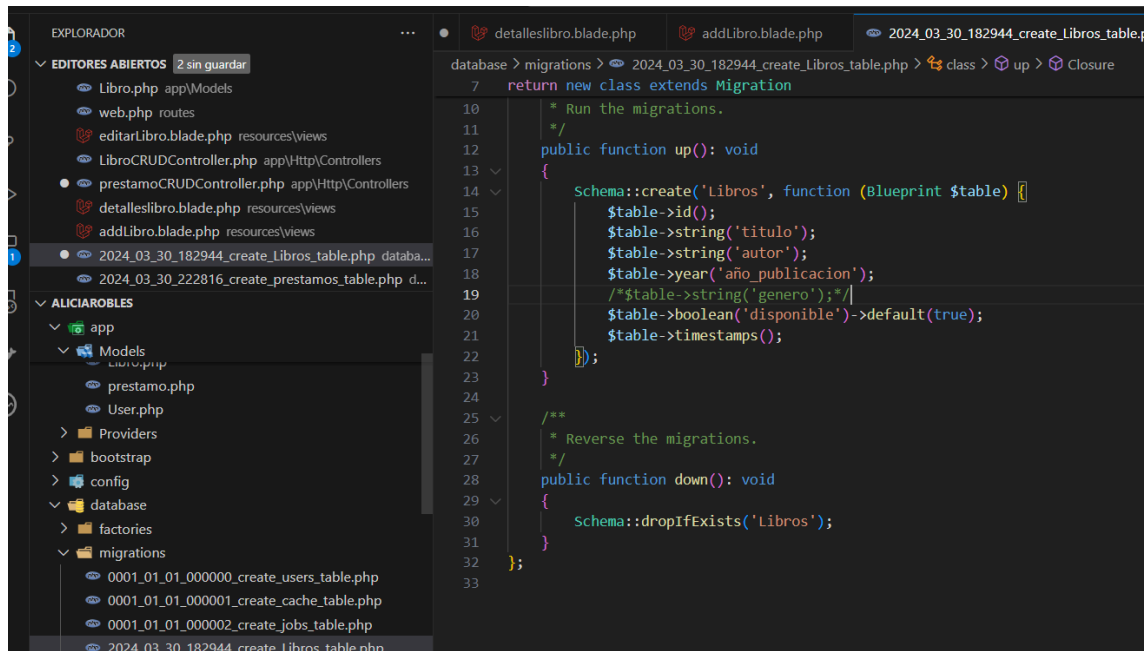
INFO Controller [C:\aliciarobles\app\Http\Controllers\prestamoCRUDController.php] created successfully.

PS C:\aliciarobles>
```

## 5. Controlador de Préstamos: En el controlador de préstamos, añade funciones para el CRUD de préstamos. Recuerda que debes validar que el libro esté disponible antes de permitir el préstamo y actualizar el estado del libro cuando se realice un préstamo o una devolución.

Voy a añadir el campo nuevo en la tabla de libros que diga si el libro esta disponible o no:

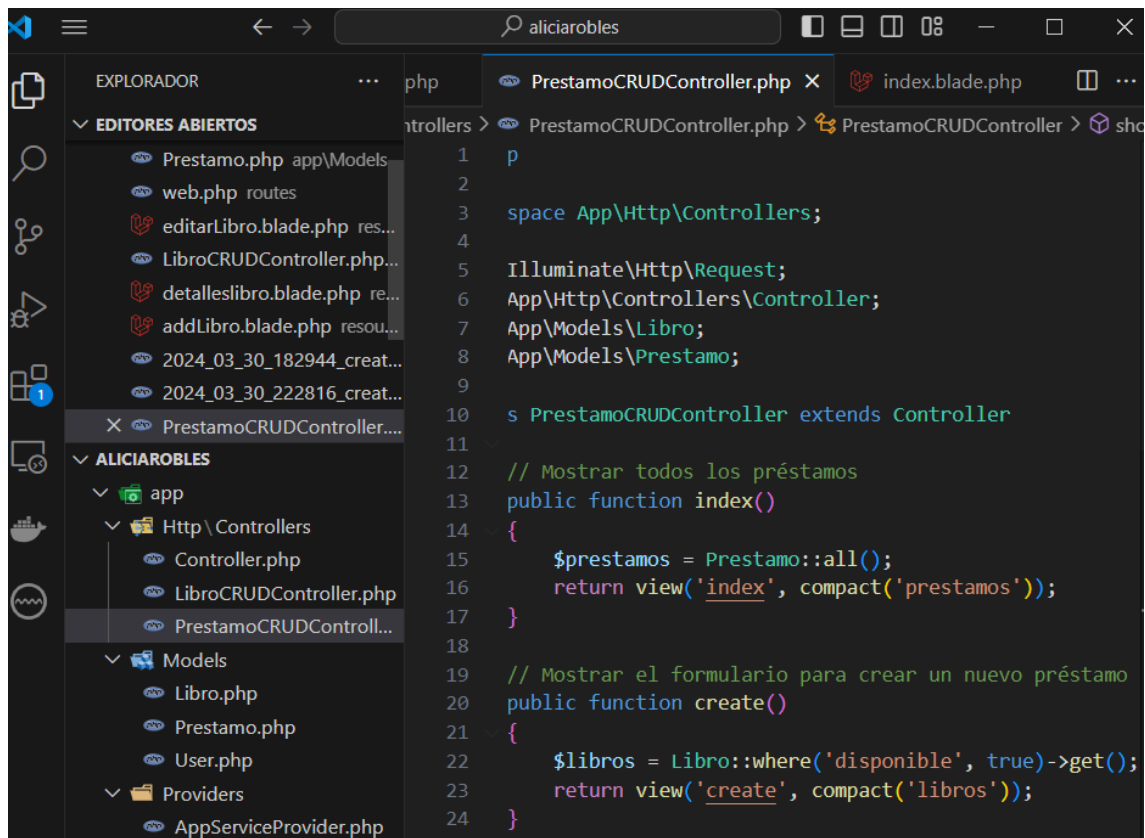
## Alicia Robles Remacho Actividad 2 Laravel



The screenshot shows the VS Code editor with a file explorer on the left and a code editor on the right. The file explorer shows the project structure, including the 'migrations' folder. The code editor displays the content of the file '2024\_03\_30\_182944\_create\_Libros\_table.php', which is a Laravel migration. The code defines a 'Libros' table with columns: 'id', 'titulo', 'autor', 'año\_publicacion', 'genero', 'disponible' (with a default value of true), and 'timestamps'.

```
database > migrations > 2024_03_30_182944_create_Libros_table.php > class > up > Closure
7 return new class extends Migration
10 {
11     * Run the migrations.
12     */
13     public function up(): void
14     {
15         Schema::create('Libros', function (Blueprint $table) {
16             $table->id();
17             $table->string('titulo');
18             $table->string('autor');
19             $table->year('año_publicacion');
20             /*$table->string('genero');*/
21             $table->boolean('disponible')->default(true);
22             $table->timestamps();
23         });
24     }
25     /**
26      * Reverse the migrations.
27      */
28     public function down(): void
29     {
30         Schema::dropIfExists('Libros');
31     }
32 };
```

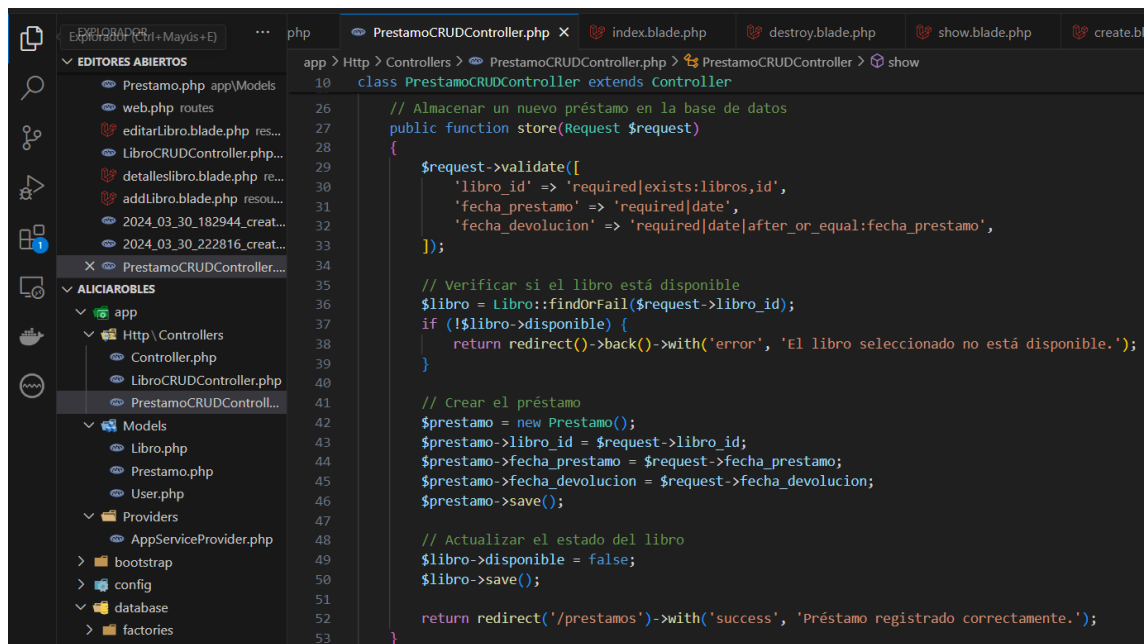
Y ahora añado en el controlador de préstamos, todas las funciones que voy a necesitar para el CRUD:



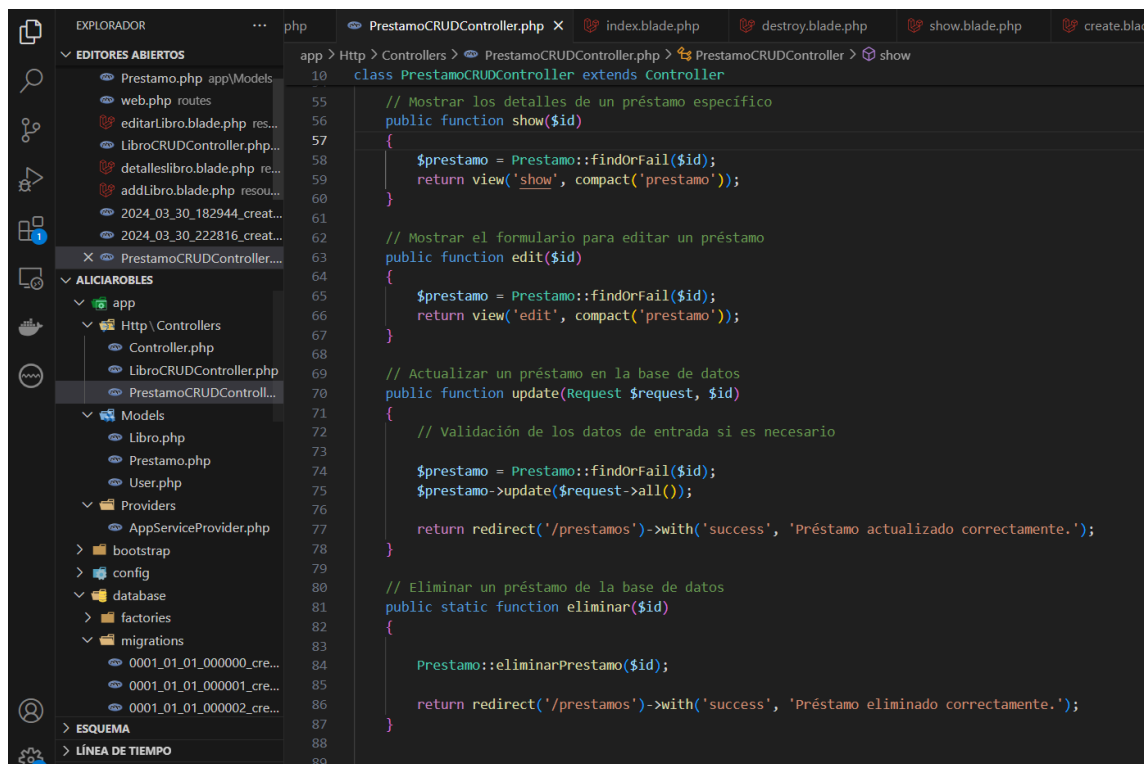
The screenshot shows the VS Code editor with a file explorer on the left and a code editor on the right. The file explorer shows the project structure, including the 'Http\ Controllers' folder. The code editor displays the content of the file 'PrestamoCRUDController.php', which is a Laravel controller. The code defines the 'PrestamoCRUDController' class, which extends 'Controller'. It includes two methods: 'index()' and 'create()'. The 'index()' method returns a view with the compacted array of all 'prestamos'. The 'create()' method returns a view with the compacted array of 'libros' where 'disponible' is true.

```
php
PrestamoCRUDController.php
index.blade.php
PrestamoCRUDController.php > PrestamoCRUDController > show
1 p
2
3 space App\Http\Controllers;
4
5 Illuminate\Http\Request;
6 App\Http\Controllers\Controller;
7 App\Models\Libro;
8 App\Models\Prestamo;
9
10 s PrestamoCRUDController extends Controller
11
12 // Mostrar todos los préstamos
13 public function index()
14 {
15     $prestamos = Prestamo::all();
16     return view('index', compact('prestamos'));
17 }
18
19 // Mostrar el formulario para crear un nuevo préstamo
20 public function create()
21 {
22     $libros = Libro::where('disponible', true)->get();
23     return view('create', compact('libros'));
24 }
25
```

## Alicia Robles Remacho Actividad 2 Laravel



```
10 class PrestamoCRUDController extends Controller
11 {
12     // Almacenar un nuevo préstamo en la base de datos
13     public function store(Request $request)
14     {
15         $request->validate([
16             'libro_id' => 'required|exists:libros,id',
17             'fecha_prestamo' => 'required|date',
18             'fecha_devolucion' => 'required|date|after_or_equal:fecha_prestamo',
19         ]);
20
21         // Verificar si el libro está disponible
22         $libro = Libro::findOrFail($request->libro_id);
23         if (!$libro->disponible) {
24             return redirect()->back()->with('error', 'El libro seleccionado no está disponible.');
```

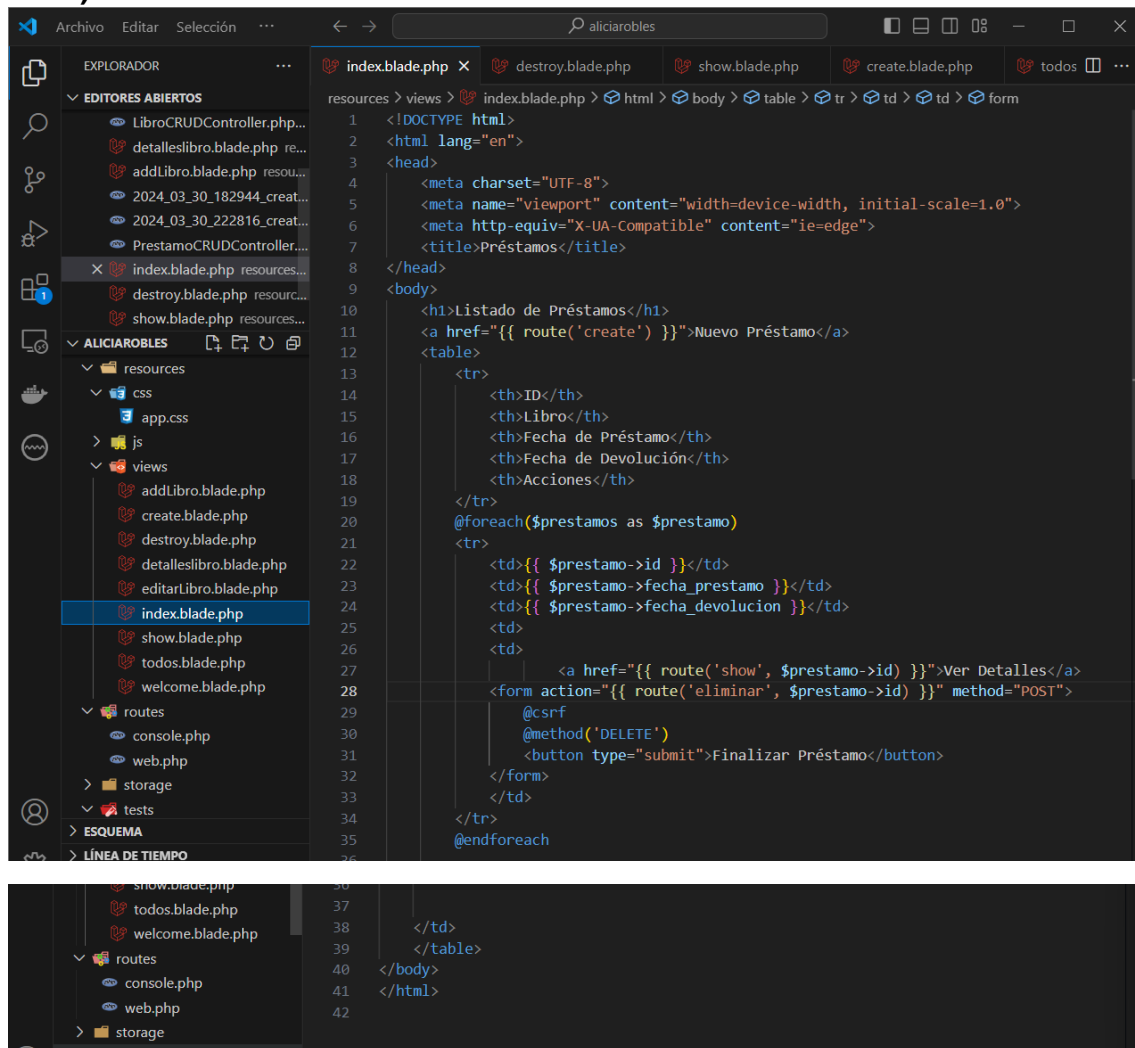


```
55 // Mostrar los detalles de un préstamo específico
56 public function show($id)
57 {
58     $prestamo = Prestamo::findOrFail($id);
59     return view('show', compact('prestamo'));
60 }
61
62 // Mostrar el formulario para editar un préstamo
63 public function edit($id)
64 {
65     $prestamo = Prestamo::findOrFail($id);
66     return view('edit', compact('prestamo'));
67 }
68
69 // Actualizar un préstamo en la base de datos
70 public function update(Request $request, $id)
71 {
72     // Validación de los datos de entrada si es necesario
73
74     $prestamo = Prestamo::findOrFail($id);
75     $prestamo->update($request->all());
76
77     return redirect('/prestamos')->with('success', 'Préstamo actualizado correctamente.');
```

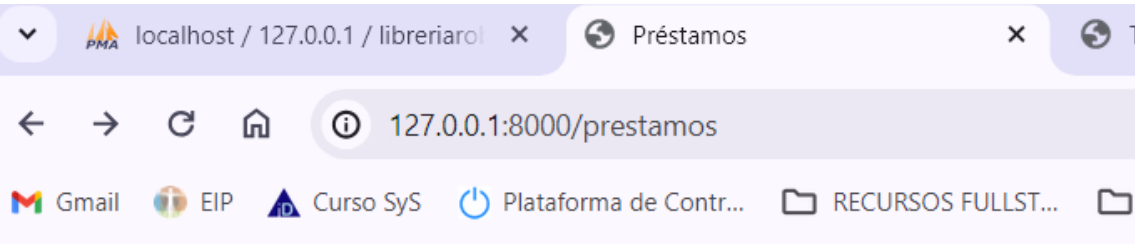


**6. Vistas de Préstamos: Crea vistas para mostrar todos los préstamos, ver detalles de un préstamo, añadir un nuevo préstamo y finalizar un préstamo existente. (El usuario sabremos cual es en la siguiente practica)**

### A) MOSTRAR TODOS LOS PRÉSTAMOS



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 <meta http-equiv="X-UA-Compatible" content="ie=edge">
7 <title>Préstamos</title>
8 </head>
9 <body>
10 <h1>Listado de Préstamos</h1>
11 <a href="{{ route('create') }}">Nuevo Préstamo</a>
12 <table>
13 <tr>
14 <th>ID</th>
15 <th>Libro</th>
16 <th>Fecha de Préstamo</th>
17 <th>Fecha de Devolución</th>
18 <th>Acciones</th>
19 </tr>
20 @foreach($prestamos as $prestamo)
21 <tr>
22 <td>{{ $prestamo->id }}</td>
23 <td>{{ $prestamo->fecha_prestamo }}</td>
24 <td>{{ $prestamo->fecha_devolucion }}</td>
25 <td>
26 <a href="{{ route('show', $prestamo->id) }}">Ver Detalles</a>
27 <form action="{{ route('eliminar', $prestamo->id) }}" method="POST">
28 @csrf
29 @method('DELETE')
30 <button type="submit">Finalizar Préstamo</button>
31 </form>
32 </td>
33 </tr>
34 @endforeach
35 </table>
36 </body>
37 </html>
```

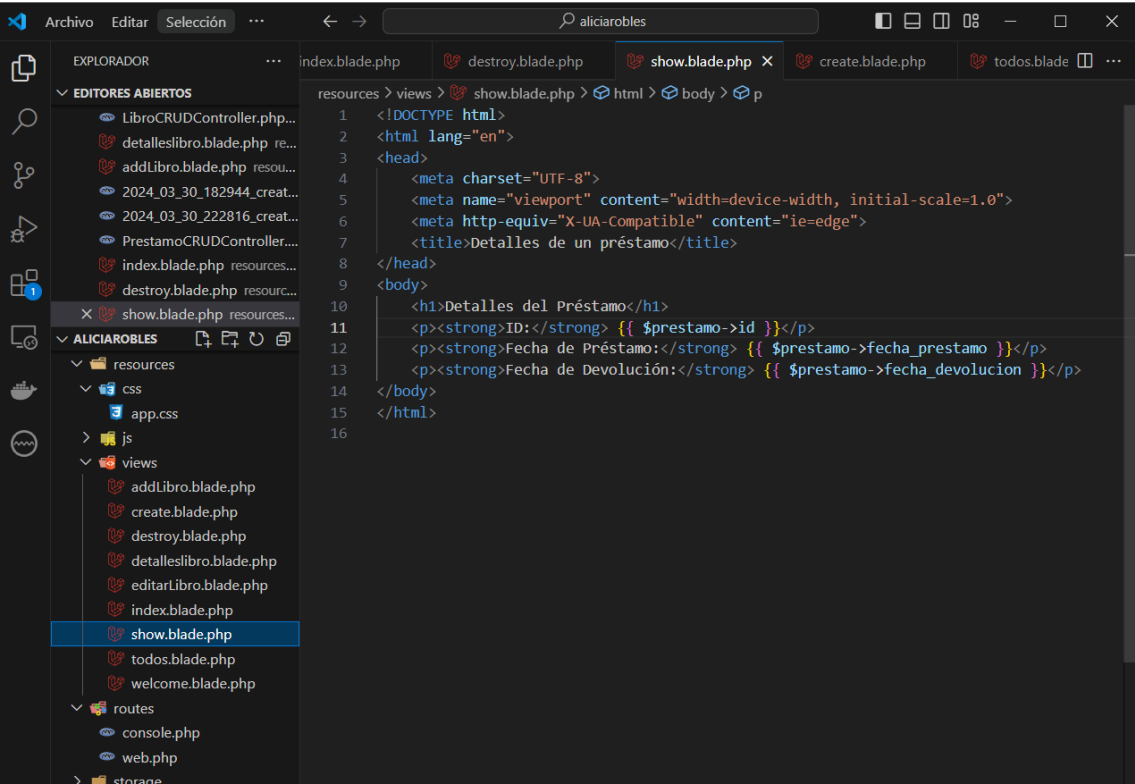


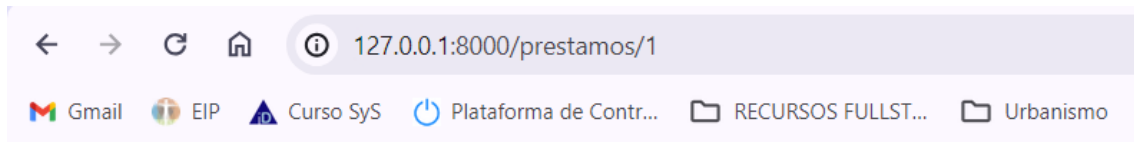
# Listado de Préstamos

## [Nuevo Préstamo](#)

ID	Libro	Fecha de Préstamo	Fecha de Devolución	Acciones
1	2024-04-05	2024-04-16		<a href="#">Ver Detalles</a> Finalizar Préstamo
2	2024-04-03	2024-05-11		<a href="#">Ver Detalles</a> Finalizar Préstamo
3	2024-04-03	2024-04-24		<a href="#">Ver Detalles</a> Finalizar Préstamo

## A) DETALLES DE UN PRÉSTAMO





## Detalles del Préstamo

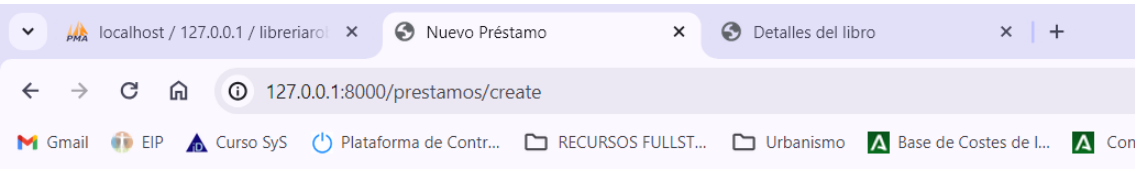
**ID:** 1

**Fecha de Préstamo:** 2024-04-05

**Fecha de Devolución:** 2024-04-16

### B) AÑADIR UN NUEVO PRÉSTAMO

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <meta http-equiv="X-UA-Compatible" content="ie=edge">
7   <title>Nuevo Préstamo</title>
8 </head>
9 <body>
10  <h1>Añadir Nuevo Préstamo</h1>
11  <form method="POST" action="{{ route('store') }}">
12    @csrf
13    <label for="libro_id">Libro:</label>
14    <select name="libro_id" id="libro_id">
15      @foreach($libros as $libro)
16        <option value="{{ $libro->id }}">{{ $libro->titulo }}</option>
17      @endforeach
18    </select><br><br>
19
20    <label for="fecha_prestamo">Fecha de Préstamo:</label>
21    <input type="date" id="fecha_prestamo" name="fecha_prestamo"><br><br>
22
23    <label for="fecha_devolucion">Fecha de Devolución:</label>
24    <input type="date" id="fecha_devolucion" name="fecha_devolucion"><br><br>
25
26    <button type="submit">Guardar Préstamo</button>
27  </form>
28 </body>
29 </html>
30
```



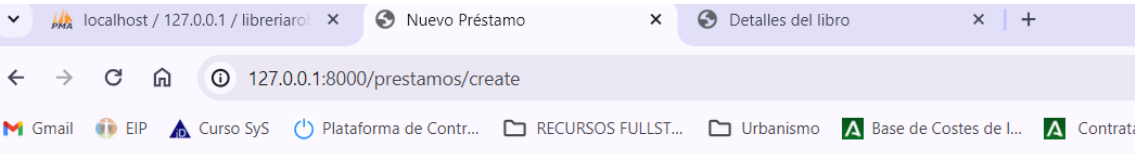
## Añadir Nuevo Préstamo

Libro: La mano de Fátima ▾

Fecha de Préstamo:

Fecha de Devolución:

Guardar Préstamo



## Añadir Nuevo Préstamo

Libro: La mano de Fátima ▾

Fecha de Préstamo:

Fecha de Devolución:

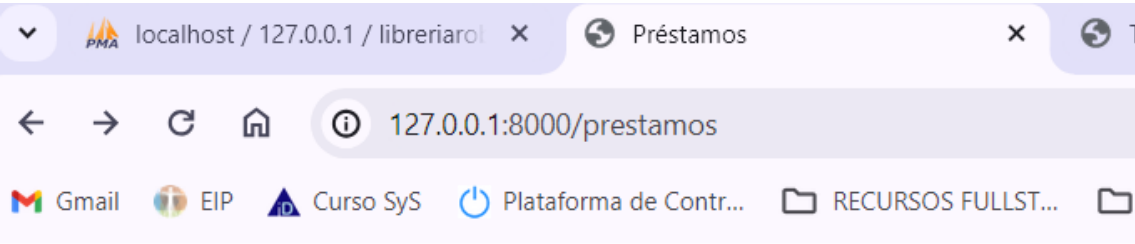
Guardar Préstamo



### C) FINALIZAR UN PRÉSTAMO EXISTENTE

He puesto un botón en la vista de Mostrar todos los préstamos que elimina el préstamo.

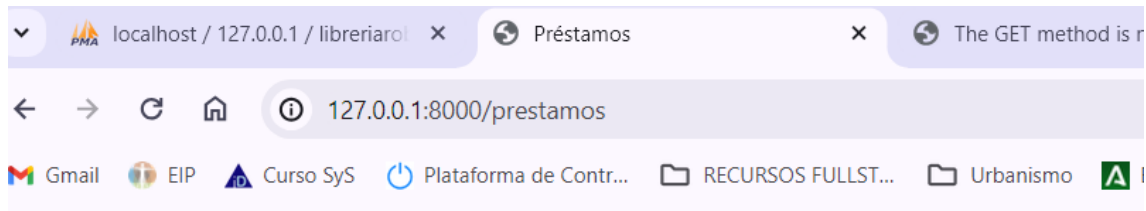
```
<a href="{{ route('show', $prestamo->id) }}">Ver Detalles</a>
<form action="{{ route('eliminar', $prestamo->id) }}" method="POST">
    @csrf
    @method('DELETE')
    <button type="submit">Finalizar Préstamo</button>
</form>
</td>
```



# Listado de Préstamos

[Nuevo Préstamo](#)

ID	Libro	Fecha de Préstamo	Fecha de Devolución	Acciones
1	2024-04-05	2024-04-16		<a href="#">Ver Detalles</a> Finalizar Préstamo
2	2024-04-03	2024-05-11		<a href="#">Ver Detalles</a> Finalizar Préstamo
3	2024-04-03	2024-04-24		<a href="#">Ver Detalles</a> Finalizar Préstamo



## Listado de Préstamos

[Nuevo Préstamo](#)

ID	Libro	Fecha de Préstamo	Fecha de Devolución	Acciones
2	2024-04-03	2024-05-11		<a href="#">Ver Detalles</a> <button>Finalizar Préstamo</button>
3	2024-04-03	2024-04-24		<a href="#">Ver Detalles</a> <button>Finalizar Préstamo</button>

## 7. Rutas: Añade rutas en tu archivo web.php para cada acción en los controladores de libros y préstamos.

Estas son las rutas utilizadas para libros:

```
Route::get('/addLibro', [LibroCRUDController::class, 'mostrarFormularioAdd']);
Route::post('/addLibroPost', [LibroCRUDController::class, 'AddLibro'])->name('AddLibro');
Route::get('/libros', [LibroCRUDController::class, 'todos']);
Route::get('/libros/{id}', [LibroCRUDController::class, 'detalleslibro']);
Route::get('/libros/{id}/editar', [LibroCRUDController::class, 'editarLibro']);
Route::put('/libros/{id}', [LibroCRUDController::class, 'update'])->name('update');
```

Y estas son las rutas utilizadas para préstamos:

```
Route::get('/prestamos', [PrestamoCRUDController::class, 'index'])->name('index');
Route::get('/prestamos/create', [PrestamoCRUDController::class, 'create'])->name('create');
Route::post('/prestamos', [PrestamoCRUDController::class, 'store'])->name('store');
Route::get('/prestamos/{id}', [PrestamoCRUDController::class, 'show'])->name('show');
Route::delete('/borrarprestamos/{id}', [PrestamoCRUDController::class, 'eliminar'])->name('eliminar');
```