Proyecto de Laravel

1. Instalacion de Composer.

Composer es una herramienta para la administración de dependencias en PHP. Este permite declarar las librerias de las cuales tu proyecto depende y este administrara (instalara y actualizara) estas por ti.

Para instalar Composer Ir al sitio web <https://getcomposer.org/download/> y bajar el archivo  [Composer-Setup.exe](https://getcomposer.org/Composer-Setup.exe" \o "Click here to download Composer-Setup installer for Windows) el proceso de instalación es muy sencillo.

1. Crear el proyecto.

Abrir un CMD y situarse a la carpeta donde se colocan los proyectos de tu servidor de PHP , por ejemplo: para un servidor un servidor XAMPP, C:\xampp\htdocs y ejecutar el comando,

C:\xampp\htdocs> composer create-project laravel/laravel [nombre del proyecto] [“#version”]

[nombre del proyecto] = crud-laravel

[“#version”]=”10.x”

El proceso de instalación depende de las características de la computadora. Si se presenta un error que involucra a archivos con extension .zip entonces debes activar esta opción en las opciones de configuración del servidor (php.init)

1. Crear y conectar la Base de Datos(BD).

Activo mi servidor Xampp y creamos una BD en el servidor Xampp con PHPMyAdmin. La base de datos la voy a denominar crud\_laravel. Ahora conecto mi BD: crud\_laravel a mi proyecto. Ahora con el CMD nos situamos en la carpeta creada para el proyecto, C:\xampp\htdocs\crud-laravel. Abro el archivo **.env** y modifico la seccion del archivo, para conectar la BD necesito los datos que se muestran,

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE=crud\_laravel

DB\_USERNAME=root

DB\_PASSWORD=root

Para verificar la correcta conexion de la BD con nuestro proyecto ejecuto una migración,

C:\xampp\htdocs\www\crud-laravel > php artisan migrate

Si la migración se realiza con éxito se pueden ver los mensajes de las tablas que crea laravel en la BD conectada. Esto también se puede ver con el PHPMyAdmin del servidor Xampp. Este proyecto consiste en una lista de tareas las cuales vamos a gestionar. Para esto necesitamos una tabla para guardar las tareas. La tabla la denominaremos tasks\_table y la creamos con,

C:\xampp\htdocs\www\crud-laravel> php artisan make:migration create\_tasks\_table

Obteniendo el correspondiente mensaje

I INFO Migration [C:\xampp\htdocs\www\crud-laravel\database\migrations/2023\_09\_04\_15184 \_create\_tasks\_table.php] created successfully.

PS C:\xampp\htdocs\www\crud-laravel>

Abrimos el código del ORM correspondiente a la tabla en el fichero

**crud-laravel/database/migrations/2023\_09\_04\_151840\_create\_tasks\_table.php**

Que como podemos observar es el mismo que aparece en la ventana de comando después de la migración. En el fichero debemos agregar el código correspondiente a la configuración de los campos de nuestra tabla,

public function up(): void

    {

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

            $table->id();

            //codigo para la configuracion de los campos de la tabla

            $table->string('title');

            $table->text('description');

            $table->dateTime('due\_date')->nullable();

            $table->enum('status',['Pendiente','En Progreso', 'Completada'])->nullable();

            //-------------------------------

            $table->timestamps();

        });

    }

Ahora hay que guardar los cambios y ejecutar la migración para que los cambios sean efectivos, ejecutamos el comando correspondiente,

PS C:\xampp\htdocs\www\crud-laravel> php artisan migrate

INFO Running migrations.

2023\_09\_04\_151840\_create\_tasks\_table ................................................................................................... 11ms DONE

PS C:\xampp\htdocs\www\crud-laravel>

Nota: si ejecutamos una migracion y deseamos deshacerla debemos ejecutar el comando: **php artisan migrate: rollback**

Ahora Creamos el controlador y el modelo.

**Creando controlador.**

Ejecutamos,

C:\xampp\htdocs\www\crud-laravel > php artisan migrate:controller TaskController --resource --model=task

A App\Models\task model does not exist. Do you want to generate it? (yes/no) [yes]

❯ yes

INFO Model [C:\xampp\htdocs\www\crud-laravel\app/Models/task.php] created successfully.

INFO Controller [C:\xampp\htdocs\www\crud-laravel\app/Http/Controllers/Task1Controller.php] created successfully.

PS C:\xampp\htdocs\www\crud-laravel>

**Crear layouts y Views.**

Ahora nos dirigimos a la carpeta **resources/views** y creamos una carpeta **Layouts** y dentro de esta un fichero **base.blade.php** copiamos el codigo

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-GLhlTQ8iRABdZLl6O3oVMWSktQOp6b7In1Zl3/Jr59b6EGGoI1aFkw7cmDA6j6gD" crossorigin="anonymous">

<title>CRUD laravel 10</title>

</head>

<body class="bg-dark text-white">

<div class="container">

@yield('content')

</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle.min.js" integrity="sha384-w76AqPfDkMBDXo30jS1Sgez6pr3x5MlQ1ZAGC+nuZB+EYdgRZgiwxhTBTkF7CXvN" crossorigin="anonymous"></script>

</body>

</html>

A continuación creamos otro fichero **index.blade.php** en **resource/views** y copiamos el codigo,

@extends('layouts.base')

@section('content')

<div class="row">

<div class="col-12">

<div>

<h2 class="text-white">CRUD de Tareas</h2>

</div>

<div>

<a href="" class="btn btn-primary">Crear tarea</a>

</div>

</div>

<div class="col-12 mt-4">

<table class="table table-bordered text-white">

<tr class="text-secondary">

<th>Tarea</th>

<th>Descripción</th>

<th>Fecha</th>

<th>Estado</th>

<th>Acción</th>

</tr>

<tr>

<td class="fw-bold">Estudiar Laravel</td>

<td>Ver video: tu primer CRUD con laravel 10 en el canal de YouDevs</td>

<td>

31/03/23

</td>

<td>

<span class="badge bg-warning fs-6">Pendiente</span>

</td>

<td>

<a href="" class="btn btn-warning">Editar</a>

<form action="" method="post" class="d-inline">

<button type="submit" class="btn btn-danger">Eliminar</button>

</form>

</td>

</tr>

</table>

</div>

</div>

@endsection

**Definición de las rutas**

Para definir las rutas nos dirigimos al fichero crud-laravel/routes/web.php

<?php

use Illuminate\Support\Facades\Route;

use App\Http\Controllers\Task1Controller; //Agregamos el controlador

/\*

|--------------------------------------------------------------------------

| Web Routes

|--------------------------------------------------------------------------

|

| Here is where you can register web routes for your application. These

| routes are loaded by the RouteServiceProvider and all of them will

| be assigned to the "web" middleware group. Make something great!

|

\*/

Route::get('/', function () {

    return view('welcome');

});

Route::resource('task', Task1Controller::class); // Agregamos la ruta correspondiente al controlador

Aqui hemos agregado el fichero correspondiente al controlador(comentario://Agregamos el controlador ),

use App\Http\Controllers\Task1Controller;

y ademas agregamos tambien

Route::resource('task', Task1Controller::class);

Observemos aqui que el parametro 'task' es el que anteriormente introducimos en el comando php artisan make:controller TaskController --resource --model=task

en el --model=task podemos ver el nombre de la ruta.

Para ver todas las rutas disponibles usamos el comando,

php artisan route:list

**Inicio del servidor**

Iniciamos el servidor con,

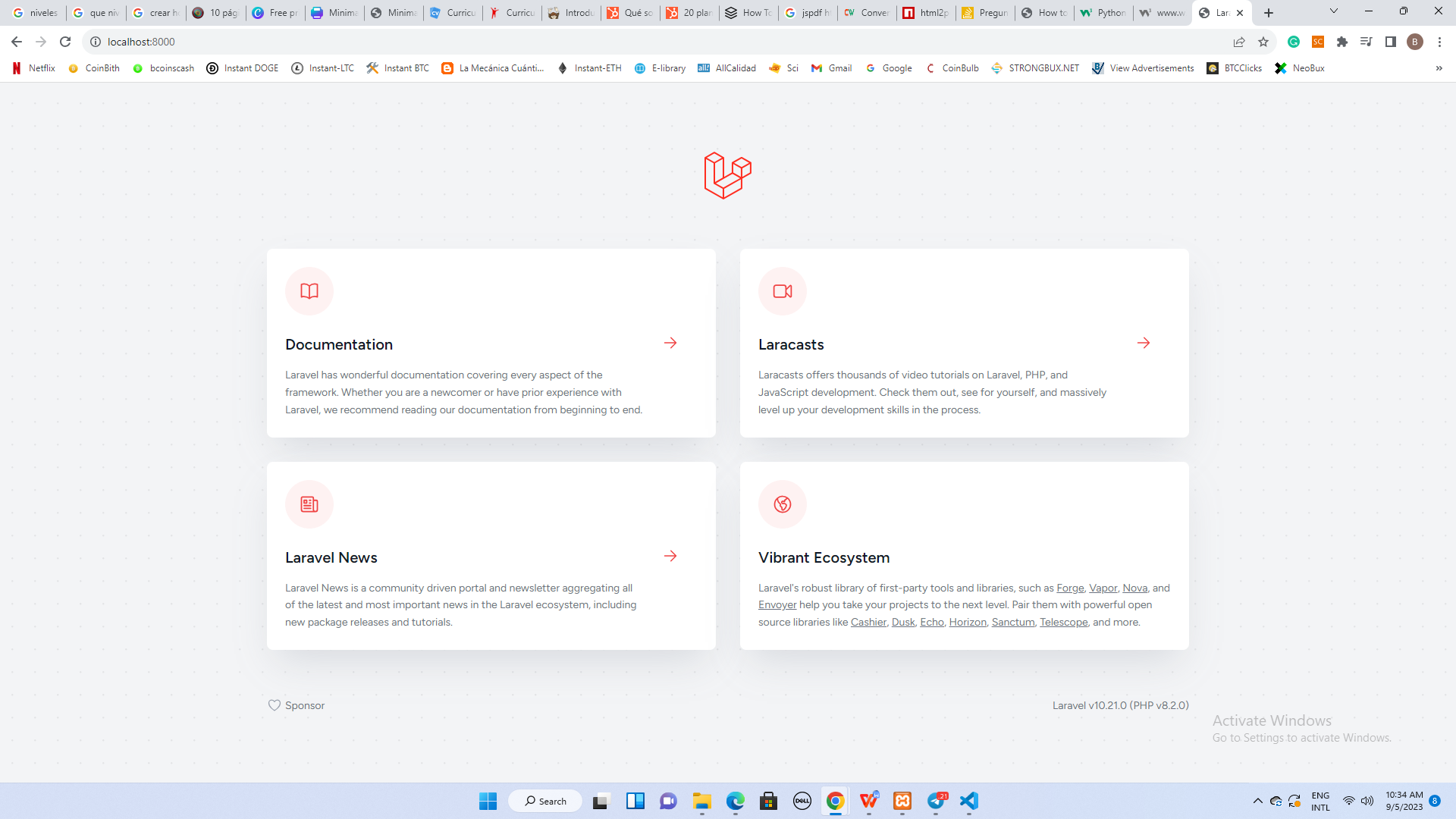
C:\xampp\htdocs\www\crud-laravel> php artisan serve

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

Press Ctrl+C to stop the server

Ahora nos dirigimos al browser y en la barra de direcciones escribimos: localhost:8000

Asi vidualizamos la pagina principal de nuestro sitio construido con laravel



**Visualización de las rutas.**

Para visualizar las rutas nos dirigimos al archivo del controlador y modificamos la funcion index( ),

<?php

namespace App\Http\Controllers;

use App\Models\task;

use Illuminate\Http\Request;

class Task1Controller extends Controller

{

    /\*\*

     \* Display a listing of the resource.

     \*/

    public function index()

    {

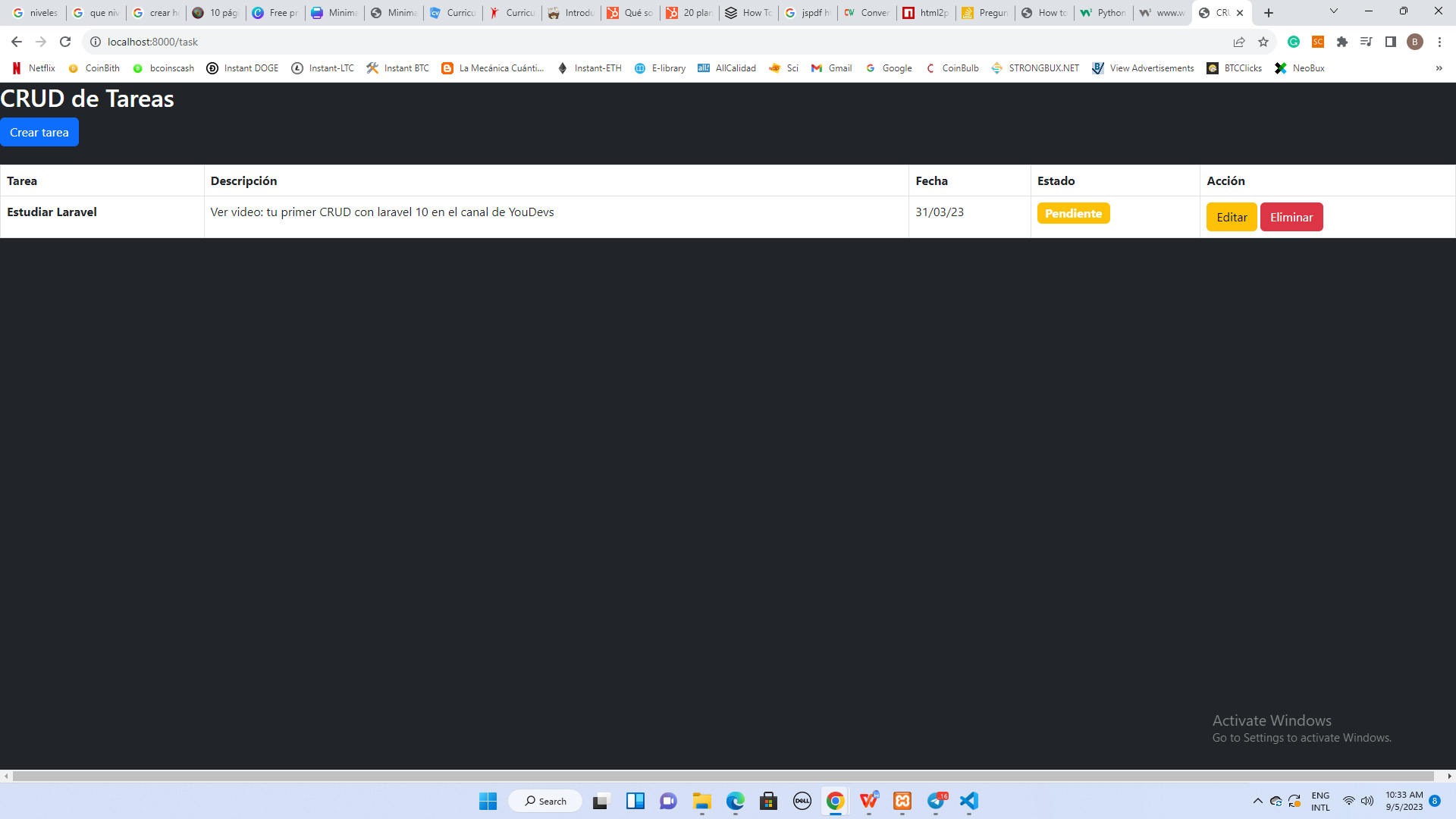
        return view('index');//Agregamos la vista correspondiente

    }

con la linea de código: return view(‘index’); esto nos permite mostrar la vista que esta en el archivo:

crud-laravel/resource/views/layouts/index.blade.php

Guardamos los cambios y a continuación podemos visualizar la ruta correspondiente con: localhost:8000/task



**Desarrollando las funcionalidades del CRUD.**