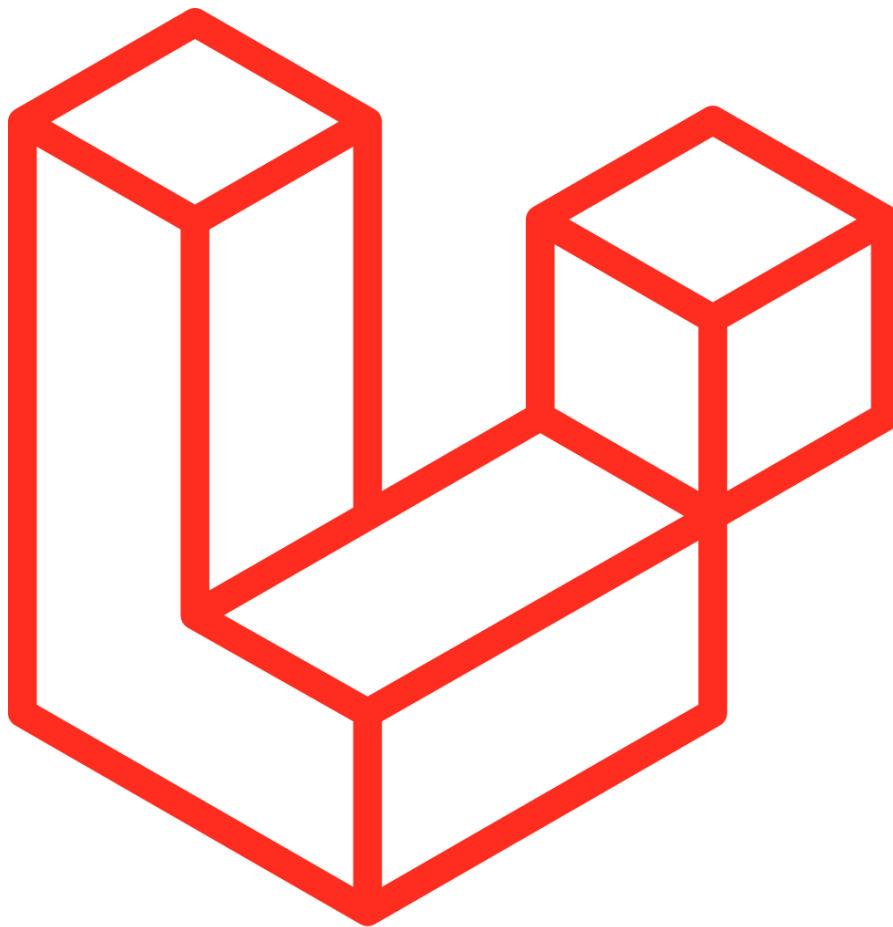


# PEC 4: LARAVEL



**Desarrollo back-end con LARAVEL**

María Cañas Encinas

## **ÍNDICE**

- 1. Instalación de laravel**
- 2. Sistema de autenticación**
- 3. Creación de las migraciones y modelos necesarios. Uso de Tinker**
- 4. Generar contenidos ficticios**
- 5. Implementar el frontend de la aplicación**
- 6. API**
- 7. Publicación en internet**
- 8. Experiencia con los entornos**

## 1. INSTALACIÓN DE LARAVEL

Para la instalación de Laravel, previamente se tiene que tener instalado en el sistema Composer, por lo que se tiene que seguir las instrucciones que encontramos en su página web <https://getcomposer.org/download/>

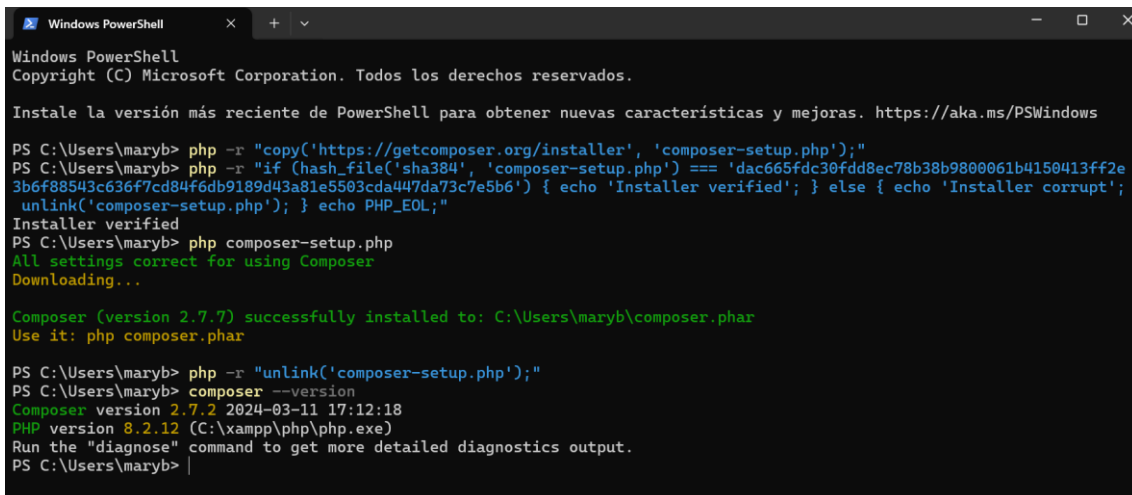
Las siguientes líneas es el código que se tiene que introducir en nuestra terminal

```
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
```

```
php -r "if (hash_file('sha384', 'composer-setup.php') ===  
'dac665fdc30fdd8ec78b38b9800061b4150413ff2e3b6f88543c636f7cd84f6  
db9189d43a81e5503cda447da73c7e5b6') { echo 'Installer verified'; } else  
{ echo 'Installer corrupt'; unlink('composer-setup.php'); } echo PHP_EOL;"
```

```
php composer-setup.php
```

```
php -r "unlink('composer-setup.php');"
```



```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

PS C:\Users\maryb> php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
PS C:\Users\maryb> php -r "if (hash_file('sha384', 'composer-setup.php') === 'dac665fdc30fdd8ec78b38b9800061b4150413ff2e3b6f88543c636f7cd84f6db9189d43a81e5503cda447da73c7e5b6') { echo 'Installer verified'; } else { echo 'Installer corrupt'; unlink('composer-setup.php'); } echo PHP_EOL;"
Installer verified
PS C:\Users\maryb> php composer-setup.php
All settings correct for using Composer
Downloading...

Composer (version 2.7.7) successfully installed to: C:\Users\maryb\composer.phar
Use it: php composer.phar

PS C:\Users\maryb> php -r "unlink('composer-setup.php');"
PS C:\Users\maryb> composer --version
Composer version 2.7.2 2024-03-11 17:12:18
PHP version 8.2.12 (C:\xampp\php\php.exe)
Run the "diagnose" command to get more detailed diagnostics output.
PS C:\Users\maryb>
```

## Problemas en la instalación de laravel

Me ha salido el problema de que no tenía habilitada la extensión zip, por lo que he tenido que modificar el archivo php.ini de xampp, para ello había que quitar el punto y coma ; que aparece al inicio de la línea de extensión=zip

Posteriormente se crea el proyecto Laravel con el siguiente comando

```
C:\Windows\System32>composer create-project --prefer-dist laravel/laravel laravel11 "11.*"
Creating a "laravel/laravel" project at "./laravel11"
Installing laravel/laravel (v11.1.1)
```

Y se puede ver como se ha creado de forma correcta

```
57 package suggestions were added by new dependencies, use `composer suggest` to see details.
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi

 INFO  Discovering packages.

laravel/sail ..... DONE
laravel/tinker ..... DONE
nesbot/carbon ..... DONE
nunomaduro/collision ..... DONE
nunomaduro/termwind ..... DONE

79 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
> @php artisan vendor:publish --tag=laravel-assets --ansi --force

 INFO  No publishable resources for tag [laravel-assets].

No security vulnerability advisories found.
> @php artisan key:generate --ansi

 INFO  Application key set successfully.

> @php -r "file_exists('database/database.sqlite') || touch('database/database.sqlite');"
> @php artisan migrate --graceful --ansi

 INFO  Preparing database.

Creating migration table ..... 13.50ms DONE

 INFO  Running migrations.

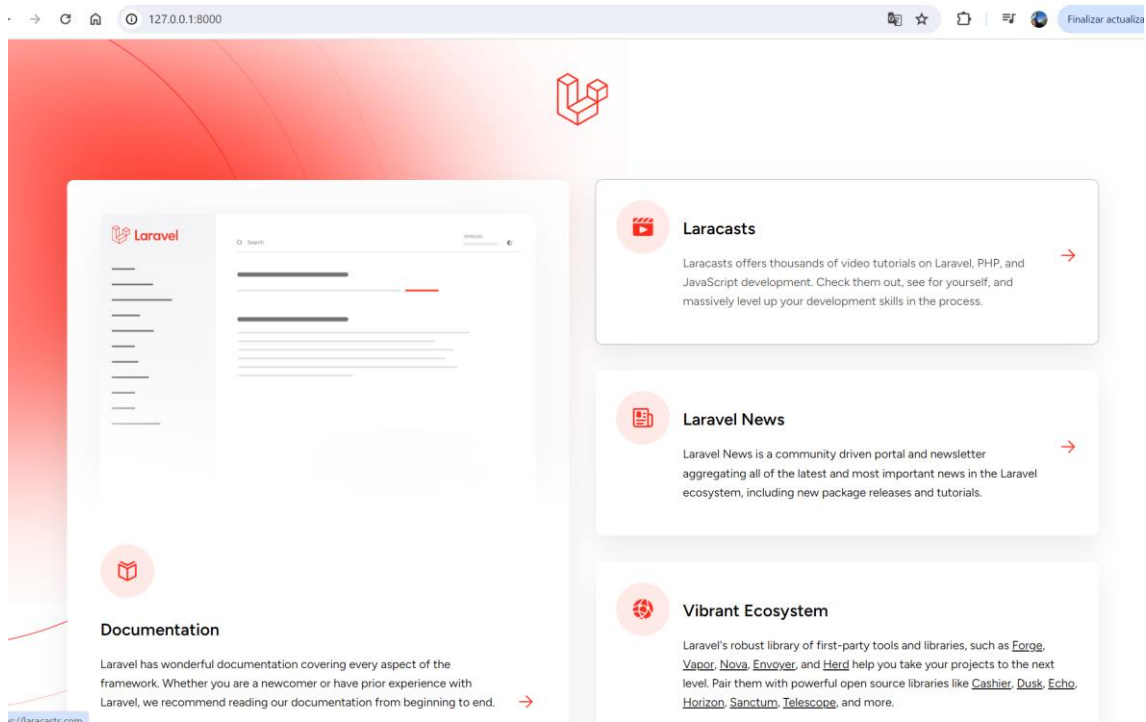
0001_01_01_000000_create_users_table ..... 34.08ms DONE
0001_01_01_000001_create_cache_table ..... 10.18ms DONE
0001_01_01_000002_create_jobs_table ..... 25.21ms DONE
```

Después de la instalación se accede al directorio laravel11 con **cd laravel11** y se inicia el servidor de desarrollo de laravel con el siguiente comando **php artisan serve** y se va a iniciar el servidor en la dirección <http://127.0.0.1:8000>

```
C:\Windows\System32\laravel11>php artisan serve

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

Press Ctrl+C to stop the server
```



## Comparación del proceso de instalación de Laravel con Drupal

En cuanto a la instalación de uno y otro, en el de laravel es de forma directa a través de Composer, y no nos toca mover archivos o configurar un servidor web, mientras que en Drupal tenemos que mover archivos y configurar de forma manual la base de datos. En cuanto a las dependencias a extensiones Laravel necesita varias, pero estas van a ser manejadas de forma automáticamente por Composer, mientras que en Drupal también necesita de extensiones pero la configuración de la base de datos no es automática sino que se tiene que hacer de forma manual. En el proceso de instalación en Laravel se hace todo desde la línea de comandos mientras que el de Drupal también se tiene que interactuar con el navegador para finalizar su configuración. En conclusión, mientras que en Laravel todo el proceso de instalación es de forma automatizada y rápida en Drupal es más de configuración manual.

## 2. SISTEMA DE AUTENTIFICACIÓN

Antes de realizar el sistema de autenticación nos tenemos que asegurar que tenemos Breeze instalado para ello vamos a poner los siguientes comandos por consola.

**composer require laravel/breeze --dev**

**php artisan breeze:install blade**

**npm install**

## npm run dev

Estos comandos lo que van a hacer es instalarnos Laravel Breeze y van a configurar la plantilla de Blade y compilan los recursos frontEnd.

Después vamos a realizar la modificación de las migraciones. Lo primero que se nos pide es modificar la migración de usuarios, cambiando el nombre de la tabla de usuarios que está puesto en “users” por defecto a “users\_pec4”. Esto se va a realizar dentro del fichero de **create\_users\_table.php** que se encuentra dentro de la carpeta **database/migrations**.

Pasando de esto:

```
return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('email')->unique();
            $table->timestamp('email_verified_at')->nullable();
            $table->string('password');
            $table->rememberToken();
            $table->timestamps();
        });
    }
};
```

A esto:

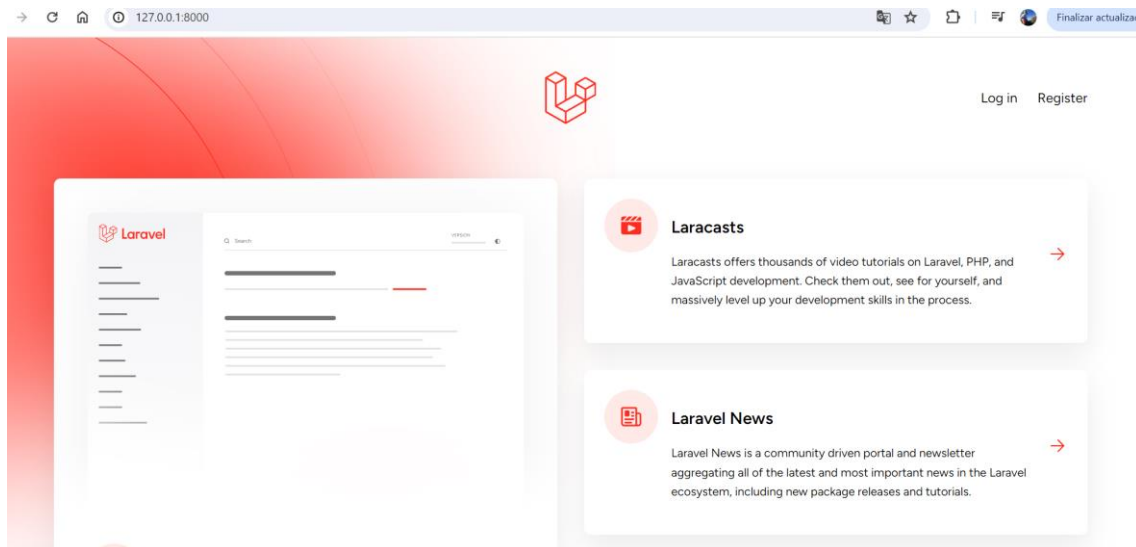
```
return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('users_pec4', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('email')->unique();
            $table->timestamp('email_verified_at')->nullable();
            $table->string('password');
            $table->rememberToken();
            $table->timestamps();
        });

        Schema::create('password reset tokens', function (Blueprint $table) {
            $table->string('token')->unique();
            $table->timestamp('created_at')->nullable();
        });
    }
};
```

Después se va a escribir por consola la siguiente línea iniciar el servidor de desarrollo php artisan serve.

```
C:\Windows\System32\laravel11>php artisan serve  
  
INFO Server running on [http://127.0.0.1:8000].  
  
Press Ctrl+C to stop the server
```

Accederemos a la dirección de <http://127.0.0.1:8000> para verificar el menú con las opciones de Log in y Register.



Para registrar a un nuevo usuario accederemos a la ruta <http://127.0.0.1:8000/register> e introducimos los datos del usuario que se nos especifican en el enunciado siendo estos

Usuario: admin

Contraseña: @dmin\_2024

Si el usuario se ha creado de forma correcta, se nos muestra esta pantalla.



## Dashboard

You're logged in!

Después vamos a cerrar sesión e iniciar sesión con el usuario creado para comprobar que podemos acceder a la cuenta de este nuevo usuario creado.

El siguiente paso que tenemos que hacer es modificar el mensaje de bienvenida, para ello vamos a acceder al fichero **dashboard.blade.php** que está situado en **resources/views/** y vamos a modificar el mensaje de bienvenida.

```
<x-app-layout>
  <x-slot name="header">
    <h2 class="font-semibold text-xl text-gray-800 leading-tight">
      | {{ __('Dashboard') }}
    </h2>
  </x-slot>

  <div class="py-12">
    <div class="max-w-7xl mx-auto sm:px-6 lg:px-8">
      <div class="bg-white overflow-hidden shadow-sm sm:rounded-lg">
        <div class="p-6 text-gray-900">
          <h1>Bienvenido/a, {{ Auth::user()->name }}!</h1>
        </div>
      </div>
    </div>
  </div>
</x-app-layout>
```

Y si accedemos a la página de Laravel con el usuario creado se nos va a mostrar ese mensaje.



## Dashboard

Bienvenido/a, admin!



### 3. CREACIÓN DE LAS MIGRACIONES Y MODELOS NECESARIOS. USO DE TINKER

Lo primero que tenemos que hacer es crear las migraciones para las entidades “**recipes**” y “**categories**” y una migración para la tabla pivot que va a relacionar ambas entidades.

Para crear las migraciones tenemos que ejecutar los siguientes comandos por consola

**php artisan make:model Recipe -m**







**php artisan make:model Category -m**

```
C:\Windows\System32>cd laravel11
C:\Windows\System32\laravel11>php artisan make:model Recipe -m
INFO Model [C:\Windows\System32\laravel11\app\Models\Recipe.php] created successfully.
INFO Migration [C:\Windows\System32\laravel11\database\Migrations\2024_06_22_132313_create_recipes_table.php] created successfully.
C:\Windows\System32\laravel11>php artisan make:model Category -m
INFO Model [C:\Windows\System32\laravel11\app\Models\Category.php] created successfully.
INFO Migration [C:\Windows\System32\laravel11\database\Migrations\2024_06_22_132319_create_categories_table.php] created successfully.
```

Y después crear la migración para la tabla pivot que va a relacionar estas dos migraciones creadas. Introduciendo por consola el comando:

**php artisan make:migration create\_recipe\_category\_table**

Con esto lo que vamos a conseguir es que se nos generen tres archivos de migración dentro de la carpeta **database/migrations**

Nombre	Fecha de modificación	tipo	Idioma
 0001_01_01_000000_create_users_table	22/06/2024 13:49	Archivo de origen PHP	2 KB
 0001_01_01_000001_create_cache_table	04/06/2024 6:28	Archivo de origen PHP	1 KB
 0001_01_01_000002_create_jobs_table	04/06/2024 6:28	Archivo de origen PHP	2 KB
 2024_06_22_132313_create_recipes_table	22/06/2024 15:23	Archivo de origen PHP	1 KB
 2024_06_22_132319_create_categories_table	22/06/2024 15:23	Archivo de origen PHP	1 KB
 2024_06_22_132724_create_recipe_category_tab...	22/06/2024 15:27	Archivo de origen PHP	1 KB

El siguiente paso, va a ser definir las migraciones dentro de cada uno de los ficheros, como en el enunciado se nos dice.

Dentro del fichero `create_recipes_table` se van a declarar los diferentes campos de la tabla:

```

    */
    public function up(): void
    {
        Schema::create('recipes', function (Blueprint $table) {
            $table->id();
            $table->string('nombre', 255);
            $table->dateTime('fecha_publicacion');
            $table->enum('nivel_dificultad', ['bajo', 'medio', 'alto']);
            $table->unsignedBigInteger('categoria');
            $table->integer('tiempo_preparacion');
            $table->text('ingredientes');
            $table->string('autor');
            $table->text('instrucciones_preparacion');
            $table->string('imagen')->nullable();
            $table->timestamps();
            $table->foreign('categoria')->references('id')->on('categories');
        });
    }
}

```

Dentro del fichero `create_categories_table` se van a declarar los diferentes valores de las categorías disponibles para las recetas:

```

    */
    public function up(): void
    {
        Schema::create('categories', function (Blueprint $table) {
            $table->id();
            $table->string('nombre');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
}

```

Dentro del fichero `create_recipe_category_table` se se va declarar el esquema para la tabla pivot que va a gestionar la relación entre recetas y categories:

```

    public function up(): void
    {
        Schema::create('recipe_category', function (Blueprint $table) {
            $table->id();
            $table->unsignedBigInteger('recipe_id');
            $table->unsignedBigInteger('category_id');
            $table->timestamps();
            $table->foreign('recipe_id')->references('id')->on('recipes')->onDelete('cascade');
            $table->foreign('category_id')->references('id')->on('categories')->onDelete('cascade');
            $table->unique(['recipe_id', 'category_id']);
        });
    }
}

```

Después de haber definido estas migraciones, se tienen que ejecutar, escribiendo el comando de **php artisan migrate** por consola para que se creen las tablas en la base de datos.

```
C:\Windows\System32\laravel11>php artisan migrate

INFO Running migrations.

2024_06_22_132313_create_recipes_table ..... 11.73ms DONE
2024_06_22_132319_create_categories_table ..... 4.48ms DONE
2024_06_22_132724_create_recipe_category_table ..... 9.55ms DONE
```

El siguiente paso es crear los modelos y relaciones entre *recipe* y *category*. Para ello accedemos al fichero correspondiente de cada modelo que está dentro de la carpeta **app/models/<nombre del Modelo.php>**

## Recipe

```
class Recipe extends Model
{
    use HasFactory;

    //Atributos, que son los campos que se pueden rellenar cuando se crea o se actualiza una recipe
    protected $fillable = ['nombre','fecha_publicacion','nivel_dificultad',
        'tiempo_preparacion','ingredientes','autor','instrucciones_preparacion',
        'imagen'];

    //Función para la relacion entre recetas y categorias

    public function categorias(){
        return $this->belongsToMany(Category::class,'category_recipe');
    }
}
```

## Category

```
class Category extends Model
{
    use HasFactory;

    //Atributos
    protected $fillable = ['nombre'];

    //Relacion con el modelo recipe
    public function recipes(): BelongsToMany{
        return $this->belongsToMany(Recipe::class, 'category_recipe');
    }
}
```

Una vez creados los modelos se va a usar Tinker para crear las dos recetas y asignar varias categorías.

Para abrir Tinker se va a escribir por consola el comando **php artisan tinker** y después se crean las 2 categorías. En mi caso he creado las categorías de mexicana e italiana.

```

C:\Windows\System32\laravel11>php artisan tinker
Psy Shell v0.12.4 (PHP 8.2.12 - cli) by Justin Hileman
> $mexicana = App\Models\Category::create(['nombre' => 'Mexicana']);
= App\Models\Category {#5069
  nombre: "Mexicana",
  updated_at: "2024-06-22 17:08:37",
  created_at: "2024-06-22 17:08:37",
  id: 1,
}

> $italiana = App\Models\Category::create(['nombre' => 'Italiana']);
= App\Models\Category {#5054
  nombre: "Italiana",
  updated_at: "2024-06-22 17:08:57",
  created_at: "2024-06-22 17:08:57",
  id: 2,
}

```

Y después se crean las dos recetas.

```

> $receta1 = App\Models\Recipe::create([
  . 'nombre' => 'Tacos de pollo',
  . 'fecha_publicacion' => now(),
  . 'nivel_dificultad' => 'medio',
  . 'tiempo_preparacion' => 45,
  . 'ingredientes' => 'Tortillas de maíz, pollo deshebrado, salsa picante, queso fresco, crema',
  . 'autor' => 'Chef Maria',
  . 'instrucciones_preparacion' => 'Cocina y desmenuza el pollo. Calienta las tortillas y rellénalas con pollo desmenu
zado. Añade salsa, guacamole, cebolla y cilantro al gusto. Sirve y disfruta de estos tacos sabrosos y fáciles de prepara
r, ideales para cualquier ocasión.',
  . ]);
= App\Models\Recipe {#6229
  nombre: "Tacos de pollo",
  fecha_publicacion: Illuminate\Support\Carbon @1719149528 {#6208
    date: 2024-06-23 13:32:08.595259 UTC (+00:00),
  },
  nivel_dificultad: "medio",
  tiempo_preparacion: 45,
  ingredientes: "Tortillas de maíz, pollo deshebrado, salsa picante, queso fresco, crema",
  autor: "Chef Maria",
  instrucciones_preparacion: "Cocina y desmenuza el pollo. Calienta las tortillas y rellénalas con pollo desmenuzado.
Añade salsa, guacamole, cebolla y cilantro al gusto. Sirve y disfruta de estos tacos sabrosos y fáciles de preparar, ide
ales para cualquier ocasión.",
  updated_at: "2024-06-23 13:32:08",
  created_at: "2024-06-23 13:32:08",
  id: 1,
}

```

```

> $receta2 = App\Models\Recipe::create([
    . 'nombre' => 'Pizza Margarita',
    . 'fecha_publicacion' => now(),
    . 'nivel_dificultad' => 'medio',
    . 'tiempo_preparacion' => 30,
    . 'ingredientes' => 'Masa para pizza, salsa de tomate, mozzarella fresca, hojas de albahaca, aceite de oliva, sal y pimienta.',
    . 'autor' => 'Chef italiano',
    . 'instrucciones_preparacion' => 'Precalienta el horno a 230C. Extiende la masa de pizza en una bandeja enharinada. Cubre la masa con salsa de tomate. Distribuye las rodajas de mozzarella sobre la salsa. Coloca las hojas de albahaca fresca sobre la pizza. Rocía un poco de aceite de oliva y sazón con sal y pimienta. Hornea durante 10-15 minutos o hasta que la pizza esté dorada y el queso se haya derretido. Retira del horno, corta en porciones y sirve caliente.'
]);
= App\Models\Recipe {#6310
    nombre: "Pizza Margarita",
    fecha_publicacion: Illuminate\Support\Carbon @1719153513 {#6289
        date: 2024-06-23 14:38:33.354555 UTC (+00:00),
    },
    nivel_dificultad: "medio",
    tiempo_preparacion: 30,
    ingredientes: "Masa para pizza, salsa de tomate, mozzarella fresca, hojas de albahaca, aceite de oliva, sal y pimienta.",
    autor: "Chef italiano",
    instrucciones_preparacion: "Precalienta el horno a 230C. Extiende la masa de pizza en una bandeja enharinada. Cubre la masa con salsa de tomate. Distribuye las rodajas de mozzarella sobre la salsa. Coloca las hojas de albahaca fresca sobre la pizza. Rocía un poco de aceite de oliva y sazón con sal y pimienta. Hornea durante 10-15 minutos o hasta que la pizza esté dorada y el queso se haya derretido. Retira del horno, corta en porciones y sirve caliente.",
    updated_at: "2024-06-23 14:38:33",
    created_at: "2024-06-23 14:38:33",
    id: 2,
}

```

Y después se asignan las categorías a las recetas que hemos creado.

```

> $receta1->categories()->sync([$mexicana->id, $italiana->id]);
= [
    "attached" => [],
    "detached" => [
        1,
        2,
    ],
    "updated" => [],
]

> $receta2->categories()->sync([$italiana->id]);
= [
    "attached" => [],
    "detached" => [
        2,
    ],
    "updated" => [],
]

```

#### 4. GENERAR CONTENIDOS FICTICIOS

Para crear recetas ficticias vamos a utilizar Factory, para ello tenemos que utilizar el comando **php artisan make:factory RecipeFactory --model=Recipe**, este comando lo que va a hacer es crear un archivo **RecipeFactory.php** dentro de la carpeta de **database/factories**

```

C:\Windows\System32\mi-proyecto-laravel> php artisan make:factory RecipeFactory --model=Recipe

INFO Factory [C:\Windows\System32\mi-proyecto-laravel\database\factories\RecipeFactory.php] created successfully.

```

Y vamos a modificar ese archivo para definir como queremos que se generen los datos ficticios de nuestras recetas, para ello se va a utilizar faker. Y después creamos el seeder que en Laravel se utiliza para llenar una base de datos de

prueba de forma automática con el comando **php artisan make:seeder RecipeSeeder** y se nos creará un fichero **recipeseeder.php** dentro de **database/seeder**, ese fichero le modificamos para que se nos creen 100 recetas de forma automática.

```
C:\Windows\System32\mi-proyecto-laravel>php artisan make:seeder RecipeSeeder  
INFO Seeder [C:\Windows\System32\mi-proyecto-laravel\database\seeders\RecipeSeeder.php] created successfully.
```

Una vez creado y modificados los ficheros ejecutamos el seeder con el siguiente comando **php artisan db:seed --class=RecipeSeeder**

```
C:\Windows\System32\mi-proyecto-laravel>php artisan db:seed  
INFO Seeding database.
```

Y para asegurarnos de que se han creado las recetas, podemos utilizar tinker y preguntar por consola el numero de recetas que tenemos en nuestra base de datos. Para ello abrimos tinker con el comando **php artisan tinker** y después poner el comando **App\Models\Recipe::count()** y nos va a mostrar 102 que son las 2 recetas que creamos al principio y las 100 que se nos han creado de forma automática.

```
C:\Windows\System32\mi-proyecto-laravel>php artisan tinker  
Psy Shell v0.12.4 (PHP 8.2.12 - cli) by Justin Hileman  
> App\Models\Recipe::count();  
= 102
```

## 5. IMPLEMENTAR EL FRONTEND DE LA APLICACIÓN

Lo primero que tenemos que hacer es crear las vistas para la página de inicio (Home) y la página que va a tener la información de la receta, para ello vamos a crear los ficheros de [home.blade.php](#) y [recipe.blade.php](#) dentro de la carpeta de **resources/views/**.

### [Home.blade.php](#)

```
<!--Listado de las 5 recetas en la página principal -->  
<h1>Listado de Recetas</h1>  
<div class="row">  
    @foreach($recipes as $recipe)  
        <div class="col-md-4">  
            <div class="card mb-4">  
                nombre }}">  
                <div class="card-body">  
                    <h5 class="card-title"><a href="{{ route('recipe.show', $recipe->id) }}">{{ $recipe->nombre }}</a></h5>  
                    <p class="card-text">Dificultad: {{ $recipe->nivel_dificultad }}</p>  
                    <p class="card-text">Categoria: {{ $recipe->categoria }}</p>  
                    <p class="card-text">Fecha Publicación: {{ $recipe->fecha_publicacion->format('d M, Y') }}</p>  
                </div>  
            </div>  
        </div>  
    @endforeach  
</div>
```

## Récipe.blade.php

```
2
3 <!DOCTYPE html>
4 <html lang="en">
5 <head>
6 |   <meta charset="UTF-8">
7 |   <title>{{ $recipe->nombre }}</title>
8 </head>
9 <body>
10 |   @include('menu')
11
12 |   <h1>{{ $recipe->nombre }}</h1>
13 |   nombre }}" class="">
14 |   <p><strong>Dificultad:</strong> {{ $recipe->nivel_dificultad }}</p>
15 |   <p><strong>Categoria:</strong> {{ $recipe->categoria }}</p>
16 |   <p><strong>Fecha Publicacion</strong> {{ $recipe->fecha_publicacion->format('d M, Y') }}
17 |
18 </body>
19
20 </html>
```

Después tenemos que crear el **recipeController.php** que será el encargado de manejar las solicitudes relacionadas con las recetas. Dentro de este archivo vamos a especificar que en la página principal se nos seleccionen 5 recetas de las cuales 2 sean las dos primeras que hemos creado con datos reales, y las otras 3 sean recetas aleatorias de las 100 que hemos creado de forma automática.

```
class RecipeController extends Controller
{
    1 reference | 0 overrides
    public function index()
    {
        // Obtener las 2 recetas que hemos creado con datos reales
        $fixedRecipes = Recipe::whereIn('id', [1, 2])->get();

        // Obtener 3 recetas aleatorias excluyendo las 2 anteriores
        //Estas van a ser 3 recetas de las que se han creado de forma elatoria
        $randomRecipes = Recipe::whereNotIn('id', [[1, 2]])->inRandomOrder()->limit(3)->get();

        $recipes = $fixedRecipes->merge($randomRecipes);

        return view('home', compact('recipes'));
    }

    1 reference | 0 overrides
    public function show($id){
        $recipe = Recipe::findOrFail($id);
        return view('recipe', compact('recipe'));
    }
}
```

Una vez que hemos creado las páginas, tenemos que añadir las rutas en nuestro archivo de **web.php** que está dentro de la carpeta **routes**

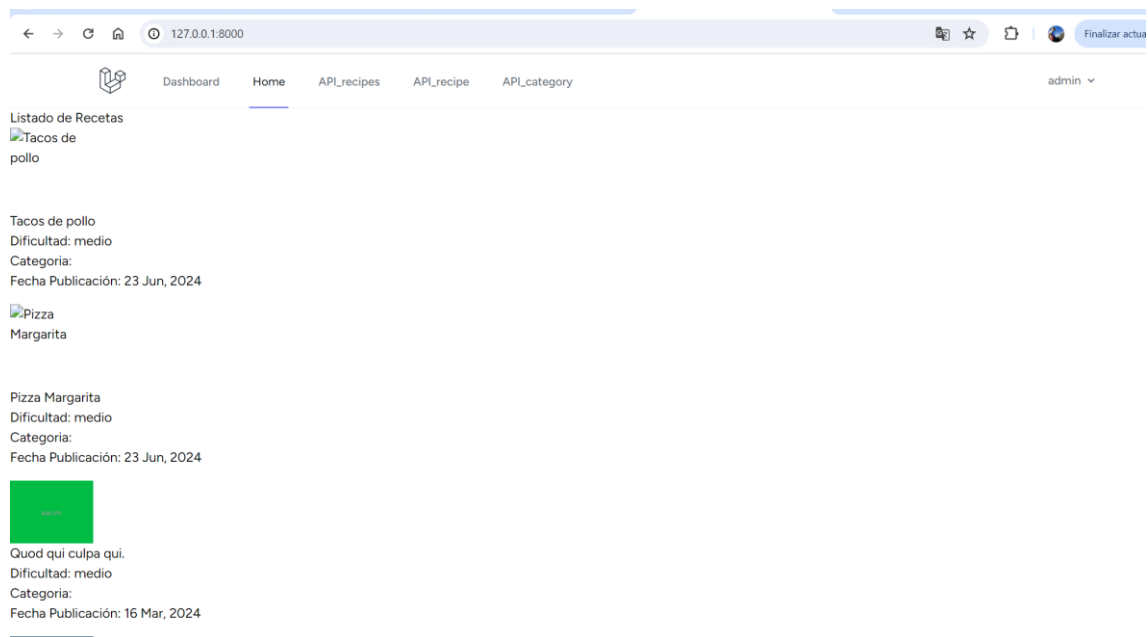
```

4 use Illuminate\Support\Facades\Route;
5
6 Route::get('/', [RecipeController::class, 'index'])->name('home');
7 Route::get('/recipe/{id}', [RecipeController::class, 'show'])->name('recipe.show');
8
9 Route::get('/dashboard', function () {

```

La página Home se nos mostrarían las 5 recetas y si se pulsa sobre el nombre de la receta se nos muestra la información de ella.

## Pantalla Home





127.0.0.1:8000

Finalizar a

Categoría:

Fecha Publicación: 23 Jun, 2024

Pizza

Margarita

Pizza Margarita

Dificultad: medio

Categoría:

Fecha Publicación: 23 Jun, 2024

Quod qui culpa qui.

Dificultad: medio

Categoría:

Fecha Publicación: 16 Mar, 2024

Est beatae qui natus enim.

Dificultad: bajo

Categoría:

Fecha Publicación: 02 May, 2024

Cupiditate perspiciatis quo.

Dificultad: alto

Categoría:

Fecha Publicación: 24 May, 2024

## Información de la receta

127.0.0.1:8000/recipe/117

DashboardHomeAPI\_recipesAPI\_recipeAPI\_category

Nesciunt possimus quia dolorum nihil.

Dificultad: alto

Categoría:

Fecha Publicacion 02 Feb, 2024

## 6. API

Modificar el fichero web.php para definir las rutas de la APIs

```
//Rutas para las APIs
Route::get('/api/recipes/{page}', [RecipeController::class, 'apiPage']);
Route::get('/api/recipe/{id}', [RecipeController::class, 'apiIndex']);
Route::get('/api/category/{id}/{page}', [CategoryController::class, 'apiIndex']);
```

Añadir las rutas de navegación en el archivo navigation.blade.php

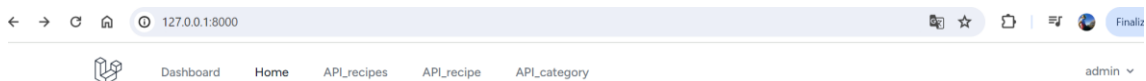
Sección navigation link

```
<x-nav-link :href="url('/api/recipes/1')" target="_blank">
|   {{ __('API_recipes') }}
</x-nav-link>
<x-nav-link :href="url('/api/recipe/1')" target="_blank">
|   {{ __('API_recipe') }}
</x-nav-link>
<x-nav-link :href="url('/api/category/1/1')" target="_blank">
|   {{ __('API_category') }}
</x-nav-link>
</div>
```

Sección responsive

```
<x-responsive-nav-link :href="url('/api/recipes/1')" target="_blank">
|   {{ __('API_recipes') }}
</x-responsive-nav-link>
<x-responsive-nav-link :href="url('/api/recipe/1')" target="_blank">
|   {{ __('API_recipe') }}
</x-responsive-nav-link>
<x-responsive-nav-link :href="url('/api/category/1/1')" target="_blank">
|   {{ __('API_category') }}
</x-responsive-nav-link>
</div>
```

Por lo que la parte de la navegación quedaría



El siguiente paso es modificar los ficheros de controller, que son **recipeController** y **CategoryController** para añadir los métodos para las APIs

Y por último si desde el menú de navegación accedemos a las opciones de las APIs podemos ver los resultados.

**Resultado de la API\_recipes.** Se nos muestra una página que va a estar en formato JSON con las recetas que forman parte de la primera página. **Dirección de la página es /api/recipes/1**

```
127.0.0.1:8000/api/recipes/1

lan formato al texto

{"current_page":1,"data":[{"id":1,"nombre":"Tacos de pollo","fecha_publicacion":"2024-06-23T13:32:08.000000Z"}, {"id":2,"nombre":"Pizza Margarita","fecha_publicacion":"2024-06-23T14:38:33.000000Z"}, {"id":3,"nombre":"Velit et tempore beatae","fecha_publicacion":"2024-02-27T03:40:16.000000Z"}, {"id":4,"nombre":"A molestiae id","fecha_publicacion":"2024-01-23T12:37:45.000000Z"}, {"id":5,"nombre":"Vel ut aut","fecha_publicacion":"2024-02-26T01:52:21.000000Z"}, {"id":6,"nombre":"Nemo voluptas velit","fecha_publicacion":"2024-02-17T17:27:11.000000Z"}, {"id":7,"nombre":"Molestiae utem","fecha_publicacion":"2024-06-03T20:30:03.000000Z"}, {"id":8,"nombre":"Aut officia natus ut","fecha_publicacion":"2024-06-23T10:05:29.000000Z"}, {"id":9,"nombre":"Rerum voluptatem doloribus mpedit","fecha_publicacion":"2024-06-17T03:40:35.000000Z"}, {"id":10,"nombre":"Excepturi minima sapiente","fecha_publicacion":"2024-02-16T02:41:59.000000Z"}], "first_page_url": "http://127.0.0.1:8000/api/recipes/1?page=1", "from": 1, "last_page": 21, "last_page_url": "http://127.0.0.1:8000/api/recipes/1?page=21", "links": [{"url": null, "label": "< Previous", "active": false}, {"url": "http://127.0.0.1:8000/api/recipes/1?page=1", "label": "1", "active": true}, {"url": "http://127.0.0.1:8000/api/recipes/1?page=2", "label": "2", "active": false}, {"url": "http://127.0.0.1:8000/api/recipes/1?page=3", "label": "3", "active": false}, {"url": "http://127.0.0.1:8000/api/recipes/1?page=4", "label": "4", "active": false}, {"url": "http://127.0.0.1:8000/api/recipes/1?page=5", "label": "5", "active": false}, {"url": "http://127.0.0.1:8000/api/recipes/1?page=6", "label": "6", "active": false}, {"url": "http://127.0.0.1:8000/api/recipes/1?page=7", "label": "7", "active": false}, {"url": "http://127.0.0.1:8000/api/recipes/1?page=8", "label": "8", "active": false}, {"url": "http://127.0.0.1:8000/api/recipes/1?page=9", "label": "9", "active": false}, {"url": "http://127.0.0.1:8000/api/recipes/1?page=10", "label": "10", "active": false}, {"url": null, "label": "...", "active": false}, {"url": "http://127.0.0.1:8000/api/recipes/1?page=20", "label": "20", "active": false}, {"url": "http://127.0.0.1:8000/api/recipes/1?page=21", "label": "21", "active": false}, {"url": "http://127.0.0.1:8000/api/recipes/1?page=2", "label": "Next", "active": false}], "next_page_url": "http://127.0.0.1:8000/api/recipes/1?page=2", "path": "http://127.0.0.1:8000/api/recipes/1", "per_page": 10, "prev_page_url": null, "to": 10, "total": 202}
```

**Resultado de la API\_recipe.** Se nos muestra una página que va a estar en formato JSON con la receta que tiene el id 1. **Dirección de la página es /api/recipe/1**

```
127.0.0.1:8000/api/recipe/1

lan formato al texto

{"id":1,"nombre":"Tacos de pollo","fecha_publicacion":"2024-06-23T13:32:08.000000Z","nivel_dificultad":"medio","tiempo_preparacion":45,"ingredientes":"Tortillas de ma\u00edz, pollo deshebrado, salsa picante, queso fresco, crema", "autor": "Chef Maria", "instrucciones_preparacion": "Cocina y desmenuza el pollo. Calienta las tortillas y rell\u00e9nalas con pollo desmenuzado. A\u00f1ade salsa, guacamole, cebolla y cilantro al gusto. Sirve y disfruta de estos tacos sabrosos y f\u00faciles de preparar, ideales para cualquier ocasi\u00f3n.", "imagen": null, "created_at": "2024-06-13T13:32:08.000000Z", "updated_at": "2024-06-23T13:32:08.000000Z"}
```

## 7. PUBLICACIÓN EN INTERNET

El código implementado junto con este documento se ha subido a un repositorio en Github.

[https://github.com/mariacanas/R4\\_BackEnd](https://github.com/mariacanas/R4_BackEnd)

## 8. EXPERIENCIA CON LOS ENTORNOS

### DRUPAL (PEC2)

#### Ventajas :

- Cuenta con una gran cantidad de ayuda por internet, ya que tiene una comunidad bastante grande
- Flexibilidad en cuanto a la configuración, ya que puede ser utilizado para desarrollar muchos tipos de sitios web
- Proporciona herramientas para la gestión de contenidos permitiéndonos crear estructuras de contenido más complejas

#### Inconvenientes :

- De los 3 entornos es el que más tiempo me llevo aprender su funcionamiento
- Depende mucho de recursos del servidor a diferencia de los demás

-

## **PHP (PEC3)**

### **Ventajas :**

- PHP cuenta con una gran cantidad de información en internet, siendo de gran ayuda para desarrollar los sitios web
- De los 3 es el que más había utilizado por lo que me resultó más fácil su uso e implementación
- Es compatible con la mayoría de base de datos por lo que facilita su implementación independientemente de la que se use

### **Inconvenientes :**

- Si desde el principio no se hace una buena estructura del código, puede volverse algo difícil de mantener
- En el caso de que el proyecto sea de gran tamaño, su mantenimiento y gestión puede resultar difícil
- Si no se hace uso de buenas prácticas, php puede ser vulnerable a ataques por lo que fallaría en la rama de la seguridad

## **LARAVEL (PEC4)**

### **Ventajas :**

- Facilita mantener un código organizado
- Laravel nos ofrece un sistema de routing fácil de usar por lo que para definir las rutas en la aplicación va a ser fácil de implementar
- Cuenta con una documentación grande y bien organizada, facilitándonos encontrar la información

### **Inconvenientes :**

- Sufre de actualizaciones frecuentes, por lo que se va a necesitar más tiempo en mantenerlo al día
- En cuanto al rendimiento, va a tener un mayor consumo de recursos del servidor por lo que los tiempos de ejecución pueden ser más lentos
- La dependencia de Composer

