**MyApi**

Un projecto para crear una api basica y entender su funcionamiennto, tambien se usa Postman para hacer el request hacia la api.

* Primero creamos un nuevo projecto con el siguiente comando:

**>>> composer create-project laravel/laravel MyApi**

* Luego debemos configurar las variables de la base de datos en el archivo **‘.env’** que se encuentra en la raiz del directorio de nuestro proyecto, se modifica tal como se describe a continuacion:

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE=MyApiDB

DB\_USERNAME=root

DB\_PASSWORD=root

Luego usaremos el siguiente comando para crear un archivo de migracion, las migraciones son una forma de interactuar con las bases de datos, permitiendo crear o modificar.

>>> php artisan make:migration create\_students\_table

En este caso específico, **create\_students\_table** es el nombre de la migración(un archivo) que estamos creando. Al ejecutar este comando, Laravel generará un nuevo archivo en el directorio ***database/migrations*** con un nombre similar a ***timestamp\_create\_students\_table.php***, donde timestamp es la fecha y hora actuales.

El comando php artisan make:migration en Laravel se utiliza para generar un nuevo archivo de migración. Las migraciones en Laravel son una forma de interactuar con la base de datos, permitiéndote crear o modificar tablas de manera programática.

En tu caso específico, create\_students\_table es el nombre de la migración que estás creando. Al ejecutar este comando, Laravel generará un nuevo archivo en el directorio database/migrations con un nombre similar a timestamp\_create\_students\_table.php, donde timestamp es la fecha y hora actuales.

**¿Qué hace este comando y cómo funciona?**

* *Creación de un Archivo de Migración:*

Laravel crea un nuevo archivo de migración con un esquema básico dentro del directorio ***database/migrations***. Este archivo contendrá una clase de migración con métodos para definir las operaciones que se desea realizar en la base de datos.

* *Definición del Esquema de la Tabla:*

Dentro de la clase de migración, puedes utilizar varios métodos proporcionados por Laravel para definir el esquema de la tabla. En este caso, es probable que usemos el método ***up()***, que define las operaciones que se realizarán al ejecutar la migración hacia adelante.

***Metodo up()***

Por ejemplo:

public function up()

{

Schema::create('students', function (Blueprint $table) {

$table->id();

$table->string('name');

$table->string('email')->unique();

$table->timestamps();

});

}

En este ejemplo con el metodo **up()** se crea una tabla llamada “students” con las siguientes columnas:

* + - id:: Identificador
    - name::Nombre::String
    - Email::String::Unique()
    - timestamps::Marcas de tiempo

***Metodo down()***

Por ejemplo:

public function down()

{

Schema::dropIfExists('students');

}

En este ejemplo con el metodo **down()** se elimina la tabla “students” si se decide revertir la migracion.

Haciendo la migracion de nuestro archivo ‘**create\_students\_table.php**’, el contenido de tal archivo sera asi:

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

return new class extends Migration

{

/\*\*

\* Run the migrations.

\*/

public function up(): void

{

Schema::create('students', function (Blueprint $table) {

$table->id();

$table->string('name');

$table->string('course');

$table->string('email');

$table->string('phone');

$table->timestamps();

});

}

/\*\*

\* Reverse the migrations.

\*/

public function down(): void

{

Schema::dropIfExists('students');

}

};

Luego para hacer la migracion hacia la base de datos, usaremos el siguiente comando en nuestra terminal:

>>> php artisan migrate

***Creacion del archivo de Modelo Student\_Model***

Los modelos en Laravel nos ayudan a Interactuar con la base de datos, permitiendonos realizar operaciones como consultas y actualizaciones de manera orientada a objetos.

En nuestro caso vamos a crear un archivo de modelo de nombre ***“Student\_Model”***  con el siguiente comando en nuestra terminal:

>>> php artisan make:model Student\_Model

Al ejecutar este comando, Laravel creara un archivo de modelo en el directorio ***‘app/Models’***  con el nombre ***‘Student\_Model’*** y con el siguuiente contenido:

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class Student\_Model extends Model

{

use HasFactory;

}

Tendremos que modificarlo un poco para que quede de la siguiente manera:

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class Student\_Model extends Model

{

use HasFactory;

protected $table = 'students';

protected $fillable = [

'name',

'course',

'email',

'phone'

];

}

Este código define un ***modelo de Eloquent llamado Student\_Model*** que está asociado con la ***tabla 'students'*** en la base de datos. La clase incluye un ***trait HasFactory*** y permite la asignación masiva de los campos ***'name', 'course', 'email' y 'phone'.***

***Creando controlador ‘StudentController***

Un controlador en Laravel actua como un intermediario entre las solicitudes del cliente y la lógica de la aplicación, ayudadno a estructurar y organizar el código de manera eficiente, una descripcion general de sus objetivos son:

**Manejar solicitudes Http**: El controlador se encarga de manejar las solicitudes HTTP entrantes, como por ejemplo GET, POST, PUT y DELETE.

**Interactuar con Modelos y Vistas**: Los controladores interactúan con los modelos para recuperar o almacenar datos en la base de datos y con las vistas para presentar esos datos al usuario. Siguen el patron ***Modelo-Vista-Controlador,***  donde los modelos representan los datos, las vistas representan la interfaz del usuario y el Controlador maneja la logica de la apliacion.

**Enrutamiento:** Los controladores estan vinculados a rutas específicas en el enrutador, cuando una solicitud llega a una ruta espec’ifica, el enrutador llama al controlador asociado para manejar esa solicitud.

**Gestionar el flujo de la aplicacion:** Los controladores ayudan a gestionar el flujo de la aplicacion al decidir que respuesta enviar al cliente en funcion de la solicitud recibida y los datos disponibles.

**Reutilizacion de Código:** Funcionalidades comunes pueden encapsularse en métodos del controlador y ser llamadas desde diferentes rutas o acciones.

**Separación de resposabilidades:** Ayudan en la separación de responsabilidades al dividir la lógica de la aplicacion en diferentes capas (Modelo-Vista-Controlador). Esto facilita el mantenimiento y la escalabilidad de la aplicación.

**Manejo del Middleware:** Los controladores pueden interactuar con middleware, que son capas intermedias que pueden realizar tareas antes o despues de que se ejecute la lógica del conrolador. Esto es útil para tareas como la autenticación, la autorización, el registro de solicitudes, etc.

En concreto se puede decir que, un controlador en Laravel actúa como un intermediario entre las solicitudes del cliente y la lógica de la aplicación, ayudando a estructurar y organizar el código de manera eficiente.

Creando un Controlador en el directorio ***‘Http/Controllers/api’*** con el nombre de archivo ‘***StudentController’*** con el siguiente comando en nuestra terminal:

>>> php artisan make:controller api/StudentController

Luego de ejecutar el comando, se creará un archivo de nombre ‘***StudentController’*** en el directorio ‘***App/Http/Controllers/api’***  y debemos agregar algunas lineas de código para que quede de la siguiente manera:

<?php

namespace App\Http\Controllers\api;

// Importa la clase 'Student\_Model' desde el namespace 'App/Models'

// Esta clase representa el modelo Eloquent asociada a la tabla de de 'students' de la DB

use App\Models\Student\_Model;

// Clase base para controladores en Laravel

use App\Http\Controllers\Controller;

// Clase que proporciona metodos para interactuar con las solicitudes HTTP

use Illuminate\Http\Request;

// Se define la clase StudentController que extiende a la clase 'Controller'

class StudentController extends Controller

{

//Se define un metodo llamado 'index', este metodo se llama cuando se realiza una

// solicitud HTTP GET a la ruta asociada al controlador

public function index()

{

//Se usa el modelo 'Student\_Model' para recuperar todos los registros de la tabla 'students'

// y los asigna a la variable $students

$students = Student\_Model::all();

// Verifica si hay registros en la variable $students

if($students->count() > 0){

// Devuelve un request json con codigo de estado 200(success) y la lista de registros

// de la tabla 'students'

return response()->json([

'status' => 200,

'students' => $students

], 200);

}else{

//devuelve un request json con un codigo de estado 404(Not found)

// y un mensaje 'No records found'

return response()->json([

'status' => 404,

'status\_message' => 'No records found'

], 404);

}

}

}

Y tambien tenemos que agregar la siguiente linea de código a nuestro archivo ***‘api.php’***  que se encuentra en el directorio ***‘routes’*** :

Route::get('students', [StudentController::class, 'index']);

Cuando se accede a ***localhost:8000/api/students*** Laravel ejecutara el método ***‘index’*** de la clase ***‘StudentController’*** que responde a las solicitudes **GET.**

***Implementando la clase para el tratamiento de solicitudes POST***

Para esto debemos